greatly differs from the random search, because it provides higher rewards to a diverse behavior. There is also an archive, which stores the previously explored areas of the search space. The novelty score is then counted as the average distance from the k-nearest neighbors. However, the distance metric is problem-dependent, which could be problematic (Lehman and Stanley, 2011a). Another interesting idea, which promotes the novel behavior, is the Behavior characterization algorithm (Meyerson et al., 2016). This algorithm maps each individual behavior to a vector space. The search is then driven towards diversity in a metric space of these behaviors. Another approach to this problem is Curiosity learning. It works with intrinsic rewards that promote exploration (Pathak et al., 2017). Building on Novelty Search, Surprise Search was later introduced as another divergent search technique working with the notion of surprise. It achieves an efficiency comparable to the Novelty Search and is also able to find solutions more frequently (Gravina et al., 2016).

The ability to generate a diverse set of high-performing solutions in evolutionary algorithms is crucial and has been addressed in many so-called Quality-Diversity algorithms, such as the MAP-Elites algorithm (Mouret and Clune, 2015) or NS with Local Competition (Lehman and Stanley, 2011b). In MAP-Elites (Mouret and Clune, 2015), the search space is discretized into unique regions, for each of which the best-performing solution is identified. The selection of the fittest individuals withing a population is restricted to a specific feature, the fittest individuals are kept in a multidimensional archive and only replaced if they are outperformed. This way a diverse group of well-performing individuals with different features can develop in the domain space in a single run. However, the problem of local optima within the elite may still occur over time and the scalability to high-dimensional spaces is also problematic due to the exponential growth of the regions with increasing dimensionality. This problem is overcome by CVT-MAP-Elites algorithm (Vassiliades et al., 2016), which extends the MAP-Elites algorithm and proposes a solution to reduce the number of regions in high-dimensional spaces.

The problem of the deceptive local optima arises not just in evolutionary algorithms, but also in reinforcement learning itself. The reinforcement approaches optimize only for the reward (e.g. DQN (Mnih et al., 2013), A3C (Mnih et al., 2016)) often fail to learn behaviors and strategies to overcome the local optima and solve the task at hand (Jackson and Daley, 2019). To solve this issue, Novelty Search and other approaches such as episodic curiosity (Savinov et al., 2019) were explored. In (Jackson and Daley, 2019) instead of measuring the final position of the agent and comparing it to other individuals of the current generation and the ones in the archive, the Levenshtein distance between the sets of actions performed by the agents was used to measure the extent of behavioral novelty. The approach of comparing the sequences of actions instead of the final positions removes the need to detect the exact position of an agent for each frame, and therefore makes it easier to generalize Novelty Search to any Atari game.

Our approach: Sugar Search

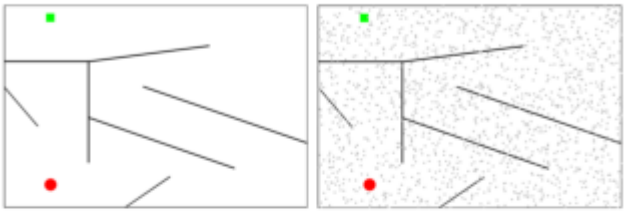


Figure 1: Visualization of the hard map with (on the right) or without the sugars (on the left). The red dot is the starting point, the green dot is the goal, and the black lines are the obstacles. The sugars are represented as gray rectangles, they are placed randomly, and their density is the only hyperparameter. Only the agent who collects the sugar first is rewarded for it.

Even though Novelty Search is conceptually interesting, it seems to employ a certain degree of supervision, which is needed to measure and decide how unique each agent's behavior is. But this is inconsistent with what we observe in nature, where it is rather the environment that formulates, motivates, and rewards individuals for being different.

This difference can be demonstrated on a maze problem specified as follows. A maze contains multiple obstacles and a target. In the environment, there are agents whose goal is to reach the target while avoiding obstacles. If an agent reaches an obstacle, its movement stops.

In the Novelty Search algorithm, the agents that stop at a point far away from the others will receive a higher fitness, which motivates them to explore the maze. On the downside, this requires us to compute the k-nearest neighbors for each agent.

Our approach is fairly different. We propose a method called Sugar Search, which introduces sugar objects as a form of a reward. Sugars are distributed randomly (and uniformly) over the map, and an agent is rewarded only if he is the first one to collect a sugar object. This way the only form of supervision is the sugar placement and, compared to Novelty Search, we can avoid the costly computation of the k-nearest neighbors and eliminate the archive completely. The concept of the sugar-based search leads to a different pattern in the exploration of the search space than Novelty Search does.

Our fitness function p for an agent x is defined by:

$$p(x) = \sum_{i=1}^{n} collect(x, s_i)$$

where n is the total number of sugars, s_i is the i-th sugar. And the function $collect$ returns 1 if s_i was collected by the agent x, otherwise it returns 0. Once a sugar is collected by a specific agent, no other agent can collect it and get rewarded again, also sugars do not re-spawn. The main loop of the algorithm is described in depth in Neural Network configuration.

The visualization of the sugar placement in a maze is shown in Figure 1, where the hard map that was used in our experiments is depicted. It is important to highlight that the sugars are placed randomly and their density is the only hyperparameter.

Pixel Novelty

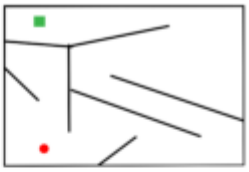
Naturally, we started thinking of ways to generalize Sugar Search in the games, where we do not have access to the position of the agent. A good example of this problem are Atari Games, where the only information we have is the screen. Therefore, agent localization and tracking are difficult. To overcome this problem, we propose Pixel Novelty, a generalization of our Sugar Search. Instead of rewarding the agent for collecting the sugar first, we will reward him for generating a new set of pixels (new screen content) first. Every time this occurs, the agent's fitness value is increased and the pixel compositions of the new screen are stored for future comparisons. If this or any other agent creates the same screen in the current generation, he is not rewarded again. Importantly, the archive of screens is reset after each generation, therefore in the next run, the same logic is applied. This idea promotes the agents to act in an unexpected, new, and never-before-experienced way, and it also solves the problem of agent localization and tracking. Eliminating this process is expected to reduce the computational time significantly.

Experiments

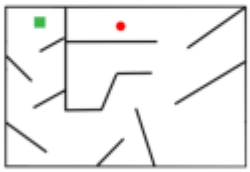
Maze navigation task

To prove our suggested approach - Sugar Search and to see how it compares to the objective-based search and Novelty Search, we decided to test these algorithms on the maze environments. These environments, proposed in (Lehman and Stanley, 2011a), are widely used due to the fact that they contain many dead ends and local optima. Algorithms that are optimizing the distance to the goal may be caught in a local minimum and thus perform worse than the divergent algorithms like ours.

(a) Medium map. A non-trivial wavy movement is required to reach the goal.



(b) Hard map. Agents have to move further away from the goal in order to reach it.



(c) Super-hard map. It introduces more dead ends and a complex movement of the agent is needed to pass it successfully.

Figure 2: Medium, hard and super-hard map. The red dot is the starting point, the green dot is the goal, and the black lines are the obstacles.

The maze navigation task takes place in a set of 3 maze environments of increasing difficulty, which are shown in Figure 2. The maps of these mazes were designed to correspond with the mazes from the original Novelty Search paper (Lehman and Stanley, 2011a). The medium map in Figure 2a has deceptive dead ends that can lead the agent to a local minimum. To overcome this problem, the agent has to develop a wavy movement to get through the obstacles to the goal. The second maze, the hard map, is more difficult because the agent has to move completely away from the goal in order to reach it, as we can observe in Figure 2b.

Finally, we have also constructed the super-hard maze shown in Figure 2c, which is a combination of the medium and the hard mazes. It contains deceptive ends and also requires a non-trivial movement to reach the goal.

For the maze navigation task, sugars were distributed randomly and uniformly over the environment with a density of 0.3 per pixel as listed in Table 1. In the fitness-based search, a distance to the goal was used to determine the fitness value of the agent. In the Novelty Search technique, 15-nearest neighbors were used.

All experiments were done under the same conditions and all the configuration specifics are listed in Table 1. The agent that gets close enough to the goal, within a fixed amount of time, is considered to be the solution.

Throughout the experiments, we use agents that are controlled by a neural network (NN) shown in Figure 3. The goal of an agent is to develop such a NN that it can navigate him from the starting point to the endpoint of the maze before the timeout. Each agent is equipped with 8 sensors in total: 4 range finder sensors, which show the distance to

the nearest obstacle on each side, and 4 radar sensors, which indicate the direction to the goal. The value of these sensors is normalized and used as the input to the neural network, which will evaluate whether the agent should move up, down, left or right. The specifics of the NN that was used throughout all the experiments are further described in Neural Network configuration.

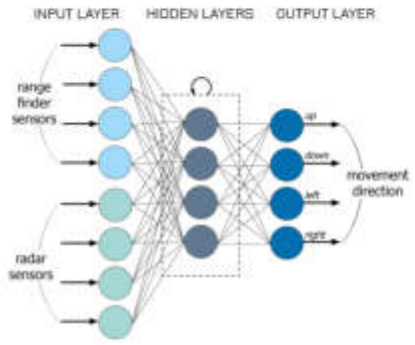


Figure 3: Agent's neural network in the maze navigation task.

Atari games task

The maze navigation task is a well-suited environment to test our approach and compare it to other works such as Novelty Search. More advanced problems such as walking robots could show the same results.

Further, we decided to implement this idea and generalize it to the environment of Atari games. Atari games have been widely used to compare many machine learning algorithms, especially deep reinforcement learning algorithms (Mnih et al., 2013). But due to the many implementations of Atari games, it is difficult to compare the results of different algorithms. That is why the OpenAI Gym (Brockman et al., 2016) games were chosen at the beginning to avoid any differences in the underlying environment and to see if our ideas could compare with some of the famous algorithms (Mnih et al., 2013)(Young et al., 2019).

Later on, due to the difficulties with the agent localization and tracking in the OpenAI Gym (Brockman et al., 2016), we have decided to test our Sugar Search approach on the MinAtar (Young and Tian, 2019). MinAtar is inspired by the Arcade Learning Environment (Bellemare et al., 2012) but simplifies the games to make experiments with the environments more accessible and efficient. Currently, MinAtar provides analogues to 5 Atari games, which we have extended by Ms. Pacman and Montezuma's revenge implementation. The environments provide a state representation, where each of the n channels corresponds to a game-specific object, such as the ball, the paddle, and the brick in the game Breakout. Thanks to this environment, we were able to track the agent and perform Sugar Search.

The goal of the agent is similar to the maze navigation task: to develop such NN that it can achieve the highest score possible. In the MinAtar environment, the agent's input consists of all the state representations of the objects. In the OpenAI gym, the input is the screen pixels. However, for computational reasons, the screen resolution was down-sampled 8 times to $(210/8) \times (160/8)$ pixels, and the RGB colors were converted into gray-scale. This alteration resulted in 520 input neurons used in the experiments, from the original 100,800.

All the necessary configuration parameters used in the Atari games task are included in Table 1. In the fitness-based search, a score was used to determine the fitness value of the agent. In the Novelty Search technique, 5-nearest neighbors were used. However, to reach even better performance, we had to modify the idea of our approach slightly by re-spawning a new sugar during the game. Because in some Atari games, e.g. Ms. Pacman, the agent has no finish point and its goal is to collect as many points as possible before getting killed. This requires traveling to places that have been explored in the past but may be beneficial to revisit in the present moment. To motivate the agent to behave in such a way, we decided to re-spawn a sugar every l frame.

Neural Network configuration

Table 1: The experiment configuration for both the Maze navigation task and the Atari games of our Sugar Search with RNN.

Parameters	***Maze env.***	***Atari env.***
Population size	250	75
Max. generations	1200	10000
Time frame (max. frames)	600	250
Sugars density per pixel	0.3	0.3
l re-spawn time frame	-	5
Hidden layer	1	1
Hidden neurons	32	32
Activation function	RelU	RelU
Gaussian noise	$\mathcal{N}(0, 0.1^2)$	$\mathcal{N}(0, 0.1^2)$

Each agent has to develop a neural network, which will allow him to achieve the goal. The challenge is how to develop this neural network. In the original paper (Lehman and Stanley, 2011a) NEAT, an evolutionary algorithm was used (Stanley and Miikkulainen, 2002). However, it has numerous hyperparameters which are difficult to optimize. Instead, we used Elman recurrent neural network (Elman, 1990) (RNN), which is a simple NN with a recurrent hidden layer. We used random Gaussian noise to mutate its weights.

In our algorithm, we randomly initialize weights to the RNN for each agent. When the time limit is reached, we evaluate 10% of the best agents and keep them for the next generation. The rest of the population is modified - random Gaussian noise is added to its weights. After this mutation, the agents are added to the next generation. This approach is much simpler and results in fewer hyperparameters.

The benefits of our concept of the NN are obvious from the experiments on the medium map, where NEAT needed 19.8 generations on average to finish the maze, whereas the simple Elman neural network (Elman, 1990) with our evolutionary algorithm only needed 10.4 generations (these results were averaged from 50 runs). This proves that our approach is much better, and for these reasons we used it in all of our experiments. All the other parameters necessary for the reproduction of the experiments are listed in Table 1.

Results

Maze navigation task

We have evaluated the results from the least to the most complex map: the medium map, then the hard map, and finally the super-hard map in Figure 2. All the results are shown in detail in Table 2. The level of success of the search techniques is measured by the average number of generations needed to reach the goal and the number of successfully finished runs out of 50 (meaning runs where we were able to reach the goal). Note that we use the same way to reference the maps as in the Novelty Search (Lehman and Stanley, 2011a) paper to avoid confusion.

On the medium map, Novelty Search and our Sugar Search performed similarly, while the fitness-based search was approximately two times slower. All of these techniques were able to successfully finish all 50 out of 50 runs. The difference between divergent searches and fitness-based search is statistically significant ($p < 0.01$; Student's t-test). On the hard map, where the deceptiveness of the environment is more compelling, a remarkable difference between all three examined algorithms can be distinguished. Sugar Search performed significantly better than Novelty Search in terms of the success rate and the number of generations needed, as shown in Table 2. The difference between Sugar Search and Novelty Search is statistically significant ($p < 0.05$; Student's t-test). Both divergent approaches largely outperformed the fitness-based search, which was unable to reach the goal in any run.

Similar results were observed on the super-hard map. The fitness-based approach failed to reach the goal in any run again. The divergent approaches were able to solve this maze with a reasonable success rate: 14 for Novelty Search versus 24 for Sugar Search, and with an average number of generations needed: 345.3 for Novelty Search versus 280.4 for Sugar Search. This difference is not statistically significant ($p < 0.05$; Student's t-test). A larger sample of runs on this map would be needed to confirm the significant difference of our Sugar Search outperforming Novelty Search, which can be observed in the results.

Table 2: Maze navigation task performance in the average number of generations needed to reach the goal and the number of successfully finished runs out of 50 (meaning runs where we were able to reach the goal). The standard deviation is in parentheses. Bold font indicates the best performance for each metric. All numbers are averaged on the successful runs only. Symbols ↑ (↓) represent that the higher (lower) the better.

Maze Results	**Generations ↓**			**Finished runs (out of 50) ↑**		
	med. map	***hard map***	***super-hard map***	***med. map***	***hard map***	***super-hard map***
Fitness-based approach	17.6 (9.18)	-	-	50	0	0
Novelty approach	**10.4** (5.08)	237.7 (172.12)	345.3 (183.42)	50	23	14
Sugar Search (ours)	11.7 (5.96)	**153.3** (97.53)	**280.4** (101.78)	50	**35**	**24**

Reward density

Naturally, while working on the maze problem, an interesting question arose. How does the density of the reward distribution on the map affect the results? Our hypothesis was that having as many sugars as there are pixels on the map would produce the best results since theoretically a higher density of sugars should lead agents to the goal more efficiently.

From a series of experiments on the medium map with a varying sugar density parameter as shown in Figure 4, we have proven this hypothesis valid. Placing sugar on every pixel produces the best results, and reducing the sugar density generally slows down the process of reaching the goal. Interestingly, performing a small-scale reduction of the reward density still produces comparable results.

These findings can be beneficial when applied to a similar problem defined in a continuous space, where the sugar placement cannot be realized per pixel for computational reasons.

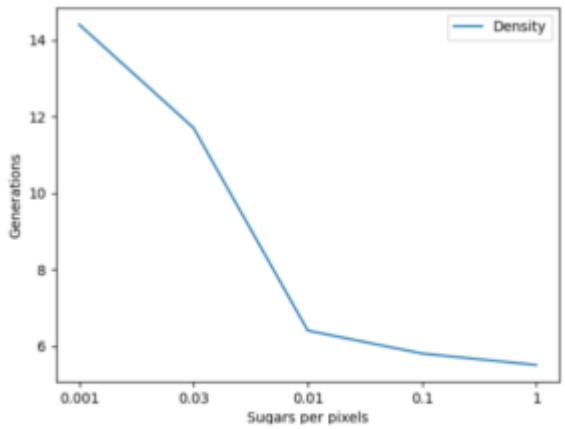


Figure 4: Average number of generations needed to reach the goal (*the lower the better*) and the sugars per pixel density trade-off curve.

Combining Sugar Search with the fitness-based approach

We could observe that in a deceptive maze problem, the divergent search, especially our Sugar Search, completely outperforms the objective-driven search. But what if we combined these two approaches?

We wanted to explore this idea further and decided to design a combined approach, where the agent is rewarded not only for being novel (divergent search) but also for moving towards the goal (fitness-based search). This combined approach requires us to define the weights of each component that would produce the best results, meaning we need to find an optimal alpha.

$$p(x) = \alpha * distance + (1 - \alpha) * sugars$$

In order to find an optimal alpha and observe how this weighted approach stands in comparison with the original Sugar Search and other techniques, we have executed a series of experiments on the medium and the hard maps. To reduce the computation time, as a proof of concept, we have excluded the super-hard map from the following experiments. Based on the results of the basic Sugar Search algorithm, we can expect to observe similar trends on the super-hard map as the hard map.

From the results shown in Table 3, we can conclude that the combined approach leads to improved performance, specifically in the average number of generations needed to reach the goal. On the other hand, an improvement is not observable in the agent's success rate. It was 29.3 on average on the hard map, which is slightly worse than Sugar Search alone.

Table 3: Maze navigation task with the weighted approach: performance in the average number of generations needed to reach the goal and the number of successfully finished runs out of 50 (meaning runs where we were able to reach the goal). The standard deviation is in parentheses. Bold font indicates the best performance for each metric. All numbers are averaged on the successful runs only. Symbols ↑ (↓) represent that the higher (lower) the better.

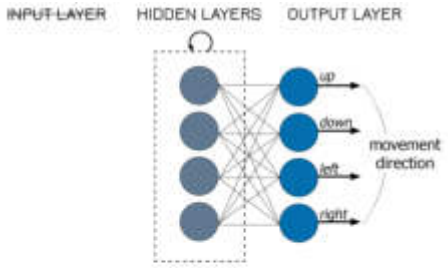
Maze Results	**Generations ↓**		**Finished runs (out of 50) ↑**	
	medium map	***hard map***	***medium map***	***hard map***
Fitness-based approach	17.6 (9.18)	-	50	0
Novelty approach approach	**10.4** (5.08)	237.7 (172.12)	50	23
Sugars approach	11.7 (5.96)	153.3 (97.53)	50	**35**
Weighted with alpha 1/4	8.0 (7.97)	148.7 (104.25)	50	32
Weighted with alpha 2/4	**5.4** (4.09)	**122.4** (86.55)	50	30
Weighted with alpha 3/4	6.7 (5.59)	139.1 (90.87)	50	26

Agents without sensors

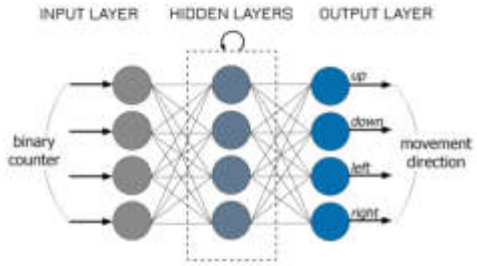
The idea behind this section is to prove that the agents are really processing the information from their sensors to avoid hitting an obstacle and that they are not simply memorizing a sequence of optimal actions instead. This problem could arise when we take into consideration the amount of memorization and the overall generalization in terms of the agents' performance in the mazes.

To prove this, we have decided to experiment with a blind agent. The original agents were blinded - meaning they had all their inputs removed and had to rely solely on their memory in the hidden layer, as shown in Figure 5a. Later, we have modified this concept further in order to elevate the memory and created a modified blind agent that had a simple binary counter as its input as shown in Figure 5b. This binary counter is a timer, which helps the agent memorize sequences of actions.

From Table 4 we can see that the blind agents can navigate throughout the map successfully without having any information about the environment, simply by memorizing a sequence of actions. However, having information from the sensors about the nearest obstacle and the location of the goal significantly accelerates the process of reaching the



(a) A blind agent relies solely on the memory in the hidden layer.



(b) A modified blind agent with a simple binary counter on the input.

Figure 5: Modified agents without the sensors and their NN.

goal. It is also important to note that the blind agents are not adaptable and will fail completely when placed in a slightly different starting point or if the task is modified.

Table 4: Maze navigation task for agents without sensors: performance in the average number of generations needed to reach the goal. The standard deviation is in parentheses. Bold font indicates the best performance for each metric. Symbols ↑ (↓) represent that the higher (lower) the better.

Maze Results	***Sensors* ↓**	***Binary counter* ↓**	***No input* ↓**
medium map	**11.7** (5.96)	37.8 (50.88)	96.1 (36.91)
hard map	**153.3** (97.53)	597.4 (354.65)	1154.7 (592.78)

MinAtar results

In this environment, we ran experiments on 7 different Atari games to test and compare the fitness-based approach, Sugar Search, and its generalization - Pixel Novelty to other techniques like Novelty Search (Lehman and Stanley, 2011a) or DQN (Mnih et al., 2013). The results in Table 5 show that Sugar Search does not always lead to a higher score compared to the fitness-based approach, but it performs comparably well with Novelty Search consistently across different games, as shown in Table 5. The scores achieved with Pixel Novelty are also competitive or comparable to other approaches, meaning that Pixel Novelty is a valid generalization of Sugar Search.

From the results in Table 5, we can conclude that the achieved scores differ based on the characteristics of each game, where some approaches seem to be more fitted than others. Generally speaking, we can observe a pattern in Atari games with no distinctive local optima, such as Breakout or Space Invaders, where a simple fitness-based search outperforms both our Sugar Search and Novelty Search. Interestingly, Pixel Novelty also performs rather well in these games (it even achieved the highest score in Breakout).

Regarding Sugar Search, it outperformed both the fitness-based search and Novelty Search by far in games like Freeway or Ms. Pacman and was even able to compete with algorithms like DQN (Mnih et al., 2013) despite its simplicity. This demonstrates that our search technique is very powerful and able to produce competitive results.

Table 5: MinAtar games performance in the average number of the achieved game score. Bold font indicates the best performance for each metric. All numbers are averaged from 100 generations/episodes. Symbols ↑ (↓) represent that the higher (lower) the better.

Game	***Fitness-based Search* ↑**	***Sugar Search (ours)* ↑**	***Novelty Search* ↑**	***Pixel Novelty (ours)* ↑**	***DQN* ↑**
Seaquest	8	7	6	5	**20**
Breakout	13	5	5	**21**	9
Asterix	12	10	14	10	17
Freeway	16	**53**	49	28	50
S. Invaders	43	10	14	21	**45**
MsPacman	1897	**3580**	1962	2480	1099
Montezuma's revenge	0	41	22	22	**46**

OpenAI Gym results

In the OpenAI Gym environment, experiments on 12 Atari games in total were performed with both Pixel Novelty and the fitness-based approach. The specific games to be included in the experiments were selected to represent equal subsets of games, half of which contains multiple obvious local minima (e.g. Montezuma's revenge) and is expected to show the advantages of our approach. The results of the experiments are presented in Table 6 in comparison with the fitness-based approach and reinforcement learning

algorithms like DQN (Mnih et al., 2013) and A3C algorithm (Mnih et al., 2016).

Table 6: OpenAI Gym games performance in the average number of the achieved game score. Bold font indicates the best performance for each metric. All numbers are averaged from 100 generations/episodes. Symbols ↑ (↓) represent that the higher (lower) the better.

Game	***Fitness-based*** ↑	***Pixel Novelty (ours)*** ↑	***DQN*** ↑	***A3C FF*** ↑
BeamRider	1188.6	900.4	8672.4	**13235.9**
Breakout	15.4	24.1	303.9	**551.6**
Enduro	86.9	52.2	**475.7**	82.2
Pong	21.0	**24.3**	16.2	11.4
Seaquest	780.4	480.3	**2793.9**	2300.2
S. Invaders	1075.7	1205.3	1449.7	**2214.7**
Riverraid	2390.5	1810.7	4065.3	**10001.2**
Freeway	**30.0**	0.4	25.8	0.1
Gravitar	**2350.4**	700.6	216.5	269.5
Zaxxon	**6240.1**	5400.4	831.0	2659.0
Venture	**440.5**	200.3	54.0	19.0
Montezuma's revenge	0.0	10.0	50.0	**53.0**

In the light of the results, rewarding the agents solely for generating new pixels in Pixel Novelty leads to the same or better results as the score-driven reward system in the fitness-based approach. These overall positive results are the outcome of the agent's behavior in Pixel Novelty: as the agent tries to generate a new unseen screen content, he conquers obstacles on the way and by doing so also achieves a reasonable score.

Additionally, when compared to DQN and A3C algorithms, we are able to achieve a higher score in 5 out of 12 games. Furthermore, it is important to highlight that our results were achieved with a significantly greater efficiency, since we only needed 20 million frames and 1 CPU for 24 hours, whereas the results from DQN and A3C were trained on 320 million frames, which is 16 times more.

We have proven that our simple evolutionary technique is a scalable competitive alternative to the reinforcement-learning algorithms (Salimans et al., 2017) and that it can achieve similar results even with a divergent-driven approach and with a largely improved efficiency.

Conclusion

The idea of placing rewards within the environment to precede the problem of local minima performs surprisingly well in the maze experiments, especially taking into consideration that this concept completely disregards the objective (score). As the experiments have shown, our technique outperformed the popular Novelty Search (Lehman and Stanley, 2011a) in both the success rate and the solution speed, furthermore, it is also conceptually simpler and easier to implement. Following this proof of concept of our technique, we have also explored the effect of multiple local minima and different reward densities on the search efficiency. The results indicate that the sparse rewards model could be applied to problems in a continuous or a higher-dimensional space.

We have subsequently generalized our search technique and studied how it performs in a more general and popular set of tasks - Atari games. Our technique performed comparably well with a basic fitness-based search and some of the most common reinforcement learning algorithms (DQN (Mnih et al., 2013), A3C (Mnih et al., 2016)). We consider it to be a great achievement considering that our results were computed on much fewer frames than DQN or A3C.

In summary, we believe that it is rather the environment that formulates, motivates, and rewards individuals for being different and novel as we can observe in nature. Also supported by the fact that the concept has been proven valid on multiple experiments in this paper. This suggests that new ways of defining novelty, which avoid any form of supervision, can generally be successful and could be worth exploring in the future.

Acknowledgements

We would like to thank GoodAI for the research funding and valuable advice. We would also like to express appreciation to Daniela Hradilova for the useful discussions and suggestions. Our work is part of the RICAIP project that has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 857306.

References

Basu, S. K. and Bhatia, A. K. (2006). Mitigating deception in genetic search through suitable coding. In *Proceedings of the 13th International Conference on Neural Information Processing - Volume Part III*, ICONIP'06, page 946–953, Berlin, Heidelberg. Springer-Verlag.

Bellemare, M., Naddaf, Y., Veness, J., and Bowling, M. (2012). The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research*, 47.

Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., and Zaremba, W. (2016). Openai gym. *arXiv preprint arXiv:1606.01540*.

Elman, J. L. (1990). Finding structure in time. *Cognitive Science*, 14(2):179–211.

Goldberg, D. E. (1987). Simple genetic algorithms and the minimal, deceptive problem.

Goldberg, D. E. and Holland, J. H. (1988). Genetic algorithms and machine learning. *Machine Learning*, 3(2):95–99.

Gravina, D., Liapis, A., and Yannakakis, G. (2016). Surprise search: Beyond objectives and novelty. In *Proceedings of the Genetic and Evolutionary Computation Conference 2016*, GECCO '16, page 677–684, New York, NY, USA. Association for Computing Machinery.

Hutter, M. and Legg, S. (2006). Fitness uniform optimization. *IEEE Transactions on Evolutionary Computation*, 10(5):568–589.

Jackson, E. C. and Daley, M. (2019). Novelty search for deep reinforcement learning policy network weights by action sequence edit metric distance.

Lehman, J. and Stanley, K. (2011a). Abandoning objectives: Evolution through the search for novelty alone. *Evolutionary computation*, 19:189–223.

Lehman, J. and Stanley, K. O. (2011b). Evolving a diversity of creatures through novelty search and local competition.

Meyerson, E., Lehman, J., and Miikkulainen, R. (2016). Learning behavior characterizations for novelty search. In *Proceedings of the Genetic and Evolutionary Computation Conference 2016*, GECCO '16, page 149–156, New York, NY, USA. Association for Computing Machinery.

Mnih, V., Badia, A. P., Mirza, M., Graves, A., Lillicrap, T., Harley, T., Silver, D., and Kavukcuoglu, K. (2016). Asynchronous methods for deep reinforcement learning. In Balcan, M. F. and Weinberger, K. Q., editors, *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pages 1928–1937, New York, New York, USA. PMLR.

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., and Riedmiller, M. (2013). Playing atari with deep reinforcement learning.

Mouret, J. and Clune, J. (2015). Illuminating search spaces by mapping elites. *CoRR*, abs/1504.04909.

Pathak, D., Agrawal, P., Efros, A. A., and Darrell, T. (2017). Curiosity-driven exploration by self-supervised prediction. ICML'17, page 2778–2787. JMLR.org.

Salimans, T., Ho, J., Chen, X., Sidor, S., and Sutskever, I. (2017). Evolution strategies as a scalable alternative to reinforcement learning.

Savinov, N., Raichuk, A., Marinier, R., Vincent, D., Pollefeys, M., Lillicrap, T., and Gelly, S. (2019). Episodic curiosity through reachability.

Stanley, K. O. and Miikkulainen, R. (2002). Evolving neural networks through augmenting topologies. *Evolutionary Computation*, 10(2):99–127.

Vassiliades, V., Chatzilygeroudis, K. I., and Mouret, J. (2016). Scaling up map-elites using centroidal voronoi tessellations. *CoRR*, abs/1610.05729.

Young, K. and Tian, T. (2019). Minatar: An atari-inspired testbed for thorough and reproducible reinforcement learning experiments. *arXiv preprint arXiv:1903.03176*.

Young, K., Wang, B., and Taylor, M. E. (2019). Metatrace actor-critic: Online step-size tuning by meta-gradient descent for reinforcement learning control.

Paths in a Network of Polydisperse Spherical Droplets

Johannes Josef Schneider[1,2], Alessia Faggian[3], Silvia Holler[3], Federica Casiraghi[3], Jin Li[2], Lorena Cebolla Sanahuja[3], Hans-Georg Matuttis[4], Martin Michael Hanczyc[3,5], David Anthony Barrow[2], Mathias Sebastian Weyland[1], Dandolo Flumini[1], Peter Eggenberger Hotz[1], Patrik Eschle[1], Aitor Patiño Diaz[3], and Rudolf Marcel Füchslin[1,6]

[1]Institute of Applied Mathematics and Physics, School of Engineering, Zurich University of Applied Sciences, Technikumstr. 9, 8401 Winterthur, Switzerland
[2]School of Engineering, Cardiff University, Queen's Buildings, 14-17 The Parade, Cardiff CF24 3AA, Wales, United Kingdom
[3]Laboratory for Artificial Biology, Department of Cellular, Computational and Integrative Biology, University of Trento, Via Sommarive, 9 - 38123 Povo, Italy
[4]Department of Mechanical Engineering and Intelligent Systems, The University of Electrocommunications, Chofu Chofugaoka 1-5-1, Tokyo 182-8585, Japan
[5]Chemical and Biological Engineering, University of New Mexico, MSC01 1120, Albuquerque, NM, 87131-0001, USA
[6]European Centre for Living Technology, S.Marco 2940, 30124 Venice, Italy
scnj@zhaw.ch, johannesjosefschneider@googlemail.com

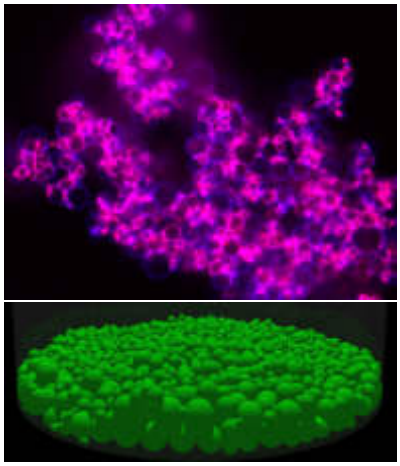


Figure 1: Top: Snapshot of an experimentally found cluster of oil droplets in water recorded with a confocal microscope. Bottom: Final configuration of 2,000 multidisperse particles at the bottom of a cylinder in a computer simulation.

Abstract

We simulate the movement and agglomeration of oil droplets in water under constraints, like confinement, using a simplified stochastic-hydrodynamic model. In the analysis of the network created by the droplets in the agglomeration, we focus on the paths between pairs of droplets and compare the computational results for various system sizes.

Introduction

Spatial arrangements of hard spheres are widely studied in physics, as these systems serve as simple models for granular matter, colloidal systems, and molecular crystals (Reiss et al., 1996; Russel et al., 1989; Mitarai and Nakanishi, 2003; Metcalfe et al., 1995; Zallen, 1998). Mostly, monodisperse and bidisperse systems are considered, i.e., all spheres exhibit the same radius value or one of two different values, but sometimes also multidisperse systems with radii chosen from a finite set of prespecified radius values have been investigated (Müller et al., 2009; Schneider et al., 2009). Also packings of non-spherical items for which the determination of overlaps is more difficult, such as ellipsoids (Pfleiderer and Schilling, 2007; Matuttis and Chen, 2014) and spherocylinders, have been studied. It has, e.g., been found that a random packing of ellipsoids with a specific aspect ratio (M & M candies) is denser than a random packing of spheres (Donev et al., 2004). Furthermore, arrangements of particles with long-range interactions, in confinement, and under constraints, such as shear forces and repulsive walls, (Ricci et al., 2007; Ochoa et al., 2006) have been investigated.

In our collaboration, we intend to develop a probabilistic compiler (Flumini et al., 2020; Weyland et al., 2020) to aid the three-dimensional agglomeration of droplets filled with various chemicals in a specific way in order to e.g. allow the creation of desired macromolecules via a successive reaction scheme (Schneider et al., 2020a,b). Neighboring droplets can form connections, either by forming bilayers (Li and Barrow, 2017) or by getting glued to each other by matching pairs of single-stranded DNA (Hadorn et al., 2012). Chemicals contained within the droplets can move to neighboring droplets either directly, as hydrophobic compounds can be exchanged between adjacent oil droplets at the contact face, or, if the oil droplets are contained in a hull comprised of amphiphilic molecules like phospholipids, through pores within these bilayers. Thus, a complex bi-

layer network is created, with the droplets being the nodes of this graph and the existing connections being the edges between the corresponding droplets. In such bilayer networks, a controlled successive reaction scheme can be effectuated to produce the intended macromolecules. Within the scope of this paper, we present computational results for basic simulations of a simplified agglomeration process of a polydisperse system of droplets, mimicking experiments. Here we want to focus on the question which influence the density of the droplets has on the agglomeration process of the particles and on some specific properties of the networks created, which are of crucial importance for the gradual reaction scheme intended. In order to focus on these questions and to exclude effects from other experimental properties, we simulate the droplets as hard spheres and ignore details of the surface structure of the particles, attractive forces as well as adhesion effects. As the extension of the bilayers is very small and as due to their small radii (Aprin et al., 2015), the droplets keep their spherical shape during the experiments, as shown in Fig. 1, this simplified approach is justified.

Networks can be in general analyzed

- either on an elementary level, i.e., by considering properties of the single nodes within a network, like the degree of a node (the number of nodes a node is attached to via edges),
- or by considering groups of nodes, e.g., by finding cliques of nodes which are fully connected with each other within such a group (Marino and Kirkpatrick, 2018),
- or by taking the overall network into account, e.g., by determining whether the network is dominated by a large cluster and even percolating (Stauffer and Aharony, 1994; Stauffer, 1986; Naftaly et al., 1991),
- or by taking a local-global attitude, e.g., by investigating the role some specific nodes play for the overall network (Schneider and Kirkpatrick, 2005).

Often random networks are considered in which nodes are connected with randomly selected edges (Bollobás, 2011). But in real-world networks, like genetic networks or the World Wide Web, the degrees of the nodes often follow a scale-free power-law distribution (Barabási and Albert, 1999) due to the tendency that a newly added node preferentially attaches itself to nodes with higher degree. In contrast, Gaussian distributions are found for degree numbers in random networks. We are mainly interested in the time evolution of the network formed by the connections between the various droplets. In this paper, we focus on the paths between pairs of droplets, i.e., on the question whether paths between them exist and, if yes, how large the geodesic distances between them are. In some applications of artificial chemistry we intend to perform on such a network of droplets, the maximum geodesic distance determines the maximum number of steps in the gradual chemical reaction scheme.

Simulation Details

We consider polydisperse systems of oil droplets in water, which are modeled as hard spheres. The radii of the particles are randomly chosen from a uniform distribution in the range of $10 - 50\mu\mathrm{m}$. Initially, they are randomly placed in a cylinder of height 4 mm and radius 1 mm without overlaps among particles or between particles and walls. The particles are initialized with zero velocity. As we are interested in the effect of particle density on the agglomeration process and on the properties of the evolving networks, we perform 100 simulations each for various numbers N of particles, with $N = 10$, 20, 30, 40, 50, 60, 70, 80, 90, 100, 110, 120, 130, 140, 150, 160, 170, 180, 190, 200, 250, 300, 350, 400, 450, 500, 550, 600, 650, 700, 750, 800, 850, 900, 950, 1000, 1100, 1200, 1300, 1400, 1500, 1600, 1700, 1800, 1900, 2000, 2500, 3000, 3500, 4000, 4500, and 5000. The presented results are averages of these 100 simulation runs.

The simulation is divided in time steps of $\delta t = 10^{-5}\mathrm{s}$. In each time step, the particles are subjected to various forces: All particles are filled with oil of the density $\varrho = 1.23\mathrm{kg/l}$, so that they sink in water due to gravity reduced by the buoyant force. Furthermore, the three spatial components of the velocity vectors $\vec{v}_i$ are subjected to random velocity changes. Each velocity component $v_i^{x,y,z}$ is changed independently by a uniform random variable chosen from the interval $[-0.05|v_i^{x,y,z}|, 0.05|v_i^{x,y,z}|]$. The particles are also subjected to the Stokes friction force

$$\vec{F}_{i,\mathrm{S}} = -6\pi\eta r_i \vec{v}_i, \tag{1}$$

with the radius r_i of the particle, the current velocity $\vec{v}_i$, and the viscosity $\eta = 0.891\mathrm{mPas}$ of water at $25°\mathrm{C}$. The concept of added mass is used (Stokes, 1851). This virtual mass is the inertia added to the mass of the particle, because an acceleration or deceleration of a body in water must move or deflect some volume of surrounding fluid when it moves through it. For a spherical particle with radius r_i far away from other boundaries, the added mass is given by $\frac{2}{3}\pi r_i^3 \varrho_{\mathrm{fluid}}$, i.e., it is half of the mass of the fluid displaced by the particle.

After the new velocities of the particles are determined in this way, their positions are updated according to

$$\vec{x}_i(t) = \vec{x}_i(t - \delta t) + \vec{v}_i(t) \times \delta t. \tag{2}$$

Due to the stochastic nature of the random velocity changes, only this Euler scheme is suitable for the determination of velocities and positions of the resulting stochastic differential equation of motion (Kloeden and Platen, 2013).

After the determination of these new positions, some conditions are enforced: First, it is checked whether a particle

collides or even overlaps with a boundary of the cylinder at its new position and whether the overlap would increase if the velocity vector of the particle remains unchanged. In this case, the collision normal is determined and the velocity vector of the particle is updated according to the standard collision rules with an elasticity factor of 0.9. If there is an overlap, the position of the particle is updated in order to resolve the overlap. Analogously, then checks for collisions and overlaps between pairs of particles are performed and their velocity vectors and positions are updated accordingly. The overlaps have indeed to be eliminated as otherwise they partially remain and can even increase over time, especially in the regime of slow velocities.

In total, 10^7 time steps of duration $\delta t = 10^{-5}$s are performed during a simulation run. A simulation run thus covers in total a time span of $\mathcal{T} = 100$s which is sufficient for finishing the agglomeration process within the dimensions of the cylindrical container, as the smallest particles have a radius of at least $r_{\mathrm{smallest}} = 10\mu$m in our simulations. Shorter δt and longer $\mathcal{T}$ would be needed for smaller r_{smallest}, as δt scales with r_{smallest}^2 and $\mathcal{T}$ scales with $r_{\mathrm{smallest}}^{-2}$ due to the Stokes friction force, because of which the sink velocity is smaller for smaller spherical particles. Thus, reducing r_{smallest} by e.g. a factor of 10 would result in an increase of the number of necessary time steps and thus of the computing time by a factor of 10^4. The computing time for one simulation also depends on the number N of particles in a quadratic way, i.e., $\mathcal{T} \sim N^2$. For example, it took roughly 12 hours on a standard laptop for $N = 2000$ particles. The bottom part of Fig. 1 exhibits a final configuration of a simulation containing 2000 particles.

At the end of each 1000th time step, a configuration is recorded for the network analysis, such that we get a set of 10^4 stored configurations and thus of networks from each simulation run, which are equally spaced in time with a time delay of $\Delta t = 10^{-2}$s between each pair of successive configurations.

Network Analysis

General Remarks

As a first step in the network analysis, we need to create a network from a spatial configuration of droplets. For answering the question whether an edge exists between a pair (i, j) of droplets, we need to determine the distance $D(i, j)$ between their midpoints. Let (x_i, y_i, z_i) be the triple of midpoint coordinates of droplet i, then the Euclidean spatial distance between two droplets i and j is given by

$$D(i,j) = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2 + (z_i - z_j)^2}. \quad (3)$$

Two droplets i and j overlap if $D(i, j)$ is smaller than the sum of their respective radii r_i and r_j. Assuming that an edge between a pair of droplets exists if there is a not yet completely resolved overlap between them or if they exactly touch each other or if they get very close to each other, we can define an $N \times N$ adjacency matrix η with

$$\eta(i,j) = \begin{cases} 1 & \text{if } D(i,j) - (r_i + r_j) < 0.1\mu\mathrm{m} \\ 0 & \text{otherwise} \end{cases} . \quad (4)$$

This symmetric matrix η contains all information about the network formed by the droplets. For example, the number e of edges is given as

$$e = \sum_{i<j} \eta(i,j). \quad (5)$$

As η turns out to be extremely sparse, we create neighborhood lists $\mathcal{N}(i)$ for each node i, containing all nodes to which node i is connected by an edge:

$$\mathcal{N}(i) = \{j | \eta(i,j) = 1\} \quad (6)$$

The number of elements of $\mathcal{N}(i)$ is called the degree $k(i)$ of node i. Using these neighborhood lists, the computation time for the determination of some network properties can be significantly reduced compared to the use of the adjacency matrix alone.

We performed 100 simulations each for various numbers N of particles and recorded 10^4 configurations from each simulation. The results shown below are ensemble averages over these 100 simulation runs or values derived from these averages. Thus, each curve in the left pictures of Figs. 2, 3, 4, 5, 6, and 8 is comprised of 10^4 data points, which are ensemble averages of 100 simulation runs, so that 10^6 networks had to be evaluated for each curve.

Geodesic Distances

As mentioned above, we intend to perform a gradual reaction scheme on this network of droplets, i.e., neighboring droplets should exchange chemicals either directly at the contact face or through pores in the bilayers, the resulting intermediary reaction products serve as educts for a next step with the next droplet connected. Thus, we are interested in the question whether such reaction paths exist and how long they are, i.e., how many steps we can perform gradually. Thus, in this paper, we focus on the investigation of distances between nodes. Hereby we are not interested in the Euclidean distance $D(i, j)$ between the midpoints of the droplets, as given in Eq. (3), or the minimum distance of points within a pair of droplets, as given by $D(i, j) - (r_i + r_j)$, but in the minimum number of edges one has to cross when trying to get from node i to node j on a path comprised by the edges of the network. For example, this distance $d(i, j)$ between pairs (i, j) of nodes, which is also called geodesic distance (González and Cascone, 2014), equals 1 if $\eta(i, j) = 1$ and $d(i, j) = 2$ if $\eta(i, j) = 0$ but $\eta(i, n) = \eta(n, j) = 1$ for at least one node $n \neq i, j$. If no path from some node i to another node j can be found, one sets $d(i, j) = \infty$. As η is very sparse, we use the Dijkstra

algorithm to determine these geodesic distances (Dijkstra, 1959). Please note that the matrix d is symmetric, as we do not apply preferential directions to our edges.

Computational Results

Growth of the Network

Mean degrees Before we investigate the existence of paths and the lengths of geodesic distances, we have a look at some observables describing the time evolution of the size of the network. We start off our investigation by considering the average degree $\langle k \rangle$ of the particles, which is usually seen as one of the most important observables when performing a local network analysis. But it is also related to the overall number e of edges by the equation

$$N \times \langle k \rangle = 2e. \tag{7}$$

Figure 2 shows for some selected numbers N of droplets that the average $\langle k \rangle$ of the degrees sigmoidally increases in time. As the inset reveals, $\langle k \rangle$ indeed increases in a double-sigmoidal way. There is a first sigmoidal increase at short time scales, in which first small groups, especially pairs of droplets, are formed with some small probability. However, we are mainly interested in the subsequent second sigmoidal increase, as this main increase is the one in which the network formation takes place. This second sigmoidal increase is the more pronounced the larger the number N of droplets is. We denote the final value of the degree of node i as $k_f(i)$ and its average over all nodes as $\langle k_f \rangle$. We have a closer look at the increase of the final mean values $\langle k_f \rangle$ of the degrees with increasing N in the right picture in Fig. 2. While $\langle k_f \rangle$ can be nicely fitted linearly for very small N and quadratically for small N, we find criticalities for larger N, such that we get the overall behavior

$$\langle k_f \rangle = \begin{cases} c_1 N & \text{for } N \leq 40 \\ c_1 N + c_2 N^2 & \text{for } N \leq 80 \\ c_3 \dfrac{N^\alpha}{(N_{\mathrm{crit},1} - N)^\gamma} & \text{for } 40 \leq N \leq 900 \\ c_4 \tanh\left(c_5 \left(N - N_{\mathrm{crit},2}\right)^\beta\right) & \text{for } 850 \leq N \end{cases} \tag{8}$$

with the prefactors $c_1, \ldots, c_5$, the critical exponents α, β, and γ, and the critical numbers $N_{\mathrm{crit},1}$ and $N_{\mathrm{crit},2}$ of particles. Various fits of the functions in Eq. 8 with similar fitting qualities result in $\alpha = 1 \ldots 1.1$, $\gamma = 0.4 \ldots 0.5$, $\beta = 0.1 \ldots 0.18$, $N_{\mathrm{crit},1} = 940 \ldots 1000$, and $N_{\mathrm{crit},2} = 780 \ldots 845$. Please note that the prefactor $c_4 = 6 \ldots 7$ provides an estimate for $\langle k_f \rangle$ in the limit $N \to \infty$. While the values for prefactors and critical numbers of particles will change if altering the simulation parameters, the theory of critical phenomena decrees that the critical exponents α, β, and γ and the form of the functions stay identical.

Cluster Numbers and Cluster Sizes Next, we would like to have a look at a prominent example of global network analysis, studying cluster numbers and cluster sizes. A cluster is defined as a subset of nodes in which a path exists between any pair of nodes in this subset. Please note that we count clusters consisting of one node only also as clusters. Figure 3 shows the sigmoidal decrease of the number N_c of clusters, normalized by the number N of droplets in order to better compare the results for various N. Each curve starts at a value of 1, as each droplet forms a cluster of its own at the beginning. Generally, we find sigmoidal decreases of N_c in time. With increasing N, the increase first becomes steeper and then less steep again. The final values $N_{c,f}$ for the number of clusters, which are shown in the right picture of Fig. 3, exhibit a very interesting behavior: the data points can be well fitted to a parabolic function for small N. $N_{c,f}$ first increases with increasing N till $N = 450$ to a value of ~ 261.41 and decreases afterwards till $N = 1400$ to a value of ~ 17.56. This symmetry can be easily explained: For small N, the bottom of the cylinder is gradually filled with a two-dimensional agglomeration of droplets, forming clusters, with increasing N. This behavior is mirrored in the bottom area free of droplets, which is gradually split in a first increasing number of clusters of connected free areas, but then more and more of these free area clusters vanish. For larger values of N, the droplets need to be stacked in a three-dimensional way and the number of droplet clusters fluctuates between 12.02 and 25.68.

Another quantity which needs to be considered is the size of the largest cluster Stauffer and Aharony (1994); Stauffer (1986); Naftaly et al. (1991), in order to find out whether one large cluster is dominating the overall system and whether even the whole network is percolating. For this purpose, we measure the size $C_{\max}$ of the largest cluster, i.e., the number of nodes contained in the largest cluster. The left picture in Fig. 4 shows that $C_{\max}$ normalized by N increases sigmoidally to almost 1 for $N \geq 1000$. The final values $C_{\max,\mathrm{f}}$ are shown in the right picture of Fig. 4. We find that these final values increase almost linearly with N for $N \geq 1000$, they are only slightly smaller than N. Almost all nodes (aside from 12-27) are part of the largest cluster. The situation is different for smaller N, as can be nicely seen in the inset. Here we find that the number of particles not being part of the largest cluster first increases, then peaks at $N = 550$ with a value of 468, and afterwards decreases again. This structure is reminiscent of the parabolic shape in the right picture of Fig. 3, but here the form of the peak is less symmetric.

Existence of Paths

In order to better understand the results for geodesic distances, we first need to know how many paths in the network exist. We count the number N_p of existing paths between pairs of nodes, i.e., the number of geodesic distances which have a finite value. Please note that while there is a geometric restriction for the maximum value of a degree of

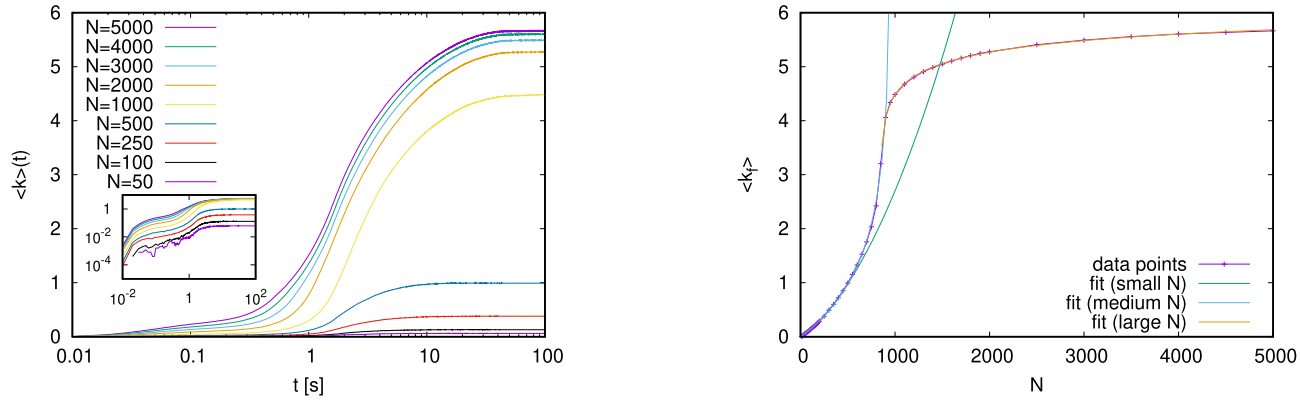


Figure 2: Left: Time evolution of the average value $\langle k \rangle$ of degrees for various numbers N of particles. In the inset, the curves are redrawn in a double-logarithmic plot. Right: Average $\langle k_f \rangle$ of final degree values vs. number N of particles: The displayed fit function for small N is given by $f(N) = 1.2 \times 10^{-3} \times N + 1.5 \times 10^{-6} \times N^2$, the fit function for intermediate N with $40 \leq N \leq 900$ by $f(N) = 2.222848282 \times 10^{-2} \times (982.95 - N)^{-0.4597383} \times N^{1.064929}$, and the fit function for large N with $850 \leq N \leq 5000$ by $f(N) = 6.4592928303905470 \times \tanh(0.41066892597996169 \times (N - 843.03747786400800)^{0.14512724870611132})$.

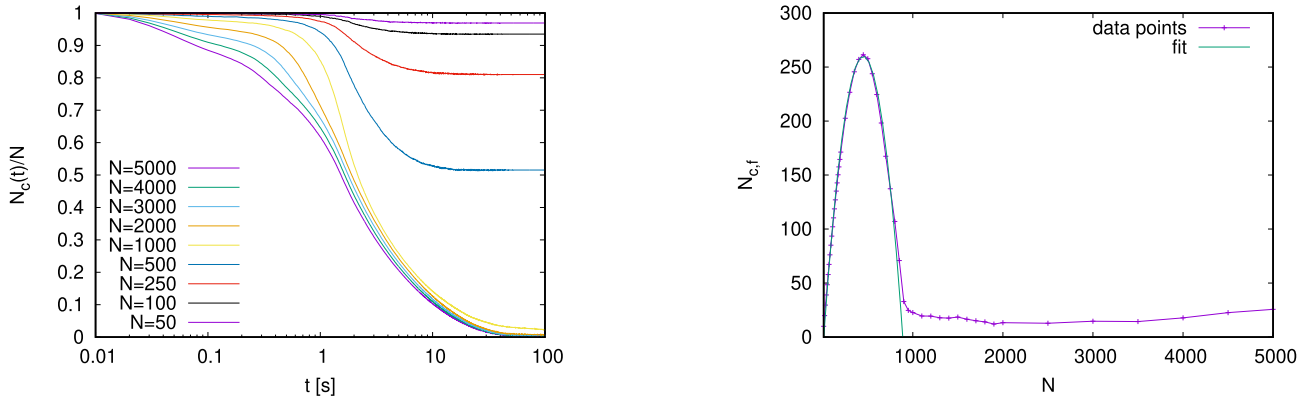


Figure 3: Left: Time evolution of the number N_c of clusters, normalized by N, for various numbers N of particles. Right: Final values $N_{c,f}$ of the number of clusters for various numbers N of particles. The parabolic shape for small N is fitted with $f(N) = 260 - 0.00135 \times (N - 450)^2$.

a node, which is also called the kissing number (Schneider et al., 2022), there is no such restriction for N_p. N_p can take values up to $N \times (N-1)/2$, for which all possible paths between pairs of nodes exist. In Fig. 5, we have a look at the fraction

$$f_p = \frac{2N_p}{N \times (N-1)} \tag{9}$$

of existing paths. This fraction is restricted to the range from $f_p = 0$, for which no edge exists in the network, to $f_p = 1$, for which the network is connected, i.e., for which a path exists from any node to every other node. The inset in the left picture of Fig. 5 with the double-logarithmic plot reveals that f_p exhibits a double-sigmoidal increase over time. We are interested in the second sigmoidal increase. The right picture in Fig. 5 shows the final values of f_p at the end of the simulation runs for various N. Here we find again a double-sigmoidal increase. Even for the largest values of N used, the fraction of existing paths does not reach a value of 1 but only of ~ 0.99. Thus, there are either singular droplets or very small groups of droplets which are not connected with the remaining system. These droplets could lie at the bottom of the cylinder without touching other droplets.

Geodesic Distances

In the next step, when intending to measure a mean geodesic distance, we have to define how exactly we intend to measure it. The main problem here is how to take those distances between pairs of nodes into account for which no path exists. We defined above that $d(i,j) = \infty$ in this case, but this defi-

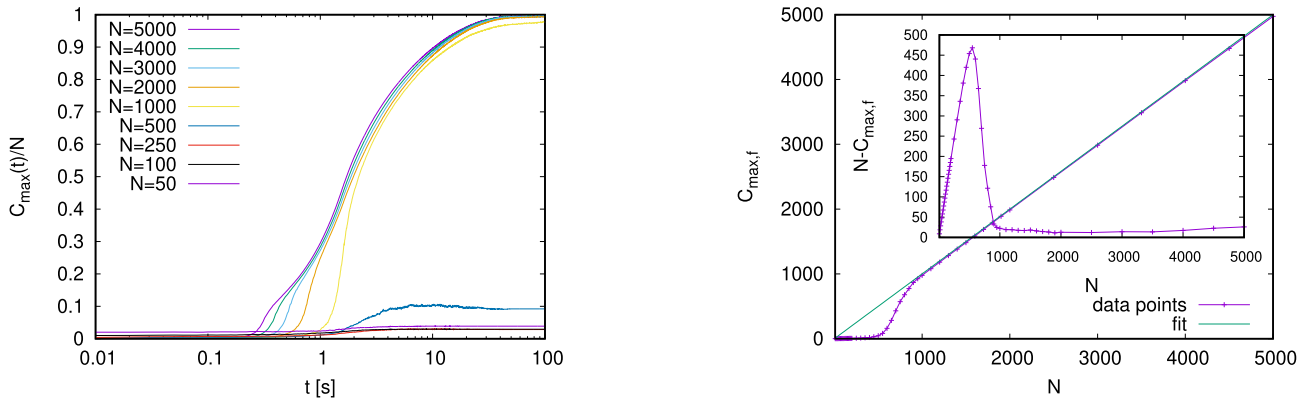


Figure 4: Left: Time evolution of the size $C_{\max}$ of the largest cluster, normalized by N, for various numbers N of particles. Right: Final values of the size $C_{\max,\mathrm{f}}$ of the largest cluster for various numbers N of particles. The fit is given by $f(N) = N$. The inset displays the difference of the numbers of particles and the final size of the largest cluster, i.e., the number of particles not included in the largest cluster.

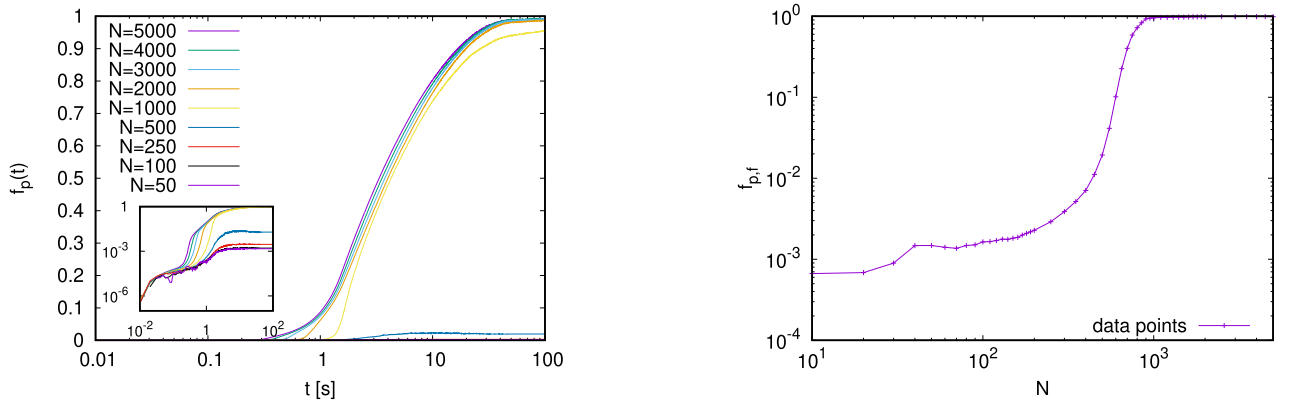


Figure 5: Left: Time evolution of the fraction f_p of existing paths between pairs of nodes for various numbers N of particles. In the inset, the curves are redrawn in a double-logarithmic plot. Right: Final values $f_{p,f}$ for the fraction of existing paths for various numbers N of particles.

nition is not suitable for taking averages. If no path between a pair (i, j) of nodes exist, some authors set $d(i, j)$ to an arbitrarily chosen value $v \geq N$, as for an existing path, a distance can only take a value in the range $1 \leq d(i, j) \leq N-1$. As the results then depend strongly on the value v, we make another choice and set

$$\langle d \rangle = \begin{cases} \frac{1}{N_p} \sum\limits_{\substack{i<j \\ \exists \mathrm{path} i \to j}} d(i,j) & \text{if } e > 0 \\ 0 & \text{otherwise} \end{cases} . \qquad (10)$$

Therefore, if there is at least one edge and thus at least one path in the network, then we take the average of the geodesic distances of the existing paths only. Otherwise, we indicate with a value of 0 that no path exists. But if there is exactly one edge and thus one path in the network, then $\langle d \rangle = 1$. At the beginning, one will thus see a transition from 0 to 1, as the probability for the existence of a first edge increases. With an increasing number of edges, also longer distances can occur, such that the mean value increases. Figure 6 shows the computational results for $\langle d \rangle$. As already mentioned above, we generally plot the ensemble average over 100 simulations. The inset nicely shows the transition whether some small network exists for small N. And also otherwise the graphic exhibits some interesting properties for $\langle d \rangle$ and its time evolution, the most important of them being the intermediary maximum occurring for larger numbers of droplets.

If we have a look at the final values $\langle d_f \rangle$ of the mean distances for various N, we find that it first increases to some intermediate maximum, decreases again and then slightly increases afterwards with increasing N, as shown in the right picture of Fig. 6. The maximum lies in a range of N, in

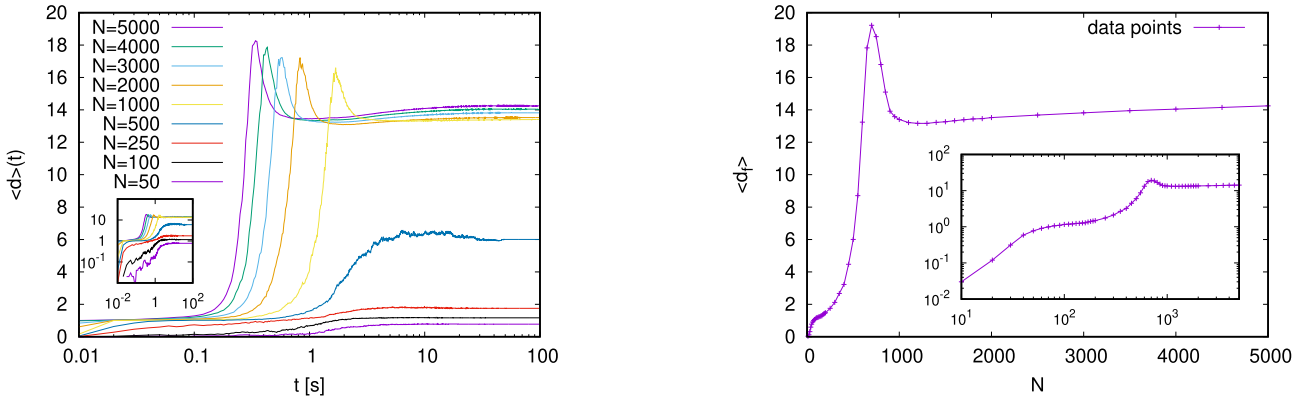


Figure 6: Left: Time evolution of the mean value $\langle d \rangle$ of the geodesic distances for various numbers N of particles. In the inset, the curves are redrawn in a double-logarithmic plot. Right: Final values $\langle d_f \rangle$ for the mean value of geodesic distances for various numbers N of particles. The inset, in which the curve is redrawn in a double-logarithmic plot, reveals a linear increase of $\langle d_f \rangle$ for small N.

which the transition from a quasi two-dimensional network at the bottom of the cylinder to a three-dimensional network starts to take place. (This maximum is missing in the left picture of Fig. 6, as no curves for $500 < N < 1000$ are shown there.) Obviously, shorter paths can at first be found by adding further droplets in the third dimension. But as the particles have to be stacked in the third dimension even further with further increasing N, the mean distance starts to increase again.

Similarly, the intermediate maxima in the curves for the largest values of N can be explained. In Fig. 7, we have a closer look at these intermediate sharp maxima which occur only for $N \geq 800$. (Wide maxima which are higher than the final values also occur at slightly smaller values of N.) The larger N, the earlier the maximum occurs at a time t_m:

$$t_m \propto \frac{1}{N} \tag{11}$$

For the largest values of N considered here, the height of the intermediate maximum increases with increasing N.

In the next step, we would like to consider the maximum distance of the network. Here again we first have to properly define how we intend to measure this maximum distance. In accordance to the definition of the mean distance in Eq. (10), we define the maximum distance as

$$d_{\max} = \begin{cases} \max\{d(i,j) | \exists \text{ path } i \to j\} & \text{if } e > 0 \\ 0 & \text{otherwise} \end{cases} . \tag{12}$$

Please note that when it comes to considering maximum distances, two types of maximum distances are often discriminated (Boitmanis et al., 2006; Erdös et al., 1989): Other authors first start off with determining a maximum distance $d_{i,\max}$ for each node defined as the maximum of distances for all paths starting at node i and ending at some other node. Then the diameter $\mathcal{D}$ of a network is defined as the maximum of these maxima,

$$\mathcal{D} = \max_i d_{i,\max}, \tag{13}$$

and the so-called radius $\mathcal{R}$ of a network as the minimum of these maxima,

$$\mathcal{R} = \min_i d_{i,\max}. \tag{14}$$

In this sense, our maximum distance $d_{\max}$ corresponds to the diameter of the network. $d_{\max}$ is of special interest for us, as we need to know the maximum number of steps possible for the gradual reaction scheme which shall be governed by the compiler we intend to develop. $d_{\max}$ corresponds to or provides a good estimate for this maximum number of steps. Contrarily, we are not interested in the radius of our networks, as we have already seen that isolated droplets or small isolated groups of droplets can lie at the bottom of the cylinder. But these isolated parts, which would define $\mathcal{R}$, play no further role in our considerations.

Figure 8 shows the time evolution of the maximum distance $d_{\max}$. At first glance, one sees the similarity to the time evolution of the mean distance $\langle d \rangle$ in Fig. 6: Again we get a double-sigmoidal increase: at first, the maximum value increases to 1, such that we now know that only pairs of droplets are formed in this stage, with the probability for pair formation increasing with time and with increasing N. The subsequent main sigmoidal increase can be analyzed as for $\langle d \rangle$. Again we have a look at the final values for various N, which are shown in the right picture of Fig. 8. The final values first sharply increase till $N = 700$, for which an ensemble average of the final maximal distances of 50.85 is found. Then it decreases again till $N = 1300$. For large N, we find a slight increase of the final values of $d_{\max}$.

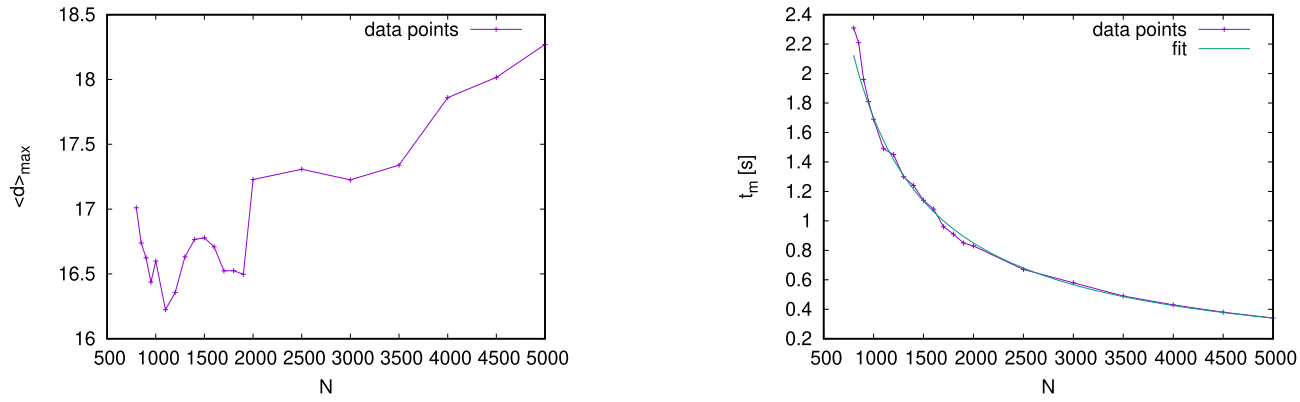


Figure 7: Left: Heights of intermediate maxima of $\langle d \rangle$ for $N \geq 800$. Right: Times at which the intermediate maxima occur for $N \geq 800$. The fit function is given by $1.7 \times 10^3 \mathrm{s} \times N^{-1}$.

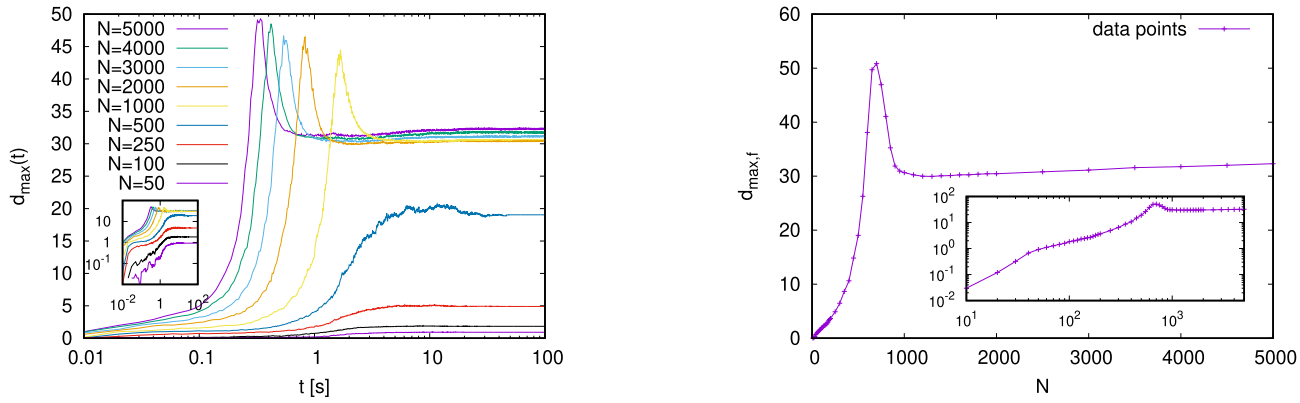


Figure 8: Left: Time evolution of the maximum $d_{\max}$ of the geodesic distances for various numbers N of particles. In the inset, the curves are redrawn in a double-logarithmic plot. Right: Final values $d_{\max,\mathrm{f}}$ for the maximum value of geodesic distances for various numbers N of particles. The inset, in which the curve is redrawn in a double-logarithmic plot, reveals a linear increase of $d_{\max,\mathrm{f}}$ for small N.

We also find intermediate maxima again for $N \geq 800$. The time $t_{\max,\mathrm{m}}$ at which $d_{\max}$ exhibits its intermediate maximum is also rather the same as the time the intermediate maxima occur for $\langle d \rangle$ and we find again the power law

$$t_{\max,\mathrm{m}} \propto \frac{1}{N}, \tag{15}$$

as shown in Fig. 9. We also find that the heights of these intermediate maxima increase with increasing N for large N. Besides these sharp maxima which we get for $N \geq 800$, we also find wide intermediate maxima with fluctuating heights for slightly smaller values of N.

Conclusion and Outlook

In this paper, we presented results of simulations for the agglomeration of droplets. As we are interested in the effects of varying numbers of particles on the agglomeration process and on the resulting droplet networks, we study a very simplified system, in which the droplets are represented as hard spheres, subjected to gravity reduced by buoyancy, as well as Stokes friction, added mass effect, random velocity changes, and almost-elastic impacts. Connections between these particles are virtually formed if they (almost) touch or overlap. The particles gradually agglomerate at the bottom of the cylindrical container. The analysis of this agglomeration process shows that the results for the time evolution and the final outcome strongly depend on the number of particles. In particular, we find two transition regimes: at small numbers N of particles, we find an over time gradually increasing number of droplets lying finally at the bottom of the cylinder where they either stay isolated or gradually form pairs and then some slightly larger groups with other parti-

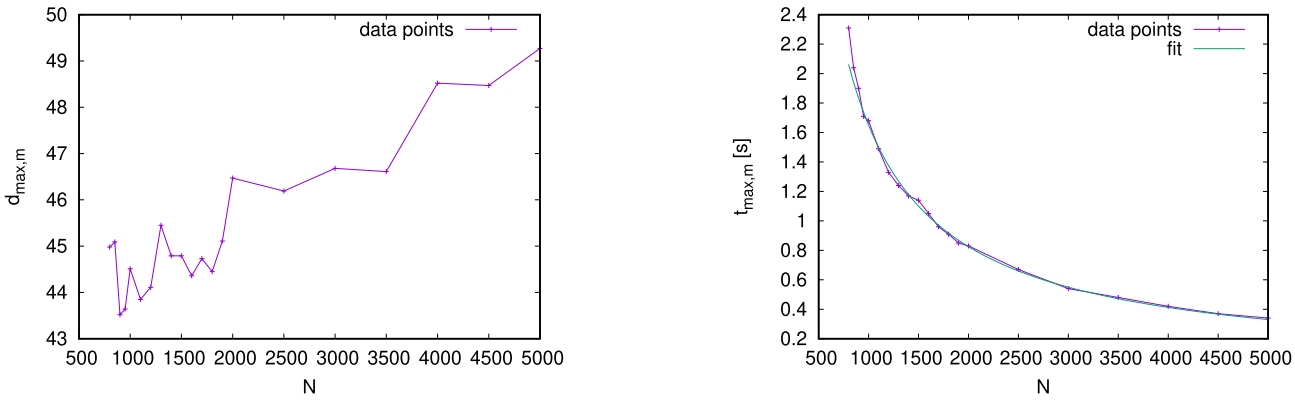


Figure 9: Left: Heights of intermediate maxima of $d_{\max}$ for $N \geq 800$. Right: Times at which the intermediate maxima occur for $N \geq 800$. The fit function is given by $1.65 \times 10^3 \mathrm{s} \times N^{-1}$.

cles reaching the bottom as well. When increasing N even further, a network of droplets is created at a bottom layer of the cylinder. During this regime, we find an increase of the mean and maximum geodesic distances, followed by a decrease for larger system sizes. For very large numbers of particles, the time evolution of the observables at first more or less reflects the final outcome for small and then intermediate numbers of particles, showing e.g. intermediate maxima for the geodesic distances (the earlier, the larger the number of particles is), before displaying the properties of truly three-dimensional networks of particles. As only large values of the maximum distance allow very extended gradual reaction schemes governed by our compiler, we suspect that it is misleading to believe that the structures optimum for our purposes need to be three-dimensional. Instead, when using a unary system of droplets, we should consider working with the largest cluster in a quasi two-dimensional layer at the bottom of the cylinder, which exhibits a fractal dimension (Falconer, 2003) significantly smaller than 2.

We intend to continue our investigations by measuring clustering coefficients, fractal dimensions, the locations of droplets with differing radii, and the importance some droplets might have for the overall network. Furthermore, we plan to extend our investigations first to binary systems, in which two particle types A and B are present and connections can only be forged between pairs of $A - B$ but not $A - A$ or $B - B$ and then to ternary systems, in which there are three particle types A, B, and C with connections between adjacent pairs of $A - B$ particles but in which the additional C-particles are unable to form any connections. Hereby we want to study the breakdown of the size of the largest cluster with increasing density of C-particles and find out whether there is a regime as well in which the maximum geodesic distance between droplets increases to suitable values. Furthermore, we want to add gluing forces between particles to find out how they change the results.

Acknowledgements

This work has been partially financially supported by the European Horizon 2020 project *ACDC – Artificial Cells with Distributed Cores to Decipher Protein Function* under project number 824060.

References

Aprin, L., Heymes, F., Laureta, P., Slangena, P., and Le Floch, S. (2015). Experimental characterization of the influence of dispersant addition on rising oil droplets in water column. *Chemical Engineering Transactions*, 43:2287–2292.

Barabási, A.-L. and Albert, R. (1999). Emergence of scaling in random networks. *Science*, 286(5439):509–512.

Boitmanis, K., Freivalds, K., Lediņš, P., and Opmanis, R. (2006). Fast and simple approximation of the diameter and radius of a graph. In Àlvarez, C. and Serna, M., editors, *Experimental Algorithms*, pages 98–108, Berlin, Heidelberg. Springer.

Bollobás, B. (2011). *Random Graphs, 2nd ed.*, volume 73 of *Cambridge Studies in Advanced Mathematics*. Cambridge University Press.

Dijkstra, E. W. (1959). A note on two problems in connexion with graphs. *Numerische Mathematik*, 1:269–271.

Donev, A., Cisse, I., Sachs, D., Variano, E. A., Stillinger, F. H., Connelly, R., Torquato, S., and Chaikin, P. M. (2004). Improving the density of jammed disordered packings using ellipsoids. *Science*, 303(5660):990–993.

Erdös, P., Pach, J., Pollack, R., and Tuza, Z. (1989). Radius, diameter, and minimum degree. *Journal of Combinatorial Theory, Series B*, 47:73–79.

Falconer, K. (2003). *Fractal Geometry: Mathematical Foundations and Applications, 2nd edition*. John Wiley & Sons, Chichester, West Sussex.

Flumini, D., Weyland, M. S., Schneider, J. J., Fellermann, H., and Füchslin, R. M. (2020). Towards programmable chemistries.

In Cicirelli, F., Guerrieri, A., Pizzuti, C., Socievole, A., Spezzano, G., and Vinci, A., editors, *Artificial Life and Evolutionary Computation, Wivace 2019, Communications in Computer and Information Science*, volume 1200, pages 145–157.

González, J. A. and Cascone, M. H. (2014). Geodesic distribution in graph theory: Kullback-Leibler-Symmetric. *Revista de Matemática: Teoría y Aplicaciones*, 21:249–260.

Hadorn, M., Boenzli, E., Sørensen, K. T., Fellermann, H., Eggenberger Hotz, P., and Hanczyc, M. M. (2012). Specific and reversible dna-directed self-assembly of oil-in-water emulsion droplets. *Proceedings of the National Academy of Sciences*, 109(50):20320–20325.

Kloeden, P. and Platen, E. (2013). *Numerical Solution of Stochastic Differential Equations*. Stochastic Modelling and Applied Probability. Springer, Berlin, Heidelberg.

Li, J. and Barrow, D. A. (2017). A new droplet-forming fluidic junction for the generation of highly compartmentalised capsules. *Lab on a chip*, 17:2873–2881.

Marino, R. and Kirkpatrick, S. (2018). Revisiting the challenges of maxclique. *CoRR*, abs/1807.09091.

Matuttis, H.-G. and Chen, J. (2014). *Understanding the Discrete Element Method: Simulation of Non-Spherical Particles for Granular and Multi-Body Systems*. John Wiley & Sons, Singapore Pte. Ltd.

Metcalfe, G., Shinbrot, T., McCarthy, J. J., and Ottino, J. M. (1995). Avalanche mixing of granular solids. *Nature*, 374:39–41.

Mitarai, N. and Nakanishi, H. (2003). Hard-sphere limit of soft-sphere model for granular materials: Stiffness dependence of steady granular flow. *Phys. Rev. E*, 67:021301.

Müller, A., Schneider, J. J., and Schömer, E. (2009). Packing a multidisperse system of hard disks in a circular environment. *Phys. Rev. E*, 79:021102.

Naftaly, U., Schwartz, M., Aharony, A., and Stauffer, D. (1991). The granular fracture model for rock fragmentation. *Journal of Physics A: Mathematical and General*, 24(19):L1175–L1184.

Ochoa, J. G. D., Binder, K., and Paul, W. (2006). Molecular dynamics simulations of the embedding of a nano-particle into a polymer film. *J. Phys.: Condens. Matter*, 18(10):2777–2787.

Pfleiderer, P. and Schilling, T. (2007). Simple monoclinic crystal phase in suspensions of hard ellipsoids. *Phys. Rev. E*, 75:020402.

Reiss, H., Ellerby, H. M., and Manzanares, J. A. (1996). Ornstein-zernike-like equations in statistical geometry: Stable and metastable systems. *The Journal of Physical Chemistry*, 100(14):5970–5981.

Ricci, A., Nielaba, P., Sengupta, S., and Binder, K. (2007). Ordering of two-dimensional crystals confined in strips of finite width. *Phys. Rev. E*, 75:011405.

Russel, W. B., Saville, D. A., and Schowalter, W. R. (1989). *Colloidal Dispersions*. Cambridge Monographs on Mechanics. Cambridge University Press.

Schneider, J. J., Barrow, D. A., Li, J., Weyland, M. S., Flumini, D., Eggenberger Hotz, P., and Füchslin, R. M. (2022). Geometric restrictions to the agglomeration of spherical particles. In *Artificial Life and Evolutionary Computation, Wivace 2021, Communications in Computer and Information Science*. accepted for publication.

Schneider, J. J. and Kirkpatrick, S. (2005). Selfish versus unselfish optimization of network creation. *Journal of Statistical Mechanics: Theory and Experiment*, 2005(08):P08007.

Schneider, J. J., Müller, A., and Schömer, E. (2009). Ultrametricity property of energy landscapes of multidisperse packing problems. *Phys. Rev. E*, 79:031122.

Schneider, J. J., Weyland, M. S., Flumini, D., and Füchslin, R. M. (2020a). Investigating three-dimensional arrangements of droplets. In Cicirelli, F., Guerrieri, A., Pizzuti, C., Socievole, A., Spezzano, G., and Vinci, A., editors, *Artificial Life and Evolutionary Computation, Wivace 2019, Communications in Computer and Information Science*, volume 1200, pages 171–184.

Schneider, J. J., Weyland, M. S., Flumini, D., Matuttis, H.-G., Morgenstern, I., and Füchslin, R. M. (2020b). Studying and simulating the three-dimensional arrangement of droplets. In Cicirelli, F., Guerrieri, A., Pizzuti, C., Socievole, A., Spezzano, G., and Vinci, A., editors, *Artificial Life and Evolutionary Computation, Wivace 2019, Communications in Computer and Information Science*, volume 1200, pages 158–170.

Stauffer, D. (1986). Percolation and cluster size distribution. In Stanley, H. E. and Ostrowsky, N., editors, *On Growth and Form: Fractal and Non-Fractal Patterns in Physics*, pages 79–100. Springer Netherlands, Dordrecht.

Stauffer, D. and Aharony, A. (1994). *Introduction to Percolation Theory, 2nd ed*. Taylor & Francis, London.

Stokes, G. G. (1851). On the effect of the internal friction of fluids on the motion of pendulums. *Transactions of the Cambridge Philosophical Society*, 9:8–106.

Weyland, M. S., Flumini, D., Schneider, J. J., and Füchslin, R. M. (2020). A compiler framework to derive microfluidic platforms for manufacturing hierarchical, compartmentalized structures that maximize yield of chemical reactions. *Artificial Life Conference Proceedings*, (32):602–604.

Zallen, R. (1998). *The Physics of Amorphous Solids*. A Wiley-Interscience publication. Wiley.

Towards Adaptive Sensorimotor Autonomy: Developing a system that can adapt to its own emergent and dynamic needs

Matthew Egbert[1,2]
[1] School of Computer Science, University of Auckland, New Zealand
[2] Te Ao Mārama, University of Auckland, New Zealand

Egbert and Barandiaran (2014) present a model of sensorimotor autonomy (Di Paolo et al., 2017), demonstrating how a pattern of sensorimotor activity can reinforce the mechanism that produces it. Subsequent investigation (Egbert, 2018) evaluated the adaptability of these autonomous sensorimotor 'habits', showing that they were robust to some perturbations, but not capable of adapting to changes to their own viability limits (Ashby, 1952) as has been demonstrated in other autonomous systems (Egbert and Pérez-Mercader, 2016).

This paper presents a new model intended to capture a more adaptive form of sensorimotor autonomy: the VIability-Sensitive Sensorimotor Autonomy (VISSA). VISSA plays the role of a 'brain' in an agent; it produces, from the sensorimotor state, a motor output and is itself transformed by its history of sensorimotor states. It interacts with a dynamic environment through a body's motors and sensors and through these interactions, self-sustaining patterns of sensorimotor behaviour emerge. This abstract provides an overview of VISSA and the first experiments that we are performing to evaluate its adaptability.

VISSA is a node-based sensorimotor-to-motor map (Woolford and Egbert, 2020) similar to the Iterant Deformable Sensorimotor Medium (IDSM) presented in (Egbert and Barandiaran, 2014; Egbert, 2018), in that it consists of a collection of 'nodes' that describe how the robot's motor activity is to change for any given sensorimotor state. Each node is a tuple: $N = \langle N_{\boldsymbol{p}}, N_a, N_{\boldsymbol{VA}}, N_{\boldsymbol{VB}} \rangle$, where $N_{\boldsymbol{p}}$ indicates the node's sensorimotor 'position,' i. e. region of sensorimotor space for which it determines the motor output; N_a is the node's 'age', a scalar value that approximates how long it has been since that node has been active; and $N_{\boldsymbol{VA}}$ and $N_{\boldsymbol{VB}}$ are the node's 'motor vectors'—ways that the node can change the system's motor output. Like in the IDSM, behaviours and the collections of nodes that generate them are 'precarious' (Di Paolo, 2009; Egbert, 2018)—when nodes are not used for an extended period of time, they cease to exist. This 'use it or lose it' dynamic means that only patterns of behaviour that maintain themselves by causing their own repetition can persist in the long term. Unlike previous architectures, VISSA includes an adaptation mechanism whereby each node adapts its influence so as to increase the likelihood of moving the sensorimotor state toward other nodes. We aim to investigate if this local 'learning' process can enable a more holistic form of adaptation whereby a cyclic collection of nodes adapts to sustain itself.

Implementation. At any given time, the 'active node,' $N^\star$, i. e. the node that is closest to the sensorimotor state, determines the change in motor output. Every iteration, the active node switches which of its motor vectors it uses to determine its output. So, if on the previous iteration $\frac{\delta m}{\delta t} = N^\star_{\boldsymbol{VA}}$, then on the current iteration $\frac{\delta m}{\delta t} = N^\star_{\boldsymbol{VB}}$, and vice versa.

After every iteration a score, S, is calculated that quantifies how well the most recently used motor vector performed at causing the sensorimotor state to approach all of VISSA's nodes,

$$S = \sum_{N^i \neq N^\star} \alpha(N^i_a)\Big(\phi(N^i_{\boldsymbol{p}}, \boldsymbol{x}_{t=t}) - \phi(N^i_{\boldsymbol{p}}, \boldsymbol{x}_{t=t-\delta t})\Big)$$

where the weight of each node, $\alpha(N_a) = \max\left(0, 1 + \frac{N_a - \mathrm{a_{max}}}{\mathrm{a_{max}} - k_{\mathrm{thresh}}}\right)$, is a truncated linear function of the node's age, that gives more weight to nodes that have been visited less recently and zero weight to nodes that have been visited very recently. The function $\phi(N_{\boldsymbol{p}}, \boldsymbol{x}) = \frac{1}{1+(20|N_{\boldsymbol{p}} - \boldsymbol{x}|)^2}$ describes a non-linear proximity of the sensorimotor state ($\boldsymbol{x}$) to the position of the node in sensorimotor space ($N_{\boldsymbol{p}}$), sampled at the current ($t = t$) and previous iteration ($t = t - \delta t$).

Scores for the two most recent uses of $N^\star_{\boldsymbol{VA}}$ and $N^\star_{\boldsymbol{VB}}$ are then used to update the active node, using a (1+1) Evolutionary Strategie (Droste et al., 2002) adjusting the motor velocities to improve at moving the sensorimotor state toward other nodes within the network, with a preference for nodes visited less recently. As the sensorimotor state changes, the active node changes and so as time passes, an adaptive self-maintaining network of nodes emerges, where each node is in a loop adapting so as to better enable the next node in the loop. At least, that is the idea! Experiments are ongoing

to evaluate how this local adaptation rule might produce an emergent collective of nodes that can also adapt in a way that prolongs its existence.

Exp. #1 A minimal demonstration of local adaptation. With a single motor variable and no sensory variables, the sensorimotor state is just a motor state, represented by a single scalar value, $m \in \mathbb{R}$. Three nodes are placed evenly in sensorimotor space and assumed to fit within a larger collection, where movement from N^0 to N^1 to N^2 eventually results in a return (via other nodes $N^3 ... N^n$, which are not simulated, back to N^0. Given this assumption, motor activity that causes motion in the positive direction (i. e. from N^0 to N^1 to N^2) is considered 'adapted' as it causes the nodes in the network to be regularly revisited and thus is good for the overall persistence of the collection. In my talk, I will describe the simulations that show that the middle node, N^1 is indeed capable of adapting its velocity vectors in a way that that would improve the overall persistence of the collection and from a variety of initial conditions.

Exp. #2: An adaptive autonomous collective? I simulate a robot situated in a 1D periodic environment. Its position $r \in (0, 1]$ changes as a function of its motor state ($\frac{dr}{dt} = m$) and determines the state of its sensor, $s = \exp(-40|r - 0.75|^2)$, where $|r - 0.75|$ is the distance between the robot and the peak stimulus location. The robot is controlled by VISSA which now operates in a 2D sensorimotor space. To initialize the nodes I simulate a brief training phase, where the robot is initially placed at $r = 0.6$ and its motor is externally controlled as a function of time $m = \frac{3\sin(2t)}{4}$. This causes the robot to move back and forth close to the stimulus—a cycle in sensorimotor space involving positive and negative motor states with high and low stimulus levels. Every 10th iteration (the time step, $\Delta t = 0.01$) a node is added with its position, $N_{\boldsymbol{p}}$ set to the current sensorimotor state of the robot; its $N_{\boldsymbol{VA}}$ set to the current rate of motor change; its $N_{\boldsymbol{VB}}$ set to a 'wrong value' (i. e. a negative fraction of $N_{\boldsymbol{VA}}$, $N_{\boldsymbol{VB}} = -0.1N_{\boldsymbol{VA}}$); and $N_a = 0$.

After training ends, the nodes adapt to correct their wrong values to those that reproduce a cycle of behaviour. The behaviour that emerges appears to be robust (Figs. 1 & 2): with a sensorimotor loop similar to that driven during training repeated many times, but with variations. Some of the variations are minor, but four times during the trial, there is a significant divergence from portions of the main loop. I will discuss these results and the additional experiments that are needed to probe the limits of the robustness and adaptability of the VISSA-based behaviour.

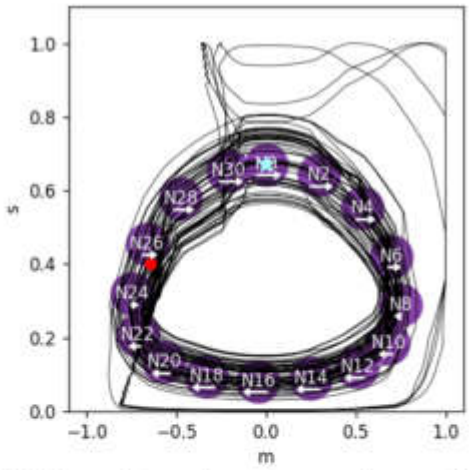


Figure 1: Node positions and sensorimotor trajectory of the robot in Exp. 2. Arrows indicate the $N_{\boldsymbol{VA}}$ values for each node at end of training. The blue star shows the sensorimotor state at the end of the training phase, and the red circle shows the sensorimotor state at the end of the trial.

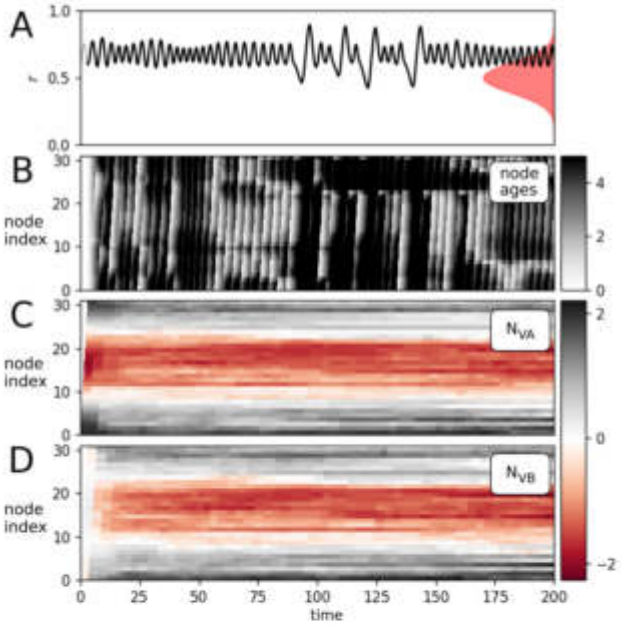


Figure 2: Exp. #2, showing (A) the position of the robot, with the the training phase indicated in a thinner line, and the stimulus associated with each location in space shown in red; (B) node ages; (C & D) node motor vectors.

References

Ashby, W. R. (1952). *Design for a Brain: The Origin of Adaptative Behaviour*. J. Wiley, London, second edition.

Di Paolo, E. A. (2009). Extended Life. *Topoi*, 28(1):9–21.

Di Paolo, E. A., Buhrmann, T., and Barandiaran, X. (2017). *Sensorimotor Life: An Enactive Proposal*. Oxford University Press, 1st edition edition.

Droste, S., Jansen, T., and Wegener, I. (2002). On the analysis of the (1+1) evolutionary algorithm. *Theoretical Computer Science*, 276(1):51–81.

Egbert, M. D. (2018). Investigations of an Adaptive and

Autonomous Sensorimotor Individual. *The 2018 Conference on Artificial Life: A Hybrid of the European Conference on Artificial Life (ECAL) and the International Conference on the Synthesis and Simulation of Living Systems (ALIFE)*, pages 343–350.

Egbert, M. D. and Barandiaran, X. E. (2014). Modeling habits as self-sustaining patterns of sensorimotor behavior. *Frontiers in Human Neuroscience*, 8.

Egbert, M. D. and Pérez-Mercader, J. (2016). Adapting to Adaptations: Behavioural Strategies that are Robust to Mutations and Other Organisational-Transformations. *Scientific Reports*, 6:18963.

Woolford, F. M. and Egbert, M. D. (2020). Behavioural variety of a node-based sensorimotor-to-motor map. *Adaptive Behavior*, 28(6):425–440.

Simulations of Vesicular Disentanglement

Peter E. Hotz[1], Johannes J. Schneider[1], Federica Casiraghi[2], Silvia Holler[2],
Martin M. Hanczyc[2], and Rudolf M. Füchslin[1],
[1]School of Engineering, Zurich University of Applied Sciences, Technikumstrasse 9, 8401 Winterthur, Switzerland
[2] Laboratory for Artificial Biology, Centre for Integrative Biology (CIBIO), University of Trento, 38123, Trento, Italy
eggg@zhaw.ch

Abstract

As part of the European Horizon 2020 project ACDC, a chemical compiler is being developed that allows the self-assembly of artificial, three-dimensional, vesicular structures to be first simulated and then translated into reality. This work reports on simulations that shed light on an important aspect: How to disentangle inter-vesicular connections?

Introduction

The scenario to be discussed in this extended abstract is illustrated in Fig.1. Two vesicles are joined by a sticker mechanism based on specific DNA-DNA interactions (Hadorn and Eggenberger Hotz (2010); Hadorn et al. (2012); Schneider et al. (2021)). The goal is to introduce a mechanism that makes this process reversible. The motivation for doing so is to correct possible errors, separate parts from a larger assembly, or create empty spaces in larger clusters Weyland et al. (2020).

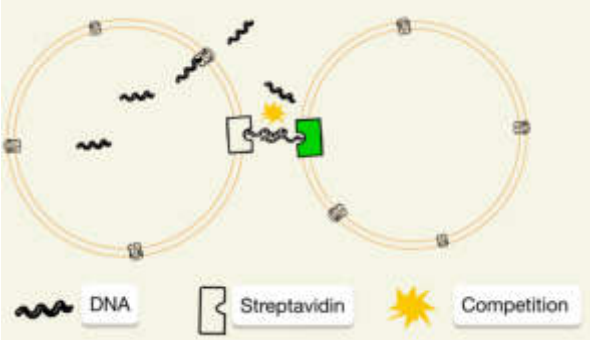


Figure 1: Two vesicles linked with streptavidin-DNA-linkers together. The incorporated pores allow the passage of DNA-strands designed to compete with the already linked ones and liberate the vesicle from its partners.

Therefore, mechanisms have been devised in which DNA tags can leave a vesicle and compete with the pre-existing DNA strands connecting the two vesicles. If the interaction of the exiting DNA is higher than that of the bound DNA, competition occurs which, if strong enough, can free the vesicle from its partners.

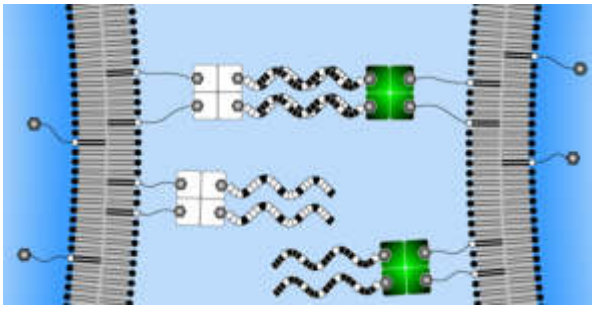


Figure 2: The protein Streptavidin (white and green squares) is anchored to the vesicular membrane with biotinylated poly-ethylen-glycol (PEG), which is linked to phospholipids (amphiphilic molecules building up the membrane). The DNA-strands (wavy lines) are linked to the Streptavidin and can bind to each other by specific DNA-DNA interactions.

The details of the linking mechanism are explained in Fig. 2. Chemicals that interfere with the specific linkers have been developed and the incorporation of them into the vesicles is done by the Pautot-Weitz-Method(Pautot et al. (2003)). The vesicles are equipped with pores that allow competing molecules to diffuse into the environment and compete with the established linkers. Using nucleic acid snippets (DNA, RNA), it is possible to shape the interaction energy between the different vesicles and ensure that the competing free snippet also breaks the existing links.

The following steps were performed to produce a sticker that can be used to selectively bond two vesicles together: Polyethylene glycol (PEG) is a non-toxic polymer with the general molecular formula $C_{2n}H_{4n} + 2O_{n+1}$. PEGs can be anchored in vesicular membranes and have a biotin at their end. The protein streptavidin has a high affinity for

biotin and is used to bind it to biotinylated PEGs, which are anchored in the vesicular membrane. The DNA strands (wavy lines) are linked to the streptavidin molecule. The use of PEG to link the vesicles has the advantage of reducing non-specific binding of the vesicles due to entropic effects. If two vesicles are now equipped with DNA strands that can specifically link to each other via hydrogen bonds, they can attach to each other. In the process, the vesicles deform due to the adhesion forces and a flat adhesion zone is formed. Unbound stickers can diffuse freely on the surface of the vesicles.

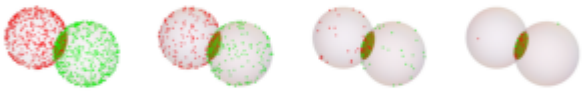


Figure 3: The linkers can diffuse freely in the vesicular membrane until they find a partner of the other vesicle to link and get trapped in the adhesion zone between the two vesicles.

A simulation was developed for the competition of the snippets with already existing links. The conservation of mass and the reactions lead to the following set of equations:

$$\begin{aligned} \frac{OL[t]}{dt} &= k1 * FL[t] * C2[t] - k2 * OL[t] \\ \frac{NL[t]}{dt} &= k3 * FL[t] * C1[t] - k4 * NL[t] \\ FL[t] &= 1.0 - NL[t] - OL[t] \end{aligned}$$

OL is the concentration of DNA-DNA linkers between the bound vesicles. As explained in Fig. 3, the linkers are free to diffuse on the surface of their vesicles until they find a partner on the other vesicles and become trapped between the vesicles. As a result, a linker zone is formed between the vesicles. We assume that all available stickers are in this zone. NL are the linkers where the old linker OL is replaced by the new one. NL is the concentration of successfully replaced linkers by the competing DNA snippet from inside the vesicle. Since the DNA-DNA interactions can also break down, there is also a concentration of free linkers, which we denote by FL. The number of all streptatividin molecules on the two vesicles remains constant and is the sum of $FL + NL + OL = const$. We set this constant to 1.0. [1]

Results and Discussion

The results in Fig. 4 are intuitively clear. The binding energy of the DNA-DNA interaction must be higher for the competing DNA strand than for the old interactions. What may not be so obvious is that the rate of resolution of the DNA-DNA interactions is also important (see Fig. 5): the higher the interaction energy of the old linkers, the longer one has to wait until equilibrium is reached. To obtain a working experiment, one should design the system with a rather weak interaction energy between the DNA-DNAs, just enough to keep the vesicles together and ensure a short waiting time until equilibrium is reached. This reflects an important design mechanism in nature: developmental processes often rely on a large number of weak interactions instead of a few strong ones, which has the advantage that the processes can be easily adjusted because changing a single interaction requires little energy.

[1] All the simulations were performed with Mathematica®, a product of Wolfram Inc.

Acknowledgements

This work has been partially financially supported by the European Horizon 2020 project ACDC under project number 824060.

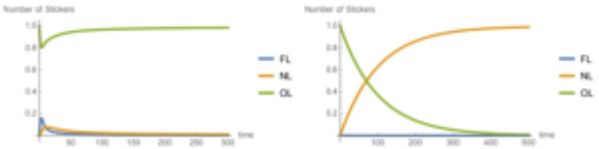


Figure 4: **Left**: The competition is weak (k1 smaller than k3) and the old linker concentration does change only slightly. NL = 0.0106716 OL = 0.987037 FL = 0.00229118 (equilibrium values)). **Right**: The competition is strong (k1 bigger than k3) and the old linkers is displaced by the new one.

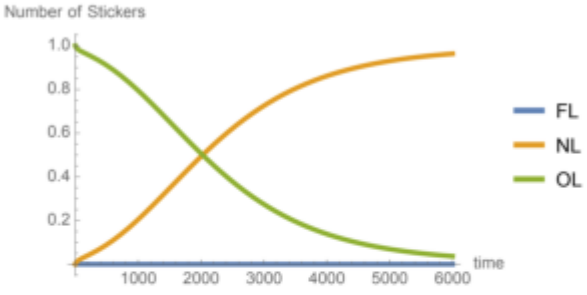


Figure 5: If the binding of the weaker linker is very high, the experimentalist has to wait a long time until equilibrium is reached. k1 = 0.5 k2 = 0.0001 k3 = 0.1 k4 = 0.001 NL = 0.963488 OL = 0.0365123 FL = 1.27385 $\times$ $10^{-}7$ (equilibrium values)

References

Hadorn, M., Boenzli, E., Sørensen, K. T., Fellermann, H., Eggenberger Hotz, P., and Hanczyc, M. M. (2012). Specific and reversible dna-directed self-assembly of oil-in-water emulsion droplets. *Proceedings of the National Academy of Sciences*, 109(50):20320–20325.

Hadorn, M. and Eggenberger Hotz, P. (2010). Dna-mediated self-assembly of artificial vesicles. *PLOS ONE*, 5(3):1–9.

Pautot, S., Frisken, B., and Weitz, D. A. (2003). Production of unilamellar vesicles using an inverted emulsion. *Langmuir*, 19:2870–2879.

Schneider, J. J., Faggian, A., Holler, S., Casiraghi, F., Li, J., Cebolla, L. S., Matuttis, H.-G., Hanczyc, M. M., Barrow, D. A., Weyland, M. S., Flumini, D., Hotz, P. E., and Füchslin, R. M. (2021). Influence of the geometry on the agglomeration of a polydisperse binary system of spherical particles. In *Alife 2021 : The 2021 Conference on Artificial Life, online, 19-23 July 2021*. MIT Press.

Weyland, M., Flumini, D., Schneider, J. J., and Füchslin, R. M. (2020). A compiler framework to derive microfluidic platforms for manufacturing hierarchical, compartmentalized structures that maximize yield of chemical reactions. In *Artificial Life Conference Proceedings*, pages 602–60.

Physical Obstacles Constrain Behavioral Parameter Space of Successful Localization in Honey Bee Swarms

Dieu My T. Nguyen[1,2], Michael L. Iuzzolino[1], Orit Peleg[1,2,3]
[1] Department of Computer Science, University of Colorado Boulder, Boulder, Colorado, USA
[2] BioFrontiers Institute, University of Colorado Boulder, Boulder, Colorado, USA
[3] Santa Fe Institute, Santa Fe, New Mexico, USA
Corresponding Author: Orit.Peleg@colorado.edu

Abstract

Honey bees (*Apis mellifera* L.) localize the queen and aggregate into a swarm by forming a collective scenting network to directionally propagate volatile pheromone signals. Previous experiments show the robustness of this communication strategy in the presence of physical obstacles that partially block pheromone flow and the path to the queen. Specifically, there is a delay in the formation of the scenting network and aggregation compared to a simple environment without perturbations. To better understand the effect of obstacles beyond temporal dynamics, we use the experimental results as inspiration to explore how the behavioral parameter space of collective scenting responds to obstacle. We extend an agent-based model previously developed for a simple environment to account for the presence of physical obstacles. We study how individual agents with simple behavioral rules for scenting and following concentration gradients can give rise to collective localization and swarming. We show that the bees are capable of navigating the more complex environment with a physical obstacle to localize the queen and aggregate around her, but their range of behavioral parameters is more limited and less flexible as a result of the spatial density heterogeneity in the bees imposed by the obstacle.

Introduction

Social insect groups often navigate complex and unknown environments. To do so, group members must effectively communicate. Insects, such as honey bees, often exchange information and coordinate group processes by communicating via pheromones, volatile chemical signals that decay rapidly in time and space (Conte and Hefetz, 2008; Lensky and Cassier, 1995). In the context of honey bee swarm formation around the queen, worker bees localize the queen by following her pheromones and propagate the signals about her location by "scenting" (McIndoo, 1914; Peters et al., 2017; Nguyen et al., 2021b). The scenting behavior consists of a given bee sensing local pheromone concentration above a given threshold and releasing pheromones from the Nasonov gland while rigorously fanning its wings to disperse the signals to other bees. Wing fanning creates a directional bias in the flow of pheromones, allowing bees farther away from the queen to sense the signal and further propagate them. This collective scenting strategy creates an effective communication network for localization and aggregation in honey bee swarms (Nguyen et al., 2021b).

Nguyen et al. (2021a) experimentally showed the robustness of collective scenting in the presence of obstacles that partially block pheromone flow and the open path to the queen. Compared to a simple environment without any obstacle, the more complex environment requires more time to explore and navigate. However, the bees still effectively employ the scenting strategy to overcome the obstacle and aggregate around the queen (Fig. 1B).

Inspired by the experiments, we turned to computational modelling to further explore how physical obstacles affect the parameter space that dictates the dynamics of localization and aggregation based on collective scenting. A previous work modeling social amoeba aggregation by chemical signaling has shown the system's robustness to physical obstacles, with agents releasing isotropic chemicals that diffuse axi-symmetrically (Fatès, 2010). To model honey bee chemical signaling, a previous study (Nguyen et al., 2021b) used agent-based modeling to study how simple behavioral rules can generate complex collective behaviors and took into consideration the directional bias of signals that provide directional information to group members. The authors studied the effect of two behavioral parameters that the bees may vary with input from the environment—the directional bias representing the magnitude of wing-fanning and the concentration threshold above which they can detect the signal. The model showed the importance of directional signals seen in the scenting strategy in efficient localization and aggregation around the queen that can avoid less desired outcomes, such as small clusters far from the queen. In this study, we build upon this model by adding physical obstructions to the system (Fig. 1A,C). Per the experiments in Nguyen et al. (2021a), we expect to observe the robustness of the bee communication system in the more complex environment. More importantly, by modeling, we aim to gain insights into how the behavioral parameter space responds to the presence of obstacle. The insights may contribute to designs and improvements in non-biological systems with individuals that are limited to local interactions but must co-

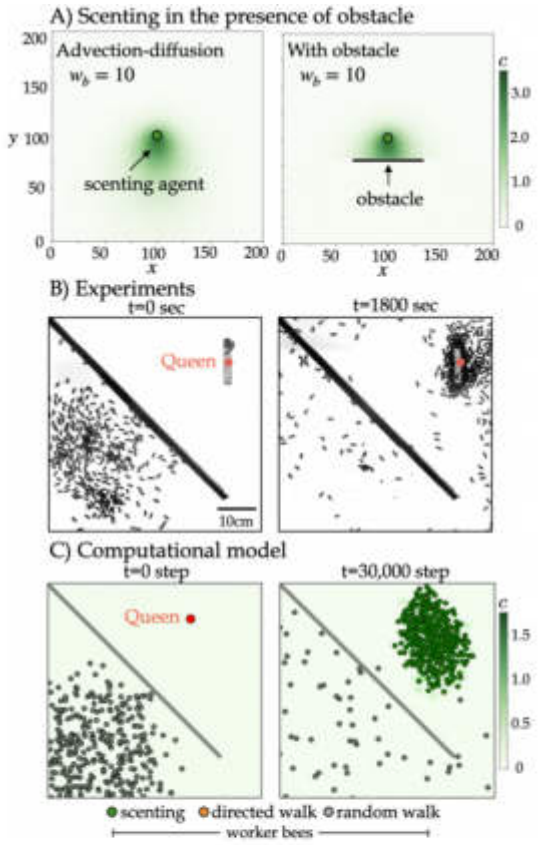


Figure 1: Experiments and model of collective scenting in the presence of a physical obstacle. A) A scenting bee agent can produce a pheromone signal with directional bias with advection-diffusion ($w_b = 10$). A physical obstacle can partially block the signal. B) An experiment where worker bees use scenting to navigate the obstacle and localize the queen. C) A computational model simulating pheromone diffusion and honey bee scenting behavior in presence of obstacle.

ordinate group processes in complex environments, such as robots that must navigate obstructions (Abiyev et al., 2010).

Methods

Experimental setup & analysis

We followed methods described in detail in Nguyen et al. (2021b) for experiments without obstacle and in Nguyen et al. (2021a) with obstacle. As this paper mainly focuses on modeling, we briefly summarize the methods. The backlit arena (50x50x1.5 cm) is semi-2-D to prevent flying, as bees have been shown to scent while standing (McIndoo, 1914). The experiments are recorded aerially with a video camera (4k resolution, 30 fps). The queen is kept in a cage (10.5x2.2x2.2 cm) at the top right corner (Fig. 1B). A wooden bar is placed diagonally for the obstacle condition. Workers are placed at the bottom left corner, and a plexiglass sheet encloses the arena. For each environmental condition, we report results for three experiments with similar number of worker bees ranging from 240 to 380. Five experiments per condition were presented in Nguyen et al. (2021a); we present three per condition here for consistency in number of bees in the simulation ($N = 300$).

To automatically detect scenting bees and their orientations, we use computer vision approaches presented in Nguyen et al. (2021b) (Appendix Fig. A1). We detect individual bees (i.e., x, y centroids) by Otsu's method of adaptive thresholding, and morphological transformations (Otsu, 1979; Dougherty, 1992). To classify a bee as scenting or non-scenting, we train a ResNet-18 convolutional neural network (CNN) model (He et al., 2016) that achieves 95.17% test accuracy. We create a regression model for orientation estimation, which achieves 96.71% test accuracy.

We then reconstruct attractive surfaces to correlate the scenting events with the spatiotemporal density of bees. For each scenting bee i at time t, we define its position as $\mathbf{s}^p_{i,t}$, and its direction of scenting as $\mathbf{s}^d_{i,t}$ (unit vector). Assuming the scenting bees provide directional information to non-scenting bees, we treat $\mathbf{s}^p_{i,t}$ and $\mathbf{s}^d_{i,t}$ as a set of gradients that define a minimal surface of height $f(x, y, t)$. Thus, $f(x, y, t)$ corresponds to the probability that a randomly moving non-scenting bee will end up at position (x, y) by following the scenting directions of scenting bees: $f(x, y) = \sum_{\forall \nabla f} \int \nabla f dx dy$ where $\nabla f = \mathbf{s}^p_{i,t} + \mathbf{s}^d_{i,t}$. We regularize the least squares solution of surface reconstruction from its gradient field, using Tikhonov regularization (Harker and O'Leary, 2008, 2011).

Finally, we obtain some time-series properties. The number of scenting bees over time is presented as a rolling mean with the window size of 100 frames. The average distance to the queen is computed as the average distance of all black pixels to the queen's location, as the bee detection method cannot detect every single individual bee when they touch or overlap. The queen's cage and the obstacle are stationary, thus the remaining black pixels in the arena represent only the moving bees and allow us to use this proxy. For each property, we average the time-series data across all experiments for each condition and obtain the standard deviation.

Modeling pheromone diffusion

We model pheromone advection-diffusion using the 2-D diffusion partial differential equation to describe pheromone concentration, $C(x, y, t)$, at a position and time:

$$\frac{\partial C(x,y,t)}{\partial t} = D\frac{\partial^2 C(x,t)}{\partial x^2} + D\frac{\partial^2 C(y,t)}{\partial y^2} - w_b w_x \frac{\partial C(x,t)}{\partial x} w_b w_y \frac{\partial C(y,t)}{\partial y} - \gamma C(x,y,t) \quad (1)$$

where $C(x, y, t)$ is the concentration at position $[x, y]$ at time t, w_x and w_y are the x and y components of emission vector respectively, D is the diffusion coefficient, and γ is the decay constant. The behavioral parameter representing the direc-

tional bias, w_b, is the magnitude of the advection–diffusion of pheromone released by a bee (Fig. 1A). Treating a single scenting bee as a point source of localized and instantaneous pheromone emission, we solve Eq. 1:

$$C(x,y,t) = \frac{C_0}{\sqrt{t}} exp\left(-\frac{A^2+B^2}{4Dt} - \gamma t\right) \tag{2}$$

where C_0 is the initial concentration, $A = (x - w_b w_x t)$, and $B = (y - w_b w_y t)$. The environmental parameters of the model include: the size of the 2–D arena (X_{min} and X_{max}) and the size of a grid cell (δ_X), the start and final time of the simulation (t_i and t_f) and the time integration constant (δ_t).

Similar to the experiments, we model the physical obstacle as a diagonal linear bar of pixels with a small opening. Pheromones cannot diffuse past the obstacle by a line-of-sight method. The obstacle forms a line segment $\overline{CD}$, and segment $\overline{EF}$ forms between the scenting agent and a given pixel. Intersection of the lines indicates that the pheromone from the scenting agent does not reach the pixel. To find the point of intersection, we solve the matrix equation: $\begin{bmatrix} A_{CD} & B_{CD} \\ A_{EF} & B_{EF} \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} C_{CD} \\ C_{EF} \end{bmatrix}$ where A_{CD} is the slope of $\overline{CD}$, A_{EF} is the slope of $\overline{EF}$, $B_{CD} = -1$, $B_{EF} = -1$, C_{CD} is the negative y-intercept of $\overline{CD}$, and C_{EF} is the negative y-intercept of $\overline{ED}$. If a solution exists, we check if the intersection point lies on both lines. If there is no solution or the solution does not lie on both lines, there is no intersection and the pheromone from the source at E is present in the pixel at F.

Modeling behavioral rules

In a discrete 2-D arena (Appendix Fig. A2A), the queen is stationary and frequently releases pheromone isotropically, i.e., without directional bias ($wb = 0$). She is the global point of convergence for the swarm. The behavioral rules of workers are: (1) A worker bee performs a random walk. Based on her distance to the queen, the bee detects the queen's pheromone if above the threshold (T). (2) If T is met, the bee orients towards the direction up the gradient. The negative vector of the gradient scaled by w_b is the direction to emit pheromone and disperse it via wing-fanning. The bee then either walks up the gradient (Appendix Fig. A2B) or stands still for a certain time to emit and fan her own pheromones, each event with a 0.5 probability. (3) Bees that detect this cascade of secondary signals will follow the same rules to head towards maximum pheromone concentration or scent and further propagate the information.

We formalize the worker bees' behavior as a probabilistic state machine (PSM) (Rabin, 1963). The PSM consists of a set of finite states that describe bee behavior and a probabilistic transition matrix for how a bee may change from one state to another. Specifically, the state model $SM_{worker} = (S, s_0, I, M)$, associated with each worker, defines her set of behavioral rules within the environment,

s_c \ s_n	**rWalk**	**tMet**	**emit**	**fan**	**dWalk**
rWalk	$c_i < T$	$c_i \geq T$	0	0	0
dWalk	$c_i < T$	$c_i \geq T$	0	0	0
tMet	0	0	0.5	0	0.5
emit	0	0	0	1	0
fan	$t_i \geq P_w \wedge c_i < T$	$t_i \geq P_w \wedge c_i \geq T$	0	$t_i < P_w$	0

Table 1: Probabilistic state machine transition matrix for honey bee behavioral rules. Variables $randomWalk$, $thresholdMet$, and $directedWalk$ abbreviated as $rWalk$, $tMet$, and $dWalk$, respectively.

and SM_{worker} components are fixed across all worker bees: $S = \{randomWalk, directedWalk, thresholdMet, emit, fan\}$ is a set of finite states, where the variable $randomWalk$ is a random walk when the threshold is not met, $directedWalk$ is the walk up the concentration gradient, $thresholdMet$ is when the threshold is met, $emit$ is the instantaneous release of pheromone, and fan is the wing fanning at a constant position. $s_0 = randomWalk$ is the initial state of each bee. $I = \{t_i, c_i\}$, is a set of flags for the input conditions on state transitions, where for a given bee, t_i is a counter for the time that bee is in the fan state and c_i is the concentration at that bee's position.

For the transition matrix M, there are two relevant parameters, P_w and T, representing the emission period made of the $emit$ and the fan state and the threshold over which a bee can be activated from state $randomWalk$. Table 1 provides the conditions and probabilities for transitioning from the current state, s_c, to the next state, s_n.

We compute the gradient of pheromone concentration for a given bee to find the direction of greatest local change:

$$\begin{aligned} \nabla_{(x,y)} C =& K_i E_i (x - x_i - w_b w_x t)\hat{x} \\ &+ K_i E_i (y - y_i - w_b w_y t)\hat{y} \end{aligned} \tag{3}$$

where $K_i = -A/2Dt\sqrt{t}$ and $E_i = exp(-(x - x_i - w_b w_x t)^2 + (y - y_i - w_b w_y t)^2/4Dt - \gamma t)$, x, y are the position of the activated bees, and x_i, y_i are the position of the scenting bees (i.e., pheromone source) i. The cumulative gradient for the concentration at a single bee's position is the sum of the normalized gradients resulting from each pheromone source or emitting bee i:

$$\begin{aligned} \nabla_{(x,y)} C_{cumulative} &= \sum_i \nabla_{(x,y)} C_i(x,y) \\ &= \sum_i K_i E_i X \hat{x} + \sum_i K_i E_i Y \hat{y} \end{aligned} \tag{4}$$

where $X = x - x_i - w_b w_x t$ and $Y = y - y_i - w_b w_y t$. This gradient defines the vector that points in the direction of the bee's heading for its directed walk. The negative vector of the gradient is the direction for this bee's pheromone emission for signal propagation, and thus its x and y components make up the w_x and w_y terms of Eq. 2.

Each discrete pixel or cell contains only one bee at a time. Pixels that make up the physical obstacle do not contain any

bees. Upon its next movement, a bee agent checks if the intended pixel is already occupied by another bee or the obstacle; if so, the bee chooses another nearby pixel within 45° over five iterations until it stays at the current position. Based on the model parameterization in Nguyen et al. (2021b), we explore a range of values for the behavioral parameters, the directional bias w_b and concentration threshold T, that bees could adjust based on input from the environment. For each combination of the parameters, we repeat the simulation three times per condition. Other parameters remain constant across all simulations (Appendix Table 2). The algorithm is presented in Appendix Algo. 1.

Construction of phase diagrams

We extract several properties from the simulation data for time-series analyses: the bees' average distance to the queen, the average number of scenting bees, and the average distance to the queen from the farthest active bee. To characterize the growth of the queen's cluster size and the number of clusters that form, we use the density-based spatial clustering of applications with noise (DBSCAN) algorithm (ε: 0.25, minimum number of bees to form a cluster: 5) to cluster bees at every time step (Ester et al., 1996).

To characterize the collective scenting behavior determined by the behavioral parameters (w_b, T), we define four possible phases (i.e., the outcome of simulations, rather than a period in a time sequence) as previously seen in Nguyen et al. (2021b): phase 1 of small clusters of bees spread throughout the arena, phase 2 of bees reaching the queen's vicinity by random walk, phase 3 of bees swarming around the queen by forming a percolation network of senders and receivers of pheromone signals created by scenting bees as seen in the experiments, and phase 4 of no clustering at all. To construct the phase diagrams, we use three properties to distinguish the phases: the final number of clusters, the final queen's cluster size, and the distance of the farthest active bee to the queen. We sequentially applied the following conditions to each simulation identify its phase group: 1) If the final number of clusters > 1.5: Phase 1 with many small clusters; 2) If the final number of clusters $0 - 1.5$ and the final queen's cluster size < 250 bees: Phase 4 with no clustering; 3) If the farthest active distance < 4.0: Phase 2 with clustering at the queen's location via random walk ; 4) If the farthest active distance ≥ 4.0: Phase 3 with clustering at the queen's location via a scenting percolation network.

Results

Bees navigate obstacles by collective scenting

As previously presented in Nguyen et al. (2021a), the experiments comparing the localization and aggregation dynamics in the presence and absence of physical obstacles show that bees are able to solve the problem in both conditions by employing the collective scenting strategy. We show snapshots of the experiment and the corresponding attractive surfaces

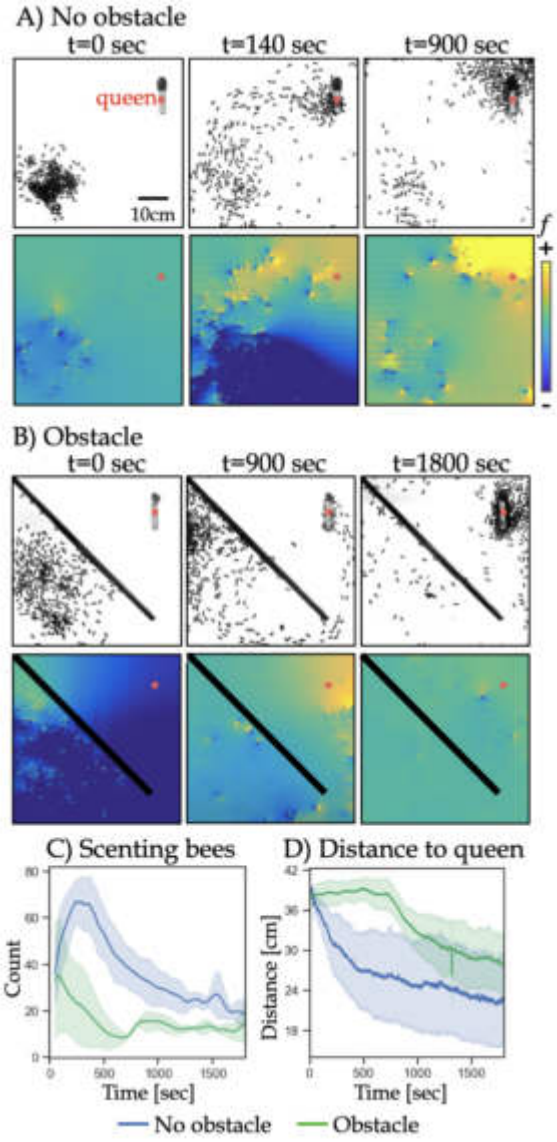


Figure 2: Experiments. A) Snapshots of an experiment where worker bees are in a semi-2D arena with a caged queen without obstacles. The corresponding attractive surfaces f show the scenting events correlating to the spatial-temporal density of bees. B) Snapshots and surfaces of an experiment where bees are initially placed on on side of a bar obstacle and the queen is on the other side. C) The average number of scenting bees over time for both conditions. D) The average distance to the queen over time.

for an example experiment without obstacles ($N = 320$) in Fig. 2A. Over 1800 seconds or 30 minutes, the bees activate a scenting network early (around $t = 140$ sec), as reflected in the surface in which the scenting directions collectively point to the queen's area (i.e., surface regions of higher f values). Most of the bees have formed a swarm around the queen around $t = 900$ sec or 15 min. In the presence of

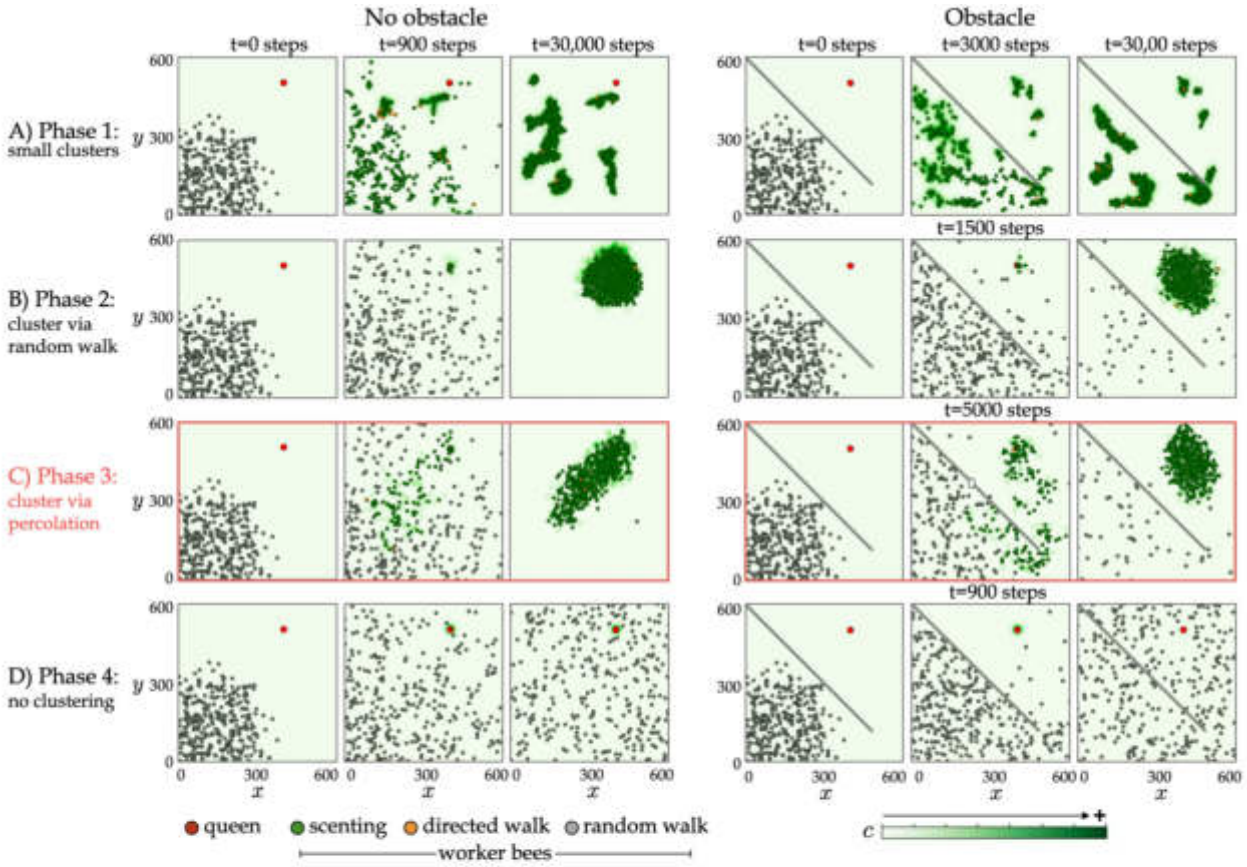


Figure 3: Simulations of four different phases. The queen is a red circle at the top right corner. Worker bees are circles colored by their internal state: scenting (green), performing a directed walk up the gradient (orange), and performing a random walk (gray). The instantaneous pheromone concentration $C(x, y, t)$ corresponds to the green color scale. Common simulation parameters are $N = 300$, $C_0 = 0.0575$, $D = 0.6$, and $\gamma = 108$. A) Phase 1 where bees aggregate into small clusters; $w_b = 0$, $T = 0.0001$ for both conditions. B) Phase 2 where bees cluster around the queen via random walk; $w_b = 30$, $T = 0.5$ for no obstacle; $w_b = 40$, $T = 0.2$ for obstacle. C) Phase 3 where bees create a percolating network of senders and receivers of the pheromone signal to cluster around the queen; $w_b = 50$, $T = 0.025$ for no obstacle; $w_b = 50$, $T = 0.075$ for obstacle. D) Phase 4 where no clustering occurs; $w_b = 60$, $T = 1.0$ for both.

obstacles, the bees generally require more time and exploration to form the scenting network. In Fig. 2B, we show snapshots of an example obstacle experiment ($N = 310$), in which bees search around the space behind the bar until a few bees find the opening and begin forming the collective scenting network at around 900 sec or 15 min. Most bees swarm around the queen by $t = 1800$ sec (30 min).

To quantitatively compare the aggregation process over time for the two conditions, we analyze the number of scenting bees over time (averaged over three experiments for each condition, with shaded area showing the standard deviation) in Fig. 2C. Without the obstacle, there is a sharp peak in the early phase of the experiment (around 350 sec) when bees quickly form the scenting network and a gradual decrease as most bees have clustered around the queen. In the presence of the obstacle, there is also a very early peak (around 50 sec); however, this peak of scenting occurs behind the obstacle before the bees find the opening. A smaller peak around 900 sec occurs when bees find the opening and forms a scenting network where the attractive surface shows the scenting directions oriented towards the queen (Fig. 2B). Further, we compared the average distance to the queen over time (Fig. 2D). The distance sharply decreases early in the absence of an obstacle. With an obstacle, the plateau from the start of the experiment until approximately 800 sec indicates the time the bees spend behind the obstacle until the first bees explore and find the opening.

Model shows constraints in behavioral parameter space in the presence of obstacles

The experimental results indicate the robustness of the collective scenting strategy in the presence of a physical obstacle and provide insights into the temporal dynamics of the aggregation process in different environments. The experi-

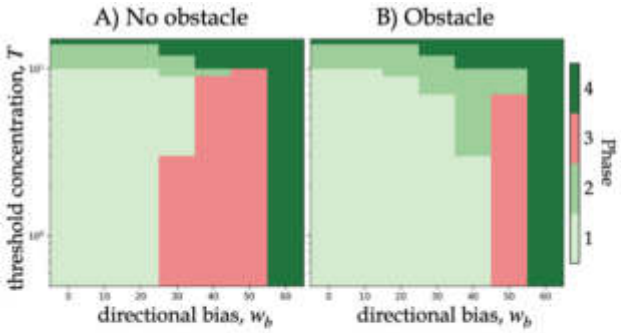


Figure 4: The effect of a physical obstacle on the phase boundaries for all simulations with varying w_b and T. Phase diagrams constructed from scenting model dynamics using summary heatmaps of the final number of clusters, the final queen's cluster size, and the distance of the farthest active bee to the queen. Phase 3 is highlighted in pink. A) Phase diagram without obstacle. B) Phase diagram with obstacle.

ments are an inspiration for the agent-based model where we further explore the behavioral parameters behind the mechanisms of the collective scenting and localization process. With and without physical obstacles, the model shows the four distinct phases defined in the Methods section:

Phase 1: Low values of both w_b and T produce small clusters of bees (Fig. 3A). Without an obstacle, signals reach the entire swarm and lead to clusters earlier than with an obstacle (around 900 and 3000 time steps, respectively).

Phase 2: High values of T produce swarms around the queen only via the bees that slowly reach the queen by random walk (Fig. 3B). When the obstacle is present, more time is required for bees to spread out throughout the arena (after 1500 time steps compared to by 900 time steps).

Phase 3: High values of w_b and low values of T lead to the percolation network of scenting (Fig. 3C). Without the obstacle, the network has begun to form by around 900 time steps, while it takes around 5000 time steps with the obstacle. Although bees in phases 2 and 3 eventually cluster at the queen's location, pheromone signals reach a much farther distance in phase 3 than in phase 2.

Phase 4: Very high values of T and w_b lead to no worker bees ever activated to scent or perform the directed walk up the gradient, and therefore no clustering (Fig. 3D).

Although all four phases are present in both conditions, the presence of the physical obstacle affects the phase areas and boundaries. The phase diagram of four phases as determined by (w_b, T) for simulations without the obstacle is shown in Fig. 4A. Treating the phase diagram as an image of a total of 36,100 pixels, phase 3 occupies approximately 11,925 pixels of 33.03% of the total diagram. When an obstacle is added to the arena, phase 3 only occurs when $w_b = 50$ and occupies approximately 4,500 pixels or 12.47% of the total diagram. Most of the parameter space that makes up phase 3 in the phase diagram for simulations

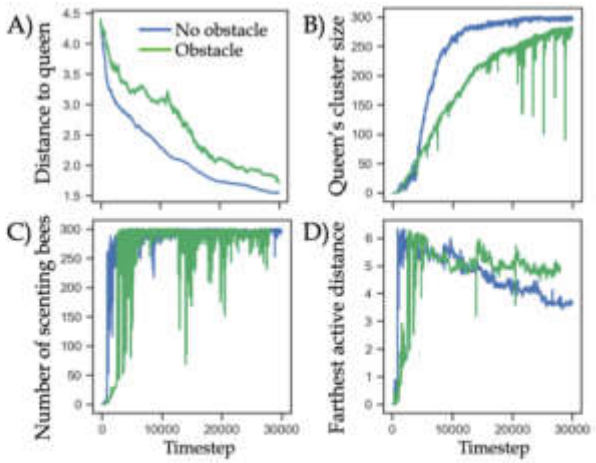


Figure 5: Time series of phase 3 simulations in Fig. 3C with and without obstacle (green and blue curves, respectively). A) The average distance of the worker bees to the queen over time. B) The average size of the queen's cluster over time. C) The average number of scenting bees over time. D) The average distance of the farthest scenting bee over time.

without obstacles becomes phase 1 in the diagram with the obstacle; the remaining phase 3 region becomes phase 2 in the diagram with the obstacle. For both conditions, phase 3 never occurs when there is no directional bias ($w_b = 0$) or when the concentration threshold is maximal ($T = 1.0$). Lastly, we compare the temporal dynamics of phase 3 simulations in Fig. 3C (time-series averaged over three repetitions). The average distance to the queen decreases faster and converges to a lower value when there is no obstacle (Fig. 5A). With the obstacle, we observe a slower decrease due to the exploration required to find the opening, and a convergence at a higher distance due to some bees that are still behind the obstacle by the end of the simulation. Similarly, the average growth in the queen's cluster size over time is faster and the final size is larger when there is no obstacle (Fig. 5B). Both the average number of scenting bees (Fig. 5C) and average distance of the farthest scenting bee (Fig. 5D) follows similar trends of sharp initial increase and plateau (number of scenting bees) or slight decrease (farthest active distance), but there is a delay in the sharp increase when the obstacle is present.

Discussion

Experimental studies show that bees employ the collective scenting strategy when localizing the queen and aggregating around her to form a coherent swarm (Nguyen et al., 2021b). This communication method is observed in a simple environment free of any perturbations as well as a more complex environment with the presence of physical obstacles (Nguyen et al., 2021b,a). The experimental results demonstrate a temporal delay in the peak of collective scenting and a slower decrease in the average bee distance to the queen as

the obstacle requires the bees to first explore the space and find the opening before forming the scenting network.

While the experiments illuminate the temporal dynamics of localization in the presence and absence of obstacles, we also turn to modeling to better understand how the behavioral parameter space is affected by the environmental perturbation. The model predicts four distinct outcomes, or phases, determined by the directional bias and concentration threshold, (w_b, T): 1) many small clusters, 2) clustering via only random walk, 3) clustering via signal propagation, and 4) no clustering. All four phases are present in both environments. However, the boundaries of the phases shift in the presence of the obstacle. Phase 4, dictated by very high T and very high w_b, remains constant. However, the successful aggregation of phase 3 occupies a smaller overlapping area of the phase diagram with the obstacle, constraining the parameters to only one value of high $w_b = 50$, while the phase spans w_b values of 30, 40, and 50 in the diagram for simulations without the obstacle. With low values of T, most of the area encompassing phase 3 in the diagram without obstacles becomes phase 1 in the diagram with obstacles—the physical wall leads to the breaking of the chains of scenting bees in the percolation network, resulting in the formation of small local clusters on both sides of the obstacle. The phase 3 region with high values of T becomes phase 2 with the obstacle, as the wall separates bees from one another and prevents the percolation network from forming.

Nguyen et al. (2021b) shows the effect of bee density on phase boundaries: As density increases, there are more bees to create and sustain the communication network, and the range of w_b and T and the area in the phase diagram for phase 3 is greater. In this study, the shrinking of phase 3 in the presence of the obstacle suggests that the wall has a similar effect to decreasing total density, by decreasing the effective density of scenting bees available to form a robust scenting percolation network. The bees are capable of navigating the obstacle environment to localize the queen and aggregate around her, but their range of potential behavioral parameters for the task is more limited and less flexible.

The model offers a simplified simulation of the bee collective scenting behavior. Some caveats and limitations that prevent the simulations from better matching the experimental data include the lack of spontaneous scenting observed in real bees but not modeled due to a lack of better understanding of the mechanisms. Further, bees in experiments seemingly stop scenting as they gather at the queen's cluster, while we simply allow bees to continue scenting without a stop function. Future analysis of the experiments is required to understand the mechanisms of these behaviors in order to improve the accuracy of the model.

Additional future directions include testing various densities in simulations with obstacles to further understand the density effect on the phase diagram and the temporal dynamics of aggregation. There may be a critical density below which the successful aggregation via propagation in phase 3 will not be present, as the obstacle dramatically reduces the effective density of scenting bees. Further, the physical obstacle in this study is relatively simple; more complex physical obstacles or other kinds of environmental perturbations (e.g., varying opening size in the obstacle, multiple obstacles, a maze, wind, or a moving queen) are of interest and can further shed light on how bee swarms navigate the complex, noisy environments found in nature. Lastly, understanding how the behavioral parameters shift in varying environments for the bees may further inform the design and development of swarm robotic systems to include a parameter space that allows for successful group coordination in the presence of physical obstacles.

Acknowledgements

This work was supported by the National Science Foundation Graduate Research Fellowship Program under Grant No. DGE1650115 (D.M.T.N.) and Physics of Living Systems Grant No. 2014212 (O.P.). Any opinion, findings, and conclusions or recommendations expressed in this material are those of the authors(s) and do not reflect the views of the NSF. We thank Chantal Nguyen for valuable feedback on this manuscript.

Appendix

Line-of-sight method. For a given line with end coordinates (x_1, y_1) and (x_2, y_2), its slope is $m = (y_2 - y_1)/(x_2 - x_1)$ and y-intercept is $b = -mx_1 + y_1$. The equation of the line in standard form, $Ax + By = C$, is: $mx - y = -b$, where the constant $A = m$, $B = -1$, and $C = -b$. The system of equations for the lines $\overline{CD}$ and $\overline{EF}$ is: $\begin{cases} m_{CD}x - y = b_{CD} \\ m_{EF}x - y = b_{EF} \end{cases}$

Transforming to a matrix equation to solve, where $A_{CD} = m_{CD}$, $A_{EF} = m_{EF}$, $B_{CD} = -1$, $B_{EF} = -1$, $C_{CD} = -b_{CD}$, $C_{EF} = -b_{EF}$: $\begin{bmatrix} A_{CD} & B_{CD} \\ A_{EF} & B_{EF} \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} C_{CD} \\ C_{EF} \end{bmatrix}$

Additional figures, table, and algorithm.

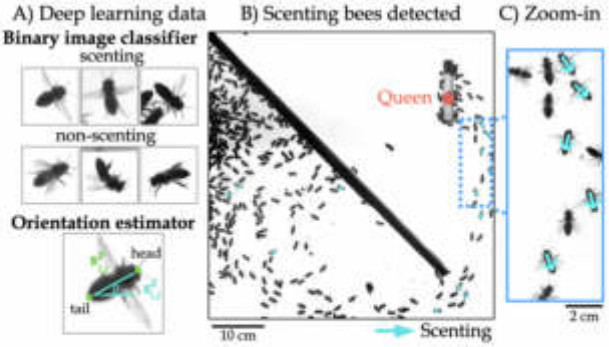


Figure A1: A) Training data examples for deep learning models to classify scenting bees and estimate orientations. B) Detections of scenting bees and their orientations (teal arrows). C) Zooming into scenting bees with wide wings.

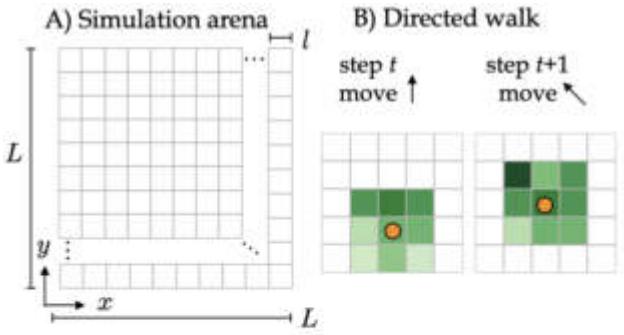


Figure A2: A) The $L \times L$ 2-d simulation arena is discretized into $l \times l$ sized pixels. B) When a bee detects a local pheromone concentration above the activation threshold ($C(x, y, t) > T$), it computes the gradient around it (using the nearest 9 pixels, highlighted in different shades of green) and walks up the gradient towards higher concentration.

Algorithm 1 Honey Bee Queen-Finding Algorithm

procedure SET ENVIRONMENTAL PARAMETERS
  Spatial: $X_{min}, X_{max}, \delta_X$
  Temporal: t_i, t_f, δ_t
  Pheromonal: D, γ
procedure SET QUEEN PARAMETERS
  $x_q, y_q, P_q, C_{0_q}, w_{b_q}$
procedure SET WORKER PARAMETERS
  x_w, y_w, P_w, C_{0_w}
  Free parameters: w_{b_w}, T, N_w
procedure SIMULATE QUEEN-FINDING
  Initialize Concentration Map
  for t_i in timesteps **do**
    ▷ Step 1. Check if each bee is emitting
    ▷ & Build list of pheromone sources
    ▷ Step 1.1. Check Queen
    if t_i % P_q == 0 **then**
      Queen emits pheromone via Eq. 3
      Add Queen to Pheromone Sources
    ▷ Step 1.2. Check Workers
    for worker_i in Workers **do**
      if state == emit **then**
        worker_i emits pheromone via Eq. 3
        Add worker_i to Pheromone Sources
    ▷ Step 2. Build Concentration Map
    ▷ & Compute gradient using Pheromone Sources
    for pheromone_source_i in Pheromone Sources **do**
      Update Concentration Map
      for worker_i in Workers **do**
        Calculate concentration gradient via Eq. 5
        Calculate directed emission direction
    ▷ Step 3. Update next state for Workers
    for worker_i in Workers **do**
      Update next state according to SM_{worker}

Algorithm 1: Queen localization and aggregation algorithm

Parameter	Meaning	Value
X_{min}	Min size of grid arena	-3
X_{max}	Max size of grid arena	3
δ_X	Grid cell size	0.01
t_i, t_f	Start and final time	0, 150
δ_t	Time integration constant	0.005
D	Diffusion coefficient	0.6
γ	Decay constant	108
(x_q, y_q)	Queen 2-D position	(1,-2)
P_q	Queen emission frequency	80
C_{0_q}	Queen initial concentration	0.0575
w_{b_q}	Queen bias scalar	0
(x_w, y_w)	Worker 2-D position	$\in [X_{min}, X_{max}]$
P_w	Worker emission period	80
C_{0_w}	Worker initial concentration	0.0575
N_w	Worker density	300
$\boldsymbol{w_{b_w}}$	**Worker bias scalar**	[0, 10, 20, 30, 40, 50, 60]
$\boldsymbol{T}$	**Worker activation threshold**	[1e-4, 1e-3, 0.01, 0.025, 0.05, 0.075, 0.1, 0.20, 0.25, 0.3, 0.4, 0.5, 0.6, 0.7, 1.0]

Table 2: Free parameters, w_b and T, of the ABM are bolded.

References

Abiyev, R., Ibrahim, D., and Erin, B. (2010). Navigation of mobile robots in the presence of obstacles. *Advances in engineering software*, 41(10-11):1179–1186.

Conte, Y. L. and Hefetz (2008). Primer pheromones in social hymenoptera. *Annu. Rev. Entomol.*, 53(1):523–542.

Dougherty, E. R. (1992). *An introduction to morphological image processing*. Society of Photo Optical.

Ester, M., Kriegel, H.-P., Sander, J., Xu, X., et al. (1996). A density-based algorithm for discovering clusters in large spatial databases with noise. In *Kdd*, volume 96, pages 226–231.

Fatès, N. (2010). Solving the decentralised gathering problem with a reaction–diffusion–chemotaxis scheme. *Swarm Intelligence*, 4(2):91–115.

Harker, M. and O'Leary, P. (2008). Least squares surface reconstruction from measured gradient fields. *CVPR*, pages 1–7.

Harker, M. and O'Leary, P. (2011). Least squares surface reconstruction from gradients - Direct algebraic methods with spectral, Tikhonov, and constrained regularization. *CVPR*.

He, K., 0006, X. Z., Ren, S., and 0001, J. S. (2016). Deep Residual Learning for Image Recognition. *CVPR*.

Lensky, Y. and Cassier, P. (1995). The alarm pheromones of queen and worker honey bees. *Bee world*, 76(3):119–129.

McIndoo, N. E. (1914). The scent-producing organ of the honey bee. *Proceedings of the Academy of Natural Sciences of Philadelphia*, pages 542–555.

Nguyen, D. M. T., Fard, G. G., Iuzzolino, M. L., and Peleg, O. (2021a). Robustness of collective scenting in the presence of physical obstacles. *Artificial Life and Robotics*, pages 1–6.

Nguyen, D. M. T., Iuzzolino, M. L., Mankel, A., Bozek, K., Stephens, G. J., and Peleg, O. (2021b). Flow-mediated olfactory communication in honeybee swarms. *Proceedings of the National Academy of Sciences*, 118(13).

Otsu, N. (1979). A threshold selection method from gray-level histograms. *IEEE transactions on systems, man, and cybernetics*, 9(1):62–66.

Peters, J. M., Gravish, N., and Combes, S. A. (2017). Wings as impellers: honey bees co-opt flight system to induce nest ventilation and disperse pheromones. *J. Exp. Biol.*, 220(Pt 12):2203–2209.

Rabin, M. O. (1963). Probabilistic automata. *Information and control*, 6(3):230–245.

Perpetual Crossers without Sensory Delay: Revisiting the Perceptual Crossing Simulation Studies

Eduardo J. Izquierdo, Gabriel J. Severino, Haily Merritt

Cognitive Science Program, Indiana University Bloomington
Luddy School of Informatics, Computing, and Engineering, Indiana University Bloomington
edizquie@iu.edu

Abstract

We revisit the perceptual crossing simulation studies, which are aimed at challenging methodological individualism in the analysis of social cognition by studying multi-agent real-time interactions. To date, all of these simulation studies have reported that it is practically impossible to evolve artificially a robust behavioral strategy without introducing temporal delays into the simulation. Also, all of the studies report on a single strategy: a perpetually crossing agent pair. Here, we systematically report on the evolutionary success of neural circuits on the perceptual crossing task, with and without sensory delay. We also report on two different strategies in the ensemble of successful solutions, only one of which had been discussed in the literature previously.

Introduction

Research on social cognition has largely assumed that studying a single individual engaged in a social interaction is sufficient to understand the dynamics and behavior that constitute a social interaction. In the last couple of decades, there have been calls to take the social interaction itself, instead of the individuals in isolation, as the object of study (Schilbach et al., 2013). From this interactionist perspective, social interaction is more than simply the arena in which social cognition plays out; it enables or constitutes social cognition (De Jaegher et al., 2010; Froese and Di Paolo, 2011).

Making social interaction the object of study, instead of a social agent, need not entail more complex experiments. One minimalist example of an interactionist experimental paradigm is perceptual crossing (Auvray et al., 2009). In these experiments, participants are asked to identify when they think they are interacting with a partner participant in a simple one-dimensional virtual environment while unaware of what they are actually interacting with. The tasks are designed to so that they cannot be solved by either participant independently; successful identification of the partner necessitates mutual interaction. The dynamics of behavior that result suggest that studies of social interaction should never be limited to analyzing a single individual's behavior. Recent work has expanded the paradigm to two dimensions (Rohde and Paolo, 2008; Rohde, 2010; Lenay et al., 2011), to the domain of human computer interaction (Barone et al., 2020) and to different populations, from adults and adolescents (Hermans et al., 2020; Froese et al., 2020; Iizuka et al., 2015; Froese et al., 2014; Lenay and Stewart, 2012) to individuals with autism (Zapata-Fonseca et al., 2018).

There has also been a growing interest in using simulation studies in order to investigate the dynamics of social interaction (Di Paolo, 2000; Quinn, 2001; Iizuka and Ikegami, 2004; Ikegami and Iizuka, 2007; Iizuka and Di Paolo, 2007; Williams et al., 2008; Di Paolo et al., 2008; Froese and Di Paolo, 2008; Reséndiz-Benhumea and Froese, 2020; Reséndiz-Benhumea et al., 2021). Some of these models have been specifically designed to generate insights for mutually informing collaborations between the field of artificial life and the traditional empirical sciences (e.g. Ikegami and Iizuka, 2007; Di Paolo et al., 2008; Rohde and Paolo, 2008). This is particularly true for the perceptual crossing paradigm, where there have been a set of rich contributions from simulation studies that have managed to successfully replicate the experiment and contribute to hypotheses to be tested in further social experiments (Di Paolo et al., 2008; Froese and Di Paolo, 2010). In particular, the simulation studies have predicted challenges and patterns of interactions that would be faced by human participants (Di Paolo et al., 2008). In some cases, these predictions have then been supported by experimental evidence from humans (Auvray et al., 2009), facilitating model-experiment dialogue (for a review see Auvray and Rohde, 2012).

Despite the advances, important questions remain open. First, simulation studies have all relied on the introduction of a sensory delay for the agents to perform the perceptual crossing task successfully (Di Paolo et al., 2008; Froese and Di Paolo, 2010, 2009). Crucially, the practical need for delays in the models has been considered a potentially important component for the explanation of the adaptive performance of the task in human participants, and has motivated psychological studies. However, the necessity of a sensory delay in human participants is unlikely (Iizuka et al., 2015). Second, the dominant (or potentially the only) strategy that has been discussed in the simulation literature so far has

been a perpetually crossing strategy, where agents continue to cross back and forth perpetually. It is unclear whether any other strategies are feasible. However, most of the simulation studies report on only one of the solutions, not on the full ensemble of possible solutions.

In this paper, we revisit the original work on the evolution of perceptual crossing agents and we extend this work to answer the open questions above. The rest of this paper is organized as follows. In the next section, we describe the perceptual crossing task and the set up of the agents for all experiments. Then we present results from a series of three experiments which explore the evolution of perceptual crossers under various conditions. Finally, we conclude with a general discussion of the experimental results, and outline some directions for future work.

Methods

We set out to replicate the agent and task as described in previous simulation studies (Froese and Di Paolo, 2010; Di Paolo et al., 2008; Froese and Di Paolo, 2009). Two agents coexist on a ring (i.e., a one-dimensional environment that wraps around; Fig. 1A). Agents are able to move around the ring with a maximum velocity (2 units of space per unit of time) in either direction. There are three distinct types of objects that can be encountered by an agent (Fig. 1A): the other agent's avatar, the shadow of the other agent, and a static object. Each object occupies a total of 2 units of space. The shadows are 48 units of space away from the agent. The ring is 600 units in circumference, and the fixed objects are placed across from each other at 150 and 450. Agents can move past each other and their respective static objects unimpeded. The neural controller that governs movement (described below) is rotated from one agent to the other, so that left and right movement aligns with the orientation of the agent. The shadows of the agents are reflected about the ring, so that one agent's shadow is to its left and the other agent's shadow is to its right, as depicted in Fig. 1A. The sensory input of an agent is activated (set to 1) when its receptor field overlaps with another object; otherwise it remains off (set to 0).

The behavior of each agent is controlled by a continuous-time recurrent neural network (Beer, 1995) with the following state equation:

$$\tau_i \dot{y}_i = -y_i + \sum_{j=1}^{N} w_{ji}\sigma(y_j + \theta_j) + g_i s + I_i \qquad (1)$$

where y_i is the state of each neuron, τ is the time constant, w_{ji} is the strength of the connection from the j^{th} to the i^{th} neuron, θ is a bias term, $\sigma(x) = 1/(1+e^{-x})$ is the standard logistic activation function, g_i is the sensory weight from the binary sensor s to neuron i, and I_i represents an external input to each neuron. The output of a neuron is

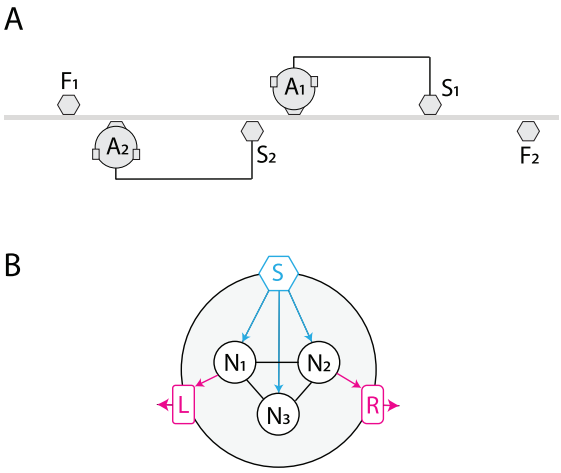


Figure 1: Task and agent setup. (**A**) The task takes place in a 1-dimensional ring where two agents face each other. Each agent can sense the other's avatar (A), a shadow of the other's avatar (S), and a fixed object (F). (**B**) Each agent has a sensor (cyan) that can send information to all N neurons (black). The neurons in the circuit are fully inter-connected, including self-connections (not depicted). The output from one neuron drives the left motor and another neuron drives the right motor (magenta). The neural circuits in the two agents are identical (i.e., they have the same parameters).

$o_i = \sigma(y_j + \theta_i)$. Following the original simulation studies (Fig. 1B), the sensor, s, is fully connected to all neurons in the circuit; the neurons are fully interconnected (including self-connections); and two of the neurons are chosen to drive the left and right motors, respectively. The velocity of an agent is proportional to the difference between the outputs of the two motor neurons: $v = \gamma(o_1 - o_2)$, where o_1 and o_2 represent the outputs of the neuron controlling the left and right motors, respectively, and γ is a constant that determines the agent's maximum possible velocity. In all of our simulations, the maximum velocity was set to $\gamma = 2$.

As with the original simulation studies, the two agents have identical neural controllers. The neural parameters of the controller are evolved using a real-valued genetic algorithm. Given that both agents are clones of each other in terms of their neural controller, each genome encodes the parameters for only one neural controller. The following neural parameters, with corresponding ranges, are evolved: time-constants $\tau \in [1, 10]$, biases $\theta \in [8, 8]$, and all connection weights (from sensors to neurons, g, and between neurons, w) $\in [8, 8]$. We used a generational algorithm with rank-based selection and a population size of 96 genotypes. Successive generations are formed by first applying random Gaussian mutations to each parent genome with a mutation variance of 0.05 (see Beer, 1996 for details). In addition, uniform crossover is applied with 50% probability. A child

replaces its parent if its performance is greater than or equal to that of the parent; otherwise the parent is retained.

The fitness evaluation is intended as a replication of the original simulation studies, such that neural controllers are evolved so that the two agents successfully find each other. We evaluate the performance of a pair of agents by systematically varying the starting location of the two agents. Specifically, the starting location for the first agent in a pair is chosen between 0 and 600 in steps of 50; the starting location for the second agent in the pair is between 0 and the first agent's starting location, for a total of 78 trials. Each trial lasts 800 time units and proceeds as follows: (1) the neural states of both agents are initialized to 0. (2) During the first 400 units of time, the agents interact without evaluation. We treat this as a transient period because it allows for agents initialized at the maximum starting distance moving at their maximum velocity enough time to traverse the ring environment and find each other. (3) For the remainder of the simulation after the transient period, we record and normalize the distance between the two agents. For a given trial, the score that a given pair of agents with a given neural controller receive is:

$$f = 1 - \frac{\bar{d} - 2}{298} \tag{2}$$

where $\bar{d}$ is the average separation between the two agents during a trial (excluding the initial transient period), 298 is the maximum spatial distance between the two agents. Since the 1-D environment wraps around between 0 and 600 units, 300 is the maximum spatial distance between points on the ring; and because the agents are 2-units wide and the sensors are binary, the agents cannot detect proximity beyond 2 units of space away from each other. The final fitness of the evaluation is the average fitness across all trials.

While we try to maintain as close a replication to the original study as possible, we summarize the key differences between the original fitness evaluation and ours. First, the distance during the initial transient period is not taken into consideration for the fitness here. Second, the fitness here is normalized to run between 0 and 1 based on the minimum distance at which an agent can sense the other agent. Finally, this evaluation is deterministic: the starting positions of the agents are deterministic, the position of the fixed objects does not change, and the relative position of both shadows to the agent is fixed. Additionally, there are two minor differences between our agent/task setup and the original study: the objects (the agent's avatars, the agents' shadows, and the static objects) occupy 2 units of space instead of 4; and the maximum velocity is 2 instead of 1. Also, the stochastic search algorithm used in our simulation is different from those used previously. As far as we can tell, each simulation study that has replicated the work has used different stochastic search algorithms, and we have no strong reason to believe that the results depend on the particularities of it. Crucially, as with the original studies, since the avatars, shadows, and fixed objects are *indistinguishable* to either agent, success in this task requires that the agents evolve a system for accurately detecting *mutual* interactions. As it has been demonstrated by previous implementations of the model, including only distance in the fitness function makes for a non-trivial task (Froese and Di Paolo, 2010, 2009).

Part I: Replicating original results

The goal of our first set of simulations was to replicate the experiments in the original studies. The agents must solve the perceptual crossing task with the original sensory delay of 2.5 units of time (25 timesteps using a step size of integration of 0.1) and with a fitness function that selects for close average proximity between the two agents. We report on the evolutionary performance across different circuit sizes and on two different strategies observed in the ensemble of successful perceptual crossers.

What general trends are observed in the evolution of perceptual crossers with sensory delay? One hundred evolutionary runs were performed for two-, three-, and four-node circuits (see Fig. 2). There are two main groups of solutions. The first and most dominant ones can be seen in the peak around 0.92 in the histograms, comprising 93% of all evolutionary runs with two-neuron circuits, 70% of three-neuron circuits, and 62% of four-neuron circuits. These solutions entirely fail to find the other agent from a small subset of the starting conditions. Because this group fails to solve the task from all possible starting configurations, we do not study them in any more detail in this paper. There is a second group of solutions that solve the problem nearly perfectly (>0.99). The size of this second group increased with the number of neurons: 1% of all two-neuron circuits, 10% of three-neuron circuits, and 16% of four-neuron circuits. We analyze the behavior of this group of solutions in more detail in the remainder of this section.

What are the overall tendencies observed in the behavior of successful perceptual crossers? We analyzed all solutions with a fitness greater than 0.99 across a wider range of conditions than were examined during evolution. Specifically, we analyzed the performance of this high-performing ensemble over 100×100 starting conditions across the full spatial range $[0, 600]$ and over a range of shadow distances $[48, 52]$. We kept track of the performance of these individuals in three different ways: (a) the performance as measured in the original simulations studies (i.e., with the transients and without normalization based on sensory-range; x-axis, Fig. 3A); (b) the performance measured without the initial transient and with normalization (y-axis, Fig. 3A); and (c) the number of times the two agents crossed each other in a trial (Fig. 3B). The results of the behavioral robustness analysis is shown for all 27 high-performing solutions in the ensemble in Fig. 3.

We highlight three key observations from this analysis. First, most of the circuits are behaviorally robust across a

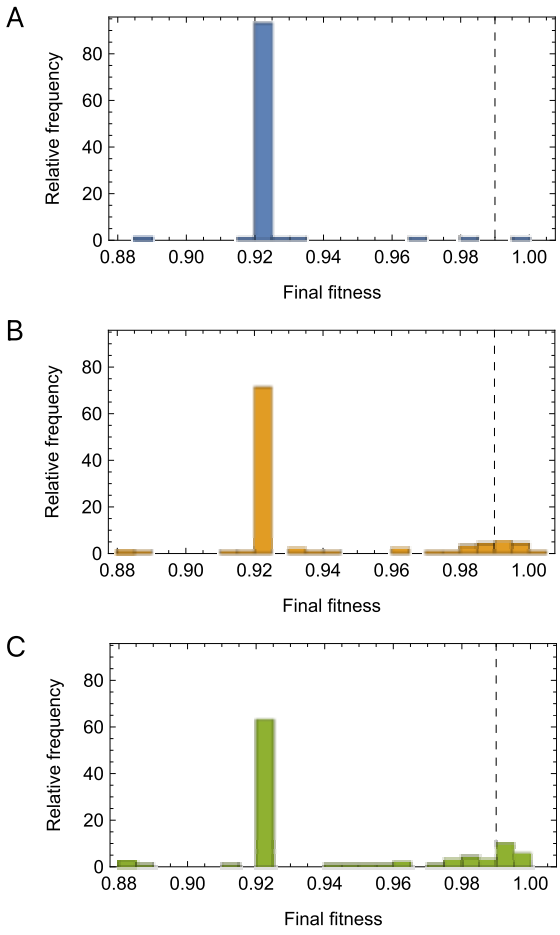


Figure 2: Evolutionary performance statistics. Performance histograms for two-neuron (**A**), three-neuron (**B**), and four-neuron (**C**) circuits. Relative frequency of binned proximity as a percentage of total trials is plotted. The maximum proximity is 1.0. The dashed line depicts the cutoff of 0.99 for agents analyzed in more detail.

wider set of conditions: 81.48% of the solutions maintained a performance greater than 0.95. This suggests that the conditions presented during evolution, including particularly a relatively small set of starting positions and a fixed shadow distance, were sufficient for agents to generalize. Second, our fitness function offers two improvements on the original fitness function: (a) Equally fit solutions no longer obtain a wide range of fitness; (b) The fitness of some great solutions is no longer indistinguishable from much worse solutions. The fitness function is one potential explanation for the difficulties evolving mentioned in the original simulation studies. Finally, not all successful solutions perpetually cross. This last observation was unexpected. As far as we gathered from the literature, there were no reports of minimally crossing solutions; only of perpetual crossing solutions.

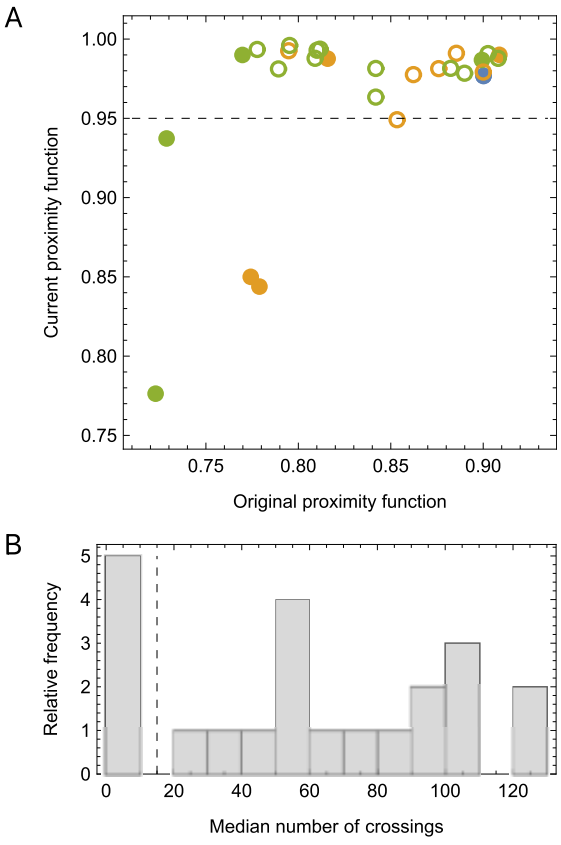


Figure 3: Behavioral robustness statistics. (**A**) Relationship between original and current measure of performance. Each point represents one solution from the top ensemble of solutions. The current measure of proximity excludes transients and normalizes the distance based on the sensory range. The color of the point represents the size of the circuit: two-neuron (blue), three-neuron (orange), and four-neuron (green). Filled disks represent solutions with minimal number of crossings. Open circles represent solutions that cross perpetually. (**B**) Median number of crossings across the solutions that achieved a robustness performance greater than 0.95 (above the dashed line in panel A). Solutions with fewer than 15 crossings (dashed line) per trial were labeled as minimal crossers. Solutions with greater than 15 crossings were labeled perpetual crossers.

Part II: Agents without sensory delay

Having replicated the original results, the second major goal of ours was to attempt to evolve perceptual crossers without a sensory delay. Given that previous reports always included a sensory delay (Di Paolo et al., 2008; Froese and Di Paolo, 2010, 2009), we did not expect these evolutionary runs to succeed. Nevertheless, it would be important and informative to understand how and why the no-sensory-delay

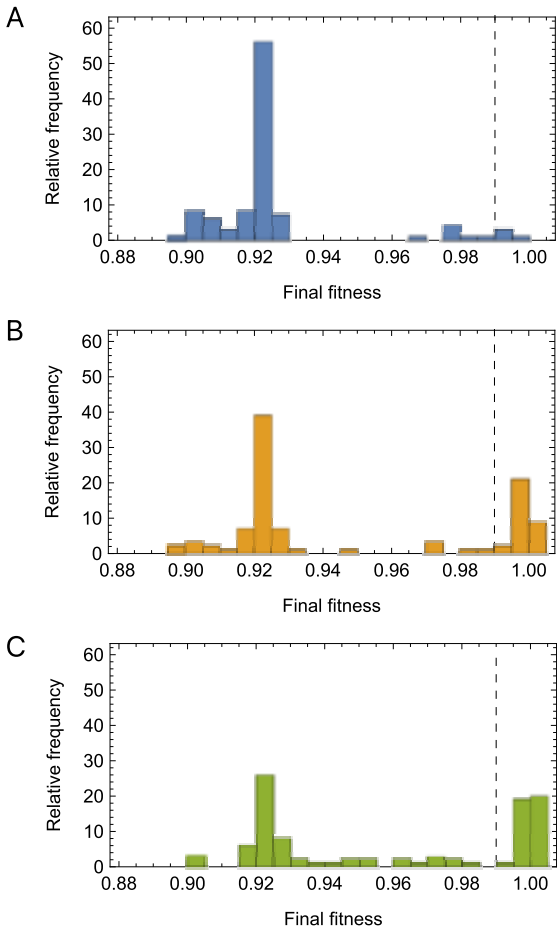


Figure 4: Evolutionary performance statistics for agents without sensory delay. Performance histograms for two-neuron (**A**), three-neuron (**B**), and four-neuron (**C**) circuits. Dashed line depicts 0.99 cutoff for selecting agents to analyze further.

condition failed.

We again performed one-hundred evolutionary runs for two-, three-, and four-node circuits to solve the perceptual crossing task, this time without a sensory delay. We maintained the same modified version of the fitness function that selects for proximity between the two agents. As in the previous experiment, there was a peak of evolutionary runs that became stuck around a fitness of 0.92 (Fig. 4). Crucially, and contrary to what had been reported until now, we observed that a substantial number of evolutionary runs succeeded (fitness$>$0.99). Moreover, the proportion of successful runs was substantially larger without a sensory delay than with it: 4% of all two-neuron circuits, 32% of three-neuron circuits, and 40% of four-neuron circuits.

Do solutions without a sensory delay generalize well such that agents find each other across a broad range of conditions? As with the first set of experiments, we performed a behavioral analysis of all 76 solutions with fitness > 0.99 (Fig. 5). We highlight the key insights from this analysis. First, solutions without the delay were largely robust to the wider range of starting conditions and to the different distances between an agent's shadow and avatar. All but one of the 76 solutions had performance above 0.95 using our updated proximity function (Fig. 5A). Second, only one of the 75 robust solutions crossed perpetually; all others crossed fewer than 15 times on average (Fig. 5B). Based on these results, we define minimal crossers as pairs of agents that cross each other fewer than fifteen times on average and perpetual crossers as pairs of agents that cross each other continuously, or more than fifteen times. This last finding prompted the next set of experiments.

Part III: Promoting perpetual crossing

When agents were evolved with sensory delay, we replicated the successful findings of the original simulation studies. Notably, in addition to the perpetual crossing strategy that had been reported originally, we observed a second strategy: minimal crossers (i.e., pairs of agents that crossed each other fewer than 15 times on average within a given trial). When agents were evolved without sensory delay, we were surprised to find that they still succeeded at finding each other, but primarily using the minimally crossing strategy. Given that the previous simulation studies primarily focused on analyzing solutions that cross perpetually, the natural follow up question was: Can we reliably generate agents without sensory delay that solve the problem with a perpetual crossing strategy?

In this third and final set of experiments, we set out to evolve agents without sensory delay to find each other and cross perpetually. Our goal was to accomplish this by only redesigning the fitness function to more closely match the desired behavioral goal. Specifically, we introduced a term that promoted perpetual crossing, in addition to proximity. Including the additional term allows us to ask whether the perpetual crossing strategy only arises in the presence of sensory delay. Thus, here our motivation was more about refining and further understanding the modeling approaches to perceptual crossing than directly replicating perceptual crossing work in humans.

The additional perceptual crossing term simply counted the number of times two agents crossed and averaged this across all the different starting conditions. We explored three different ways to introduce this additional term and we only succeeded with one. In one batch of experiments, we multiplied or added the proximity term and the crossing-count term together. In a second batch, we used an incremental approach in two stages: in the first stage, only proximity was evaluated; in the second stage, the proximity term and the crossing-count term were again multiplied or added together. Neither of these two strategies successful results.

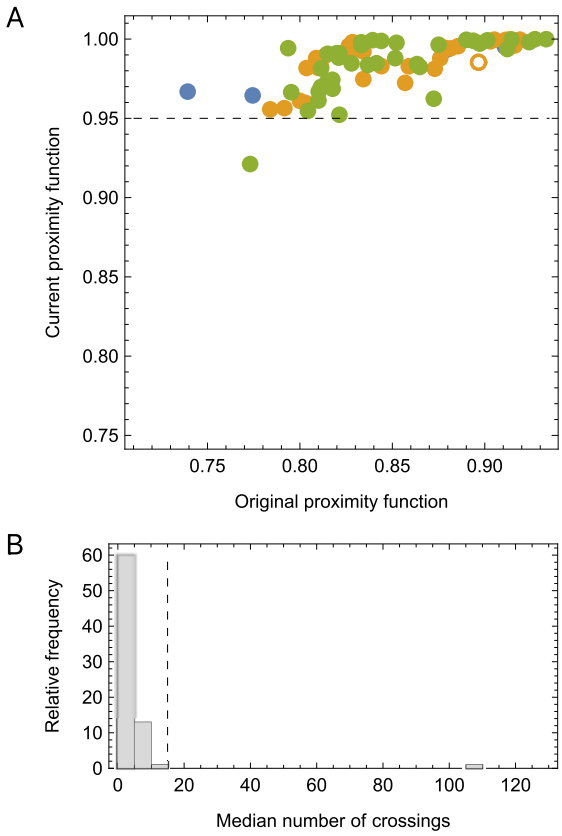


Figure 5: Behavioral robustness statistics of agents with no sensory delay. (**A**) Relationship between original and current measure of performance. Each point represents one solution from the top-performing ensemble. Color represents circuit size: two-neuron (blue), three-neuron (orange), and four-neuron (green). Filled disks represent solutions with minimal number of crossings. One open circle represents a perpetual crossing solution. (**B**) Median number of crossings across the solutions that achieved a robustness performance greater than 0.95 (above the dashed line in panel A).

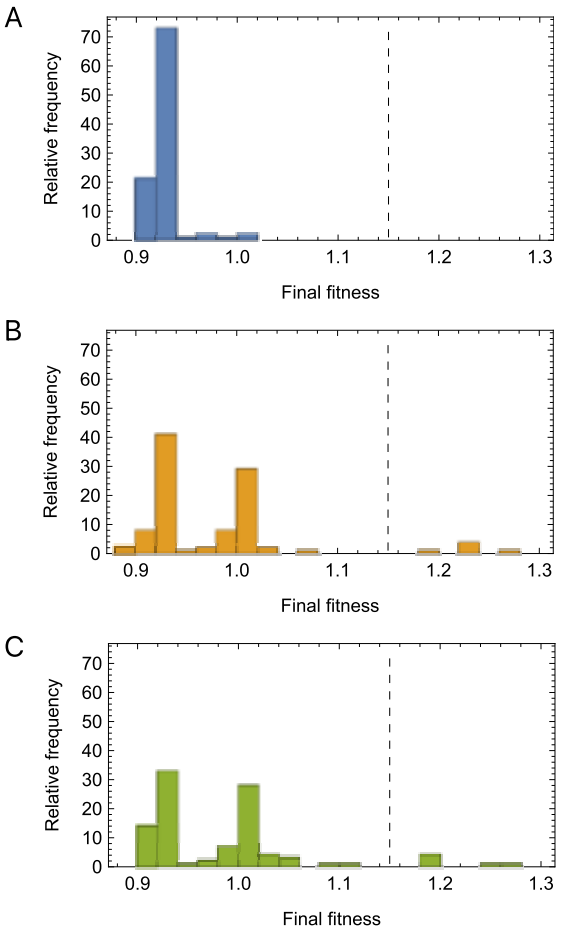


Figure 6: Evolutionary performance statistics for agents without sensory delay and with a fitness function that encourages crossing. Performance histograms for two-neuron (**A**), three-neuron (**B**), and four-neuron (**C**) circuits. The dashed line depicts the cutoff of 1.15 for selecting agents to analyze in more detail.

In the final batch of experiments, we employed a conditional fitness evaluation: if the proximity term was lower than 0.99, only it counted towards fitness; if the proximity term was higher than 0.99, then the fitness involved the sum of the proximity and the crossing-count term. Using this formulation, agent pairs can achieve a fitness greater than 1. Only with this approach did we obtain successful perpetual crossers.

Is it possible to evolve perpetual crossers without a sensory delay? We performed the final set of one-hundred evolutionary runs for two-, three-, and four-node circuits. As with all previous experiments, there was again a large number of runs that became stuck around a fitness of 0.92 (Fig. 6). In this run, a smaller batch of runs got stuck also at a fitness of around 1. This corresponds to solutions that can find each other perfectly, but do not cross perpetually. Only a small fraction of the evolutionary runs surpassed both challenges. Based on observations of the behaviors, we counted the number of successful solutions as those that surpassed a fitness of 1.15 on the combined task: none of the two-neuron circuits, 6% of three-neuron circuits, and 6% of four-neuron circuits. All 12 of the successful perpetual crossing solutions had a performance above 0.95 on the robustness test (Fig. 7A); and all of them had a median number of crossings above 80 (Fig. 7B).

What can we learn from the behavior of a perceptual crossing agent without sensory delay that crosses perpetually? Although a detailed analysis of the dynamics of one of these circuits is outside the scope of this contribution, we

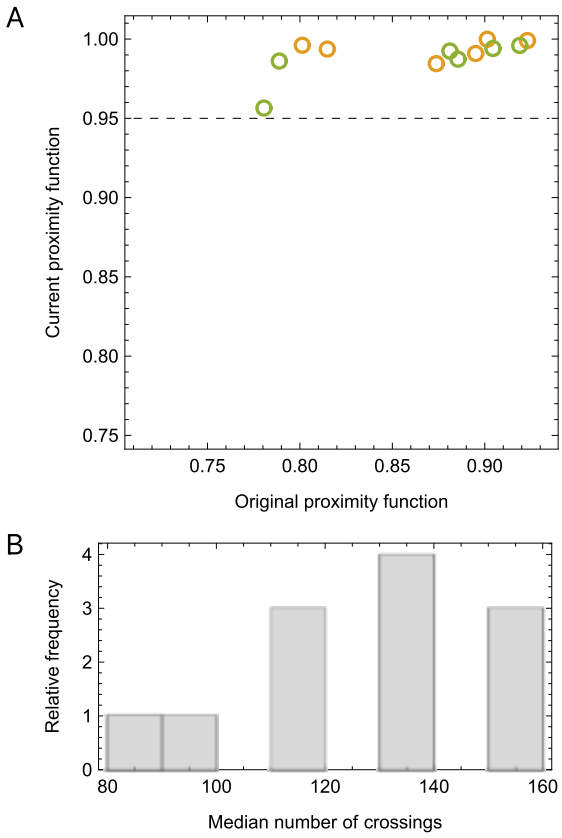


Figure 7: Behavioral robustness statistics for agents without sensory delay evolved to cross perpetually. (**A**) Performance of the best solutions (each point) using the original and the current measure of proximity. Color represents circuit size: two-neuron (blue), three-neuron (orange), and four-neuron (green). All solutions are perpetual crossers, as depicted by the open circles. (**B**) Median number of crossings across the solutions that achieved a robustness performance greater than 0.95 (region above dashed line in panel A).

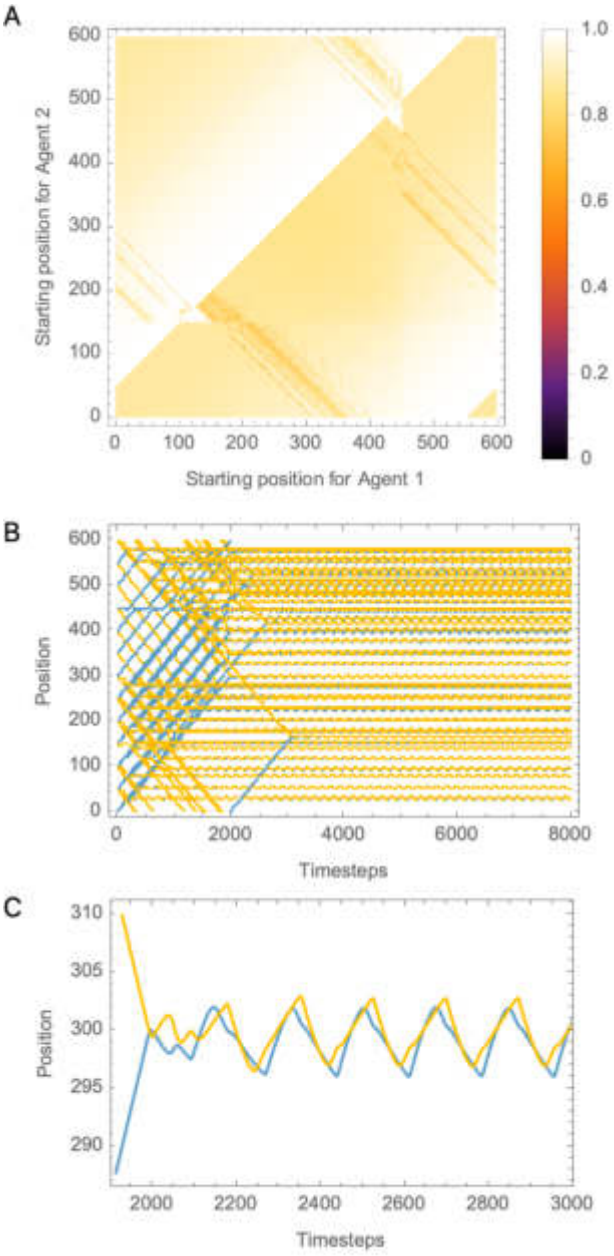


Figure 8: Behavior of one of the top three-neuron circuits without sensory delay. (**A**) Proximity performance of the two agents as a function of their starting positions. Color indicates proximity performance. (**B**) A sample of 78 trials from the full range of different starting conditions as examined during the fitness evaluation. (**C**) Detailed look at the interaction between two agents for one trial.

can learn something about the operation of these circuits by looking at examples of their behavior. In this final section, we visualize the behavior of one of the top three-neuron circuits without sensory delay in three stages of detail (Fig. 8). First, we visualize the average proximity performance of the two agents throughout the full duration of a trial as a function of a wide range of starting conditions (100×100) (Fig. 8A). It is important to keep in mind that in this solution, like in most of the successfully evolved solutions, the agents always find each other and remain close to each other thereafter. This map of performance, then, does not reflect their ability to find each other, but rather how much exploration the agents exhibit as a function of the starting positions. A performance of 1.0 represents starting configurations where the agents find each other early in the evaluation trial; while a performance of 0.8 represents agents that find each other after the first 200 units of time. As a second step, we can observe the agents' movement in the one-dimensional ring

over time given a smaller set of starting conditions (78 total) across all of time (Fig. 8B). One thing that is interesting to note here is that these agents find each other somewhat uniformly along the length of the environment; other solutions in the ensemble exhibited different patterns, and not always uniform. Finally, we can visualize a single trial over the small window of time where the agents first interact and then maintain a mutual crossing (Fig. 8C). One thing to note is that the pattern of crossing was quite different across all 12 of the different top three- and four-node circuits. For some of the solutions, the absolute position of the cycle of crossing stayed constant; for other solutions the pair drifted slowly in time while continuing to cross around each other. Overall, the main take-home message from looking at examples of the behavioral trajectories of some of these agents is that there is a wide variety of patterns of behavior according to which they could be grouped.

Discussion

In this paper, we set out to replicate the perceptual crossing simulation studies (Froese and Di Paolo, 2010; Di Paolo et al., 2008; Froese and Di Paolo, 2009) and refine the approaches used. First, we observed that evolving agents with a sensory delay resulted in two clearly distinct behavioral strategies: perpetual crossers (agents that find each other and continuously cross) and minimal crossers (agents that stop moving after crossing each other a limited number of times). As far as we are aware, only the former had been reported in the literature (Froese and Di Paolo, 2010; Di Paolo et al., 2008; Froese and Di Paolo, 2009). Presumably, perpetual crossers are preferred because they continuously interact. Second, we succeeded at artificially evolving agents without the sensory delay, contrary to what has been previously reported (Di Paolo et al., 2008; Froese and Di Paolo, 2010). However, an analysis of the successful solutions revealed that nearly all of them adopted a minimally crossing strategy. Finally, by modifying the fitness function to select for both proximity and crossings, we were able to generate agents without sensory delay that adopted the perpetually crossing strategy.

There are two factors that are likely to have played an important role in the success of the evolutionary runs in our simulation studies. First, by including the initial transient of the behavioral trajectories of the agents in each trial, the fitness function in the original simulation studies blurred the performance of otherwise successful circuits (c.f. Fig. 3A). By eliminating the transients, we could measure with more precision the percentage of trials where agents found each other. Second, the original simulation studies involved a stochastic fitness evaluation, varying the starting position of the agents and the distance of the shadows in each trial. The purpose of this was to make sure the agents learned the task robustly. However, for this task, a deterministic fitness evaluation was sufficient to produce equally robust solutions.

Finally, the positioning of the shadow in these experiments led to some initial exploratory and counter-intuitive results. In every one of the formulations of the task that we examined (Froese et al., 2014; Froese and Di Paolo, 2010), there was no explicit statement on whether the shadow was positioned to the left or the right of the agent/participant. Most crucially, it was not stated whether the shadow of one agent was reflected or rotated with respect to the other agent. However, in all the schematics of the task (Froese et al., 2014; Froese and Di Paolo, 2010), except for one of them (Auvray and Rohde, 2012), the shadow appeared to be reflected. Importantly, the original paper (Auvray et al., 2009) has a schematic that represents the shadows as reflected. In preliminary experiments, we examined both conditions (although we only report here on the reflected condition). The rotated-shadow condition provided counter intuitive results. Although we might predict that the task would be impossible because agents would end up mutually oscillating around each other's shadows, thinking they have found the other agent, evolution reliably found a clever hack that relied on the symmetry of the nervous systems. Because the two agents start moving in the same direction (left or right), they always encounter the other's avatar and shadow in the same sequence: one first and then other (depending on the direction). This allows them to "hardcode" which stimulus to center on without the requirement to mutually interact. This is, of course, not a possibility for the condition where the shadows are reflected.

Future Work

The next step for future work is to perform detailed analyses of the evolved perceptual crossers without sensory delay. Using the mathematical tools of dynamical systems theory, one can identify how the underlying dynamical structure of the agents supports their joint interaction. A psychophysical analysis might illuminate additional differences between the perpetual and minimal crossers that we identify. Such an analysis might also be useful for connecting our results to empirical work on humans, further strengthening the model-experiment loop that the perceptual crossing paradigm has established. Of particular interest would be to run new empirical experiments with humans to test whether participants can be prompted to use perpetual versus minimal strategies depending on either the sensory-delay condition and/or different variations of the task prompt.

Acknowledgements

We are thankful for the reviewers' comments. This material is based upon work supported by the National Science Foundation under Grant No. 1845322 and NSF Research Traineeship Grant No. 1735095, "Interdisciplinary Training in Complex Networks and Systems."

References

Auvray, M., Lenay, C., and Stewart, J. (2009). Perceptual interactions in a minimalist virtual environment. *New Ideas in Psychology*, 27(1):32–47.

Auvray, M. and Rohde, M. (2012). Perceptual crossing: the simplest online paradigm. *Frontiers in Human Neuroscience*, 6:181.

Barone, P., Bedia, M. G., and Gomila, A. (2020). A minimal turing test: reciprocal sensorimotor contingencies for interaction detection. *Frontiers in Human Neuroscience*, 14:102.

Beer, R. D. (1995). On the dynamics of small continuous-time recurrent neural networks. *Adaptive Behavior*, 3(4):469–509.

Beer, R. D. (1996). Toward the evolution of dynamical neural networks for minimally cognitive behavior. *From Animals to Animats*, 4:421–429.

De Jaegher, H., Di Paolo, E., and Gallagher, S. (2010). Can social interaction constitute social cognition? *Trends in Cognitive Sciences*, 14(10):441–447.

Di Paolo, E. A. (2000). Behavioral coordination, structural congruence and entrainment in a simulation of acoustically coupled agents. *Adaptive Behavior*, 8(1):27–48.

Di Paolo, E. A., Rohde, M., and Iizuka, H. (2008). Sensitivity to social contingency or stability of interaction? modelling the dynamics of perceptual crossing. *New Ideas in Psychology*, 26(2):278–294.

Froese, T. and Di Paolo, E. A. (2008). Stability of coordination requires mutuality of interaction in a model of embodied agents. In *International Conference on Simulation of Adaptive Behavior*, pages 52–61. Springer.

Froese, T. and Di Paolo, E. A. (2009). Toward minimally social behavior: social psychology meets evolutionary robotics. In *European Conference on Artificial Life*, pages 426–433. Springer.

Froese, T. and Di Paolo, E. A. (2010). Modelling social interaction as perceptual crossing: an investigation into the dynamics of the interaction process. *Connection Science*, 22(1):43–68.

Froese, T. and Di Paolo, E. A. (2011). The enactive approach: Theoretical sketches from cell to society. *Pragmatics & Cognition*, 19(1):1–36.

Froese, T., Iizuka, H., and Ikegami, T. (2014). Embodied social interaction constitutes social cognition in pairs of humans: a minimalist virtual reality experiment. *Scientific Reports*, 4(1):1–10.

Froese, T., Zapata-Fonseca, L., Leenen, I., and Fossion, R. (2020). The feeling is mutual: clarity of haptics-mediated social perception is not associated with the recognition of the other, only with recognition of each other. *Frontiers in Human Neuroscience*, 14:373.

Hermans, K. S., Kasanova, Z., Zapata-Fonseca, L., Lafit, G., Fossion, R., Froese, T., and Myin-Germeys, I. (2020). Investigating real-time social interaction in pairs of adolescents with the perceptual crossing experiment. *Behavior Research Methods*, 52(5):1929–1938.

Iizuka, H. and Di Paolo, E. A. (2007). Minimal agency detection of embodied agents. In *European Conference on Artificial Life*, pages 485–494. Springer.

Iizuka, H. and Ikegami, T. (2004). Adaptability and diversity in simulated turn-taking behavior. *Artificial Life*, 10(4):361–378.

Iizuka, H., Saitoh, S., Marocco, D., and Yamamoto, M. (2015). Time delay effect on social interaction dynamics. In *Proceedings of the 3rd International Conference on Human-Agent Interaction*, pages 217–219.

Ikegami, T. and Iizuka, H. (2007). Turn-taking interaction as a cooperative and co-creative process. *Infant Behavior and Development*, 30(2):278–288.

Lenay, C. and Stewart, J. (2012). Minimalist approach to perceptual interactions. *Frontiers in Human Neuroscience*, 6:98.

Lenay, C., Stewart, J., Rohde, M., and Amar, A. A. (2011). "you never fail to surprise me": The hallmark of the other: Experimental study and simulations of perceptual crossing. *Interaction Studies*, 12(3):373–396.

Quinn, M. (2001). Evolving communication without dedicated communication channels. In *European Conference on Artificial Life*, pages 357–366. Springer.

Reséndiz-Benhumea, G. M. and Froese, T. (2020). Enhanced neural complexity is achieved by mutually coordinated embodied social interaction: A statespace analysis. In *Second International Workshop on Agent Based Modelling of Human Behaviour*.

Reséndiz-Benhumea, G. M., Sangati, E., Sangati, F., Keshmiri, S., and Froese, T. (2021). Shrunken social brains? a minimal model of the role of social interaction in neural complexity. *Frontiers in Neurorobotics*, 15:72.

Rohde, M. (2010). *Enaction, embodiment, evolutionary robotics: simulation models for a post-cognitivist science of mind*, volume 1. Springer Science & Business Media.

Rohde, M. and Paolo, E. D. (2008). Embodiment and perceptual crossing in 2d. In *International Conference on Simulation of Adaptive Behavior*, pages 83–92. Springer.

Schilbach, L., Timmermans, B., Reddy, V., Costall, A., Bente, G., Schlicht, T., and Vogeley, K. (2013). Toward a second-person neuroscience 1. *Behavioral and Brain Sciences*, 36(4):393–414.

Williams, P. L., Beer, R. D., and Gasser, M. (2008). Evolving referential communication in embodied dynamical agents. In *Eleventh International Conference on the Simulation and Synthesis of Living Systems*, pages 702–709.

Zapata-Fonseca, L., Froese, T., Schilbach, L., Vogeley, K., and Timmermans, B. (2018). Sensitivity to social contingency in adults with high-functioning autism during computer-mediated embodied interaction. *Behavioral Sciences*, 8(2):22.

PPS^{3D}: A 3D Variant of the Primordial Particle System

Martin Stefanec[1] and Thomas Schmickl[1]

[1]Artificial Life Lab, Institute of Biology, Field of Excellence COLIBRI, University of Graz, Austria
martin.stefanec@uni-graz.at

Abstract

We demonstrate here that the morphogenetic Primordial Particle System, which was originally defined for two dimensions only, can also operate with minimal adaptations in a three-dimensional setting, producing similar life-like structures and dynamics.

Introduction

Self-organizing morphogenetic complexity arising from simple local interactions is a core topic of Artificial Life. Systems like the Game of Life [1] and many of its variants, e.g., HighLife [2], produce complex emergent phenomena in discrete 2D space. The search for a 3D variant of such systems is ongoing, although several candidates have been suggested [3,4]. With a comparable level of microscopic simplicity, the Primordial Particle System (PPS) shows comparable emergent macroscopic features in continuous 2D space, as a system of self-propelled particles [5] following a simple motion law (Equ. 1). Here we extend PPS to 3D space as the new PPS^{3D}.

The PPS Mechanics: From 2D to 3D

Our original 2D PPS [6] is a set Π of particles. In each time step t, each particle $i \in \Pi$ is defined by its position in space $(x(i,t), y(i,t))$, and by its orientation $\varphi(i,t)$. Particles move with velocity v and rotate with the following motion law

$$\frac{\Delta\varphi}{\Delta t} = \alpha + \beta \cdot sgn(R(i,t) - L(i,t)) \cdot N(i,t), \qquad (1)$$

where $R(i,t)$ and $L(i,t)$ are the number of other particles inside of the semicircle right and left of the particle within radius r, thus the total neighborhood size of each particle i is $N(i,t) = R(i,t) + L(i,t)$. The parameter α gives a constant rotation, while β scales the density-dependent rotation. The function $sgn()$ refers to the signum function, returning the sign of a value as +1, 0, or -1. The 2D PPS described in [6] is $PPS \leftarrow [\alpha = 180°, \beta = 17°, v = 0.67, r = 5.0]$. We here extend this system to 3D space, where each particle's position is defined by $(x(i,t), y(i,t), z(i,t))$ and each heading is $(\varphi(i,t), \psi(i,t))$. $\varphi(i,t)$ is the pitch and $\psi(i,t)$ is the yaw of the particle, there is no roll rotation in this system. Our focal PPS^{3D} is $PPS \leftarrow [\alpha = 180°, \beta = 24°, v = 0.67, r = 3.5]$. In order to keep computational loads in 3D space low, we decrease the neighborhood radius to $r = 3.5$ and therefore increase the neighborhood-dependent rotation coefficient to $\beta = 24°$. In a PPS^{3D}, the number of particles within the hemispheres left $L(i,t)$ and right $R(i,t)$ of each particle i determines its yaw rotation, and the number of particles within the hemisphere above $A(i,t)$ and below $B(i,t)$ each particle i determines its pitch rotation. Particles start with randomized initial positions and orientations. We simulated 1600 particles in a wrapped cubic space of 40 voxels side-length. We use Euler integration to solve the system for a period of $0 \leq t \leq T$ with $\Delta t = 1.0$. Figure 1 details the update procedure.

Implementing the microscopic rules of PPS^{3D} in a nutshell

∀ *time steps* $t \in \{1, 2, 3, ..., T\}$: (sequential oder)

∀ *particles* $i \in \Pi$: (randomized order)

YAW

1. Measure $R(i,t)$ and $L(i,t)$ within radius r
2. Calculate $N(i,t) = R(i,t) + L(i,t)$
3. Update (rotate) the particle's yaw $\psi(i,t)$:
   $\frac{\Delta\psi}{\Delta t} = \alpha + \beta \cdot sgn(R(i,t) - L(i,t)) \cdot N(i,t)$
4. Update $x(i,t)$, $y(i,t)$ *and* $z(i,t)$:
   Move particle forward by a distance of $v/2$

PITCH

5. Measure $A(i,t)$ and $B(i,t)$ within radius r
6. Update $N(i,t) = A(i,t) + B(i,t)$
7. Update (rotate) the particle's pitch $\varphi(i,t)$:
   $\frac{\Delta\varphi}{\Delta t} = \alpha + \beta \cdot sgn(A(i,t) - B(i,t)) \cdot N(i,t)$
8. Update $x(i,t)$, $y(i,t)$ *and* $z(i,t)$:
   Move particle forward by a distance of $v/2$

COL

9. Update final particle color depending on $N(i,t)$

Figure 1: Particle update procedure in a PPS^{3D}.

Particles' colors shown in Figure 3 indicate their participation in emergent structures. Particle colors depend purely on the local particle density. With $N(i,t) \leq 7$ they are green (*nutrients*), with $7 < N(i,t) \leq 12$ they are brown (*immature spores*), with $12 < N(i,t) \leq 20$ they are blue (*cell surface*). With $N(i,t) > 20$ they are yellow (*cell core*). If the number of other neighboring particles exceeds 7 within a small radius ($r = 1.3$), a particle's color will be magenta (*mature spore*).
Our main goal here is to show that a PPS with almost the same microscopic set of rules can also work well in 3D if the rotational motion law for the rotations yaw and pitch are executed in an alternating regime. As a measure of success,

we investigate if the PPS^{3D} shows the characteristic properties of the original PPS: (A) Spontaneously forming *spores* from freely roaming *nutrients*. (B) Growth of such *spores* to *cells* if enough *nutrients* are available. (C) Division of *cells* into two *spores* or (D) into two *cells*. (E) Long-term habitat conquest by emerging structures. (F) Logistic growth of populations.

Results

For space constraints we show mostly exemplary snapshots here, a video that shows the full runs of the features (A-E) is available online [7]. Figure 3A shows that initially randomly distributed particles can first aggregate to an *immature spore* (brown), which then can attract more particles to form a *mature spore* (magenta). Figure 3B shows that such a *spore* can grow into a *cell* structure, with a dense core (yellow) and a medium-density outer layer (blue). Such a *cell* can then either divide into two *spores* (Figure 3C) or into two *cells* (Figure 3D). Figure 3E shows that – starting from a single emerging spore – the whole habitat gets conquered and altered due to the replication process in PPS^{3D}. We found emergent *spores* to consist of 14±1 and emergent *cells* of 30±13 particles. Figure 3F shows the population dynamics of this process, by showing the median number of particles assigned to *cells* and *spores*, respectively, from 24 different runs, as well as the population dynamics of the exemplary run shown in Figure 3E. We fitted (LSD method) a variant of the classic Lotka-Volterra competition model [8,9] to our data (Figure 3F), describing the population dynamics of particles in *cells* $C(t)$, in *spores* $S(t)$, and as free *nutrients* $F(t)$. The macroscopic OΔE-model captures the PPS^{3D} dynamics well (Figure 2). RS and RC are the populations' growth rates, KS and KC are carrying capacities and b is a competition coefficient. We start the model with one cell and one spore of minimal size.

Macroscopic OΔE model of emerging PPS^{3D} dynamics

$\Delta S/\Delta t = RS \cdot S(t) \cdot (1 - (S(t) + b \cdot C(t))/KS)$,
$\Delta C/\Delta t = RC \cdot C(t) \cdot (1 - C(t)/KC)$, and
$F(t) = 1600 - S(t) - C(t)$ with $\Delta t = 1.0$ and with a parametrization of $RS = 0.0014$, $KS = 270$, $b = 0.75$, $RC = 0.0023$, $KC = 136$, $S(0) = 13$, $C(0) = 17$.

Figure 2: Macroscopic model of the PPS^{3D} dynamics.

Discussion & Conclusion

We translated the original PPS with only minimal adaptations into a three-dimensional system, keeping all emergent structures and dynamics that make the PPS interesting. The shapes, the behaviors, and the population dynamics of emergent structures described in Figure 3 resemble the observations from the original 2D version of the PPS [6] closely. It is yet open for future investigations, how and why such a simple rotational motion law of particles can yield a "virtual ecosystem" of emergent life-like structures. In contrast to other morphogenetic systems operating in continuous space, e.g. self-propelled particles in 'swarm chemistry' (SC) [10-12] or wave-based models in 'Lenia' [13], a PPS is microscopically (far) less complex. But, as a PPS is based on permanently moving particles, the emerging structures are less stable compared to Lenia or SC, which also made the 2D-to-3D transition recently [13,14]. We followed the classic ALIFE approach of "complexity from simplicity" intentionally: Keeping the system's microscopic simplicity, while also keeping its macroscopic complexity, contributes to the feasibility of a future physical embodiment of this system, e.g., with an autonomous swarm of loosely coupled micro- or nanorobots – a step towards active reconfigurable matter.

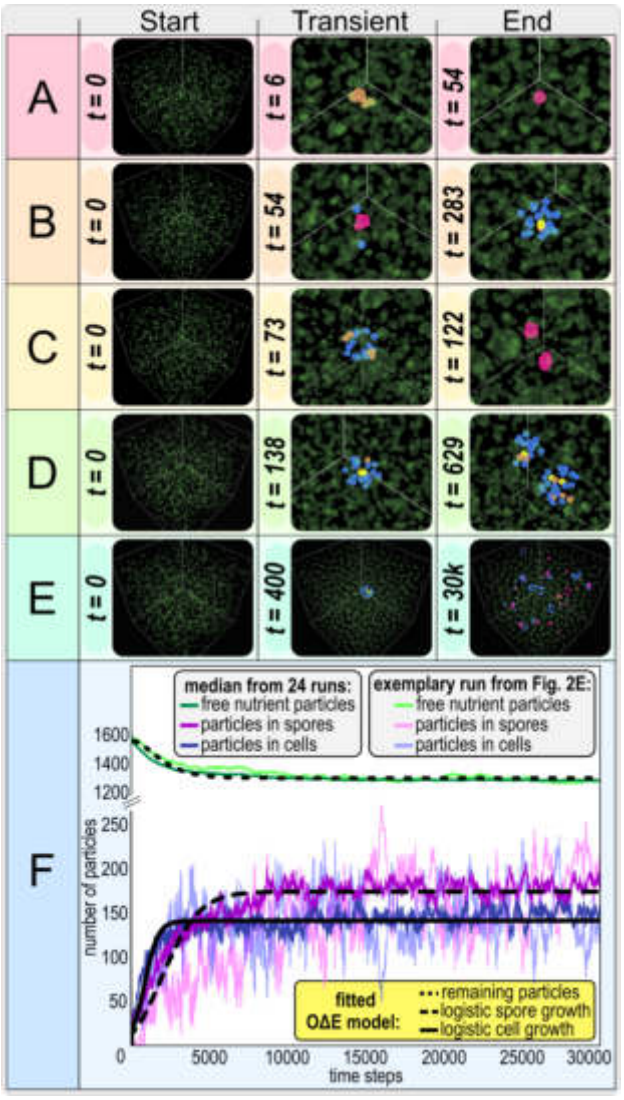


Figure 3: We found all significant macroscopic properties emerging in a PPS^{3D}, which are known from the original PPS.

Acknowledgements

Supported by the Field of Excellence COLIBRI (Complexity of Life in Basic Research & Innovation), University of Graz.

References

[1] Gardener, M. (1970). Mathematical games: The fantastic combinations of John Conway's new solitaire game "life". *Scientific American*, 223(4): 120–123.

[2] Eppstein, D. (2010). Growth and decay in life-like cellular automata. In Adamatzky, A., editor, *Game of Life Cellular Automata*, pages 71-97, Springer, London.

[3] Bays, C. (1987). Candidates for the game of life in three dimensions. *Complex Systems*, 1(3): 373–400.

[4] Bays, C. (2006). A note about the discovery of many new rules for the game of three-dimensional life. *Complex Systems*, 16(4): 381.

[5] Czirók, A. and Vicsek, T. (2000). Collective behavior of interacting self-propelled particles. *Physica A: Statistical Mechanics and its Applications*, 281(1-4), 17–29.

[6] Schmickl, T., Stefanec, M. and Crailsheim, K. (2016). How a life-like system emerges from a simplistic particle motion law. *Scientific Reports*, 6, 37969. DOI: 10.1038/srep37969

[7] Stefanec, M. and Schmickl, T. (2022). A 3D Variant of the Primordial Particle System. *Zenodo*. https://doi.org/10.5281/zenodo.6557999

[8] Volterra, V. (1926). Variations and fluctuations of the number of individuals in animal species living together. *Animal Ecology*, 409-448.

[9] Lotka, A. J. (1925). Elements of physical biology. *Williams & Wilkins*.

[10] Sayama, H. (2009). Swarm chemistry. *Artificial Life*, 15(1): 105–114.

[11] Sayama, H. (2011). Swarm Chemistry Evolving. In Fellermann, H., Dörr, M., Hanczyc, M.M., Laursen, L.L., Maurer, S.E., Merkle, D., Monnard, P.-A., Stoy, K., and Rasmussen, S. editors, *Artificial Life XII: Proceedings of the Twelfth International Conference on the Synthesis and Simulation of Living Systems*, pages 32–36, MIT Press, Cambridge, MA.

[12] Sayama, H. (2018). Seeking Open-Ended Evolution in Swarm Chemistry II: Analyzing Long-Term Dynamics via Automated Object Harvesting. In *Proceedings of ALIFE 2018: The 2018 Conference on Artificial Life*, pages 59-66, MIT Press, Cambridge, MA.

[13] Chan, B.W.C. (2020). Lenia and Expanded Universe. In *Proceedings of ALIFE 2020: The 2020 Conference on Artificial Life*, pages. 221-229, MIT Press, Cambridge, MA.

[14] Sayama, H. (2012). Morphologies of self-organizing swarms in 3d swarm chemistry. In *Proceedings of the 14th annual conference on Genetic and evolutionary computation,* pages. 577-584, Association for Computing Machinery, New York, NY.

Firefly-inspired vocabulary generator for communication in multi-agent systems

Chantal Nguyen[1], Isabella Huang[2] and Orit Peleg[1,2,3*]

[1] BioFrontiers Institute, University of Colorado, Boulder, Boulder, CO, USA
[2] Department of Computer Science, University of Colorado, Boulder, Boulder, CO, USA
[3] Santa Fe Institute, Santa Fe, NM, USA
*orit.peleg@colorado.edu

Abstract

Fireflies' dazzling light displays are courtship rituals: flying males announce their presence as suitable mates to the females on the ground. Their light signal is composed of a species-specific on/off light sequence repeated periodically. However, thousands of fireflies flashing in a swarm can create immense visual clutter that hinders the detection of potential mates. A partial solution to this visual clutter problem is to flash according to sequences that are more distinct and detectable than those of other individuals. Here, we investigate how distinguishable flash sequences can co-evolve by developing a method for simulating sequences that minimize their mutual similarity with each other while minimizing their energetic cost and predation risk. This simple set of rules produces flash sequences that are remarkably similar to those of real fireflies. In particular, we observe an emergent periodicity in the resulting sequences, despite the lack of any periodicity requirements on the sequences. In addition, we demonstrate a method of reconstructing the evolutionary pressures acting on sets of firefly species. We do so by carrying out simulations that follow known phylogenetic relationships of extant species alongside their characteristic flash patterns.

Introduction

The multitude of flashes that punctuate a summer's night are mating calls from fireflies (Lampyridae): a chorus of airborne males announces their presence as suitable mates to females on the ground (Fig. 1A). Fireflies emit light from an abdominal "lantern" organ capable of producing bioluminescence. Some species emit steady glows, while others emit patterns of discrete flashes (Stanger-Hall et al., 2007; Lewis and Cratsley, 2008). Here, we focus on the latter, as these have the potential to temporally encode information and can further provide insight into the development of communications for multi-agent systems (Lewis and Cratsley, 2008).

The flying males flash according to a species-specific pattern. Each instance of the pattern is followed by a longer 'quiet' or 'dark' period wherein the females can respond with a species-specific delay (Lewis and Cratsley, 2008; Stanger-Hall and Lloyd, 2014). If a pair of fireflies recognizes their flashes as indicative of the same species, they

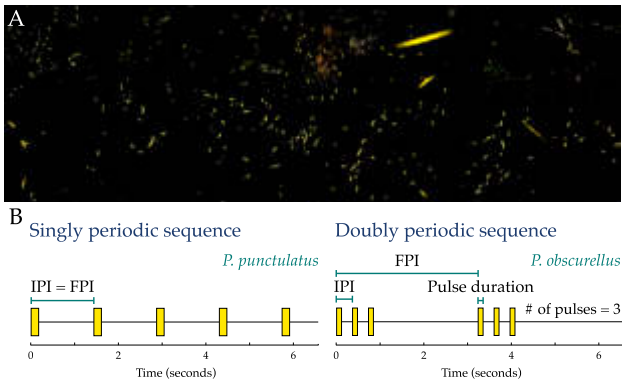


Figure 1: Firefly swarms flash to attract mates. A: Long exposure of a swarm of *Photinus carolinus* fireflies at the Great Smoky Mountains National Park, Tennessee, USA. B: Firefly flash patterns are typically either singly or doubly periodic, as illustrated with the characteristic patterns of *Photinus punctulatus* and *Photinus obscurellus*, respectively. Flash patterns can be parameterized by four values: the pulse duration, the number of pulses in a flash burst, the IPI (interpulse interval, or short period) and the FPI (flash-pattern interval, or long period). For singly periodic sequences, the IPI is necessarily equal to the FPI.

can continue to dialogue until they locate each other. On some occasions, fireflies misidentify a signal from a firefly of another species as being from their own. Female fireflies do mistakenly respond to signals from males of the wrong species, while males also locate and approach females of other species, leading to the male realizing the mistake and abandoning the interaction (Lloyd, 1968, 1969; Stanger-Hall and Lloyd, 2014). Moreover, female *Photuris* fireflies, dubbed "femme fatales", are able to mimic (though not necessarily perfectly) the flash responses of various species of *Photinus* males, luring the male into a deceptive dialogue that results in its attack by the female (Lloyd, 1957; Stanger-Hall and Lloyd, 2014).

While a female's flash response is relatively simple and

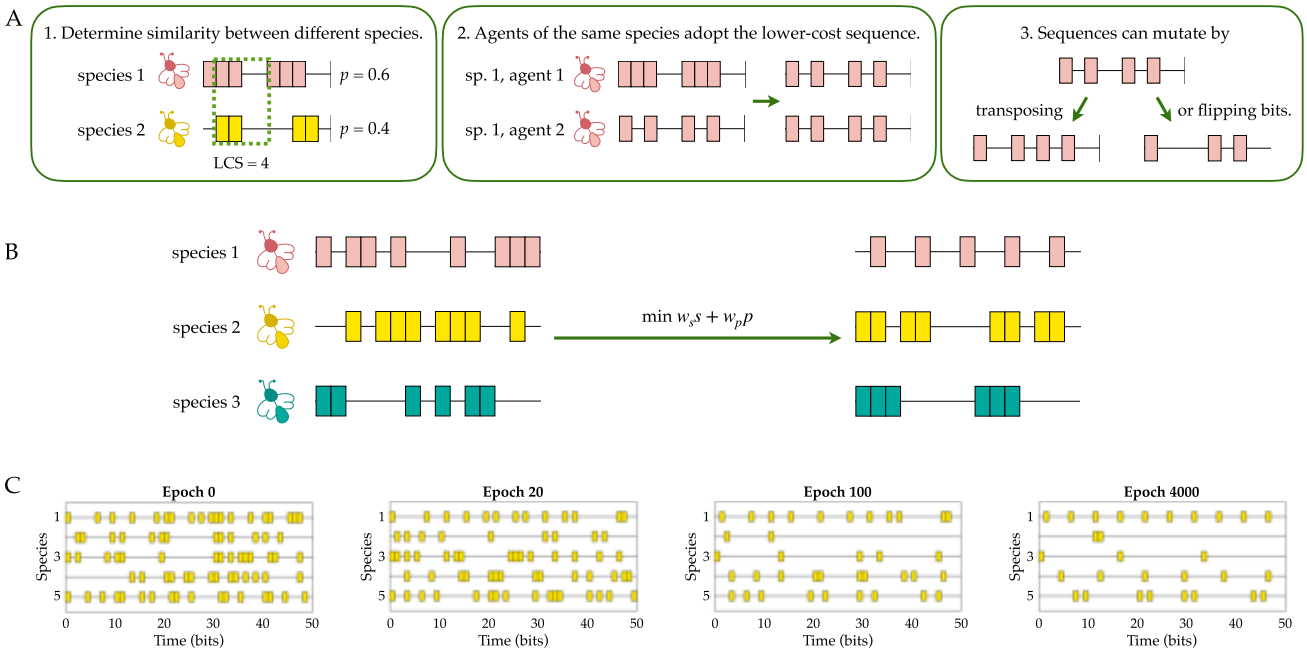


Figure 2: In the vocabulary generator, firefly agents simultaneously minimize a cost function in order to arrive at species-specific flash sequences. A: The similarity between sequences of different species is determined by calculating the longest common substring (LCS). In this example, the LCS is 4 bits. The predation score p, given by the proportion of *on* bits in the sequence, is also shown. Agents of the same species compare sequences, and the lower-cost sequence is adopted by both. Sequences can also mutate by flipping or transposing bits. B: Starting from random initial conditions, multiple species minimize the cost function to achieve distinguishable sequences. C: Evolution of a set of firefly sequences over the course of multiple epochs of the vocabulary generator. Starting from random initial conditions in epoch 0, the five species' sequences become increasingly periodic by epoch 4000.

often consists of a single flash (Lewis and Cratsley, 2008; Stanger-Hall and Lloyd, 2014), male flash sequences can be more complex. Among North American firefly species, male sequences have been observed to be either singly or doubly periodic (Fig. 1B) (Stanger-Hall and Lloyd, 2014). Singly periodic sequences consist of single flashes (of equal duration) spaced apart by regular quiet intervals. Doubly periodic sequences consist of flash "bursts" of several pulses in a row, separated by longer quiet periods. A flash sequence can be parameterized by 4 values (Fig. 1B): the pulse duration, the interpulse interval (IPI), the flash-pattern interval (FPI), and the number of pulses per pattern. The IPI represents the short period of a sequence, consisting of a pulse and the subsequent pause before the next pulse in a burst. The FPI represents the long period, consisting of the entire flash burst and the succeeding quiet period. If the sequence is singly periodic, the IPI and FPI are equal. As male sequences have been more extensively documented for a range of species than female responses, we focus here on only male signals.

Thousands of fireflies flashing in a swarm presents the possibility of immense visual clutter. This results in a "cocktail party problem": fireflies must detect the correct pattern of their potential mates while filtering out flashes of other species, similar to how a partygoer can focus on a single conversation while filtering out background music and irrelevant chatter (Bee and Micheyl, 2008). In some species, fireflies synchronize their flashes with swarm mates (Buck, 1988), a social phenomenon that can alleviate visual clutter (Moiseff and Copeland, 2010). The mechanisms, driving factors, and consequences of synchronization in fireflies remain a widely investigated topic (Moiseff and Copeland, 1994, 2020; Sarfati et al., 2021) and source of inspiration for the design of decentralized robotic systems (Christensen et al., 2009; Perez Diaz, 2016).

In more visually complex scenarios, tens of different firefly species can occupy the same geographical area (Lloyd, 1969; Stanger-Hall and Lloyd, 2014). Even if individual fireflies synchronize, it would be beneficial for different species to have distinct flash patterns to facilitate species recognition. Indeed, character displacement has been observed in North American firefly species, where species with overlapping geographical distribution (sympatric) exhibit greater differences in their flash patterns than (allopatric) species that do not (Stanger-Hall and Lloyd, 2014). Sym-

patric species can differ in their mating times and breeding habitats as well (Lloyd, 1966; Lewis et al., 2004). It has also been observed that fireflies decode species information primarily from the temporal characteristics of flash patterns, namely the pulse duration, number of pulses, IPI, and/or FPI, rather than the color of the light or the movement patterns of the flashing firefly (Lloyd, 1966; Lewis et al., 2004). Hence, we focus solely on temporal information in flash patterns in this paper.

The more conspicuous a flash pattern is, the more noticeable the firefly is to potential mates, but also nearby predators. Bats and spiders, for example, may be attracted to the firefly's flash (Lloyd, 1973). Predation pressure has resulted in some species foregoing their flash altogether, and instead relying on less efficient pheromone signaling to evade danger (Stanger-Hall et al., 2007; Stanger-Hall and Lloyd, 2014). There may also be energy costs, albeit minimal, associated with flashing (Lewis and Cratsley, 2008), and pressure to minimize energy expenditure would likewise diminish the amount of flashing. As such, we treat both energy cost and predation risk as the same driving force, and will henceforth refer to this cost mechanism as predation risk. Hence, we investigate how firefly flash patterns can co-evolve to be distinguishable under a potentially competing pressure to reduce predation risk.

In this paper, we develop a method for simulating distinguishable firefly-like signals according to an evolutionary process that we term the "vocabulary generator". In the vocabulary generator, firefly agents possess binary flash sequences; the sequences mutate at a specified rate, and agents adopt others' sequences in order to minimize a cost function that penalizes both high amounts of flashing (a predation risk), and similarity between sequences of different species. We demonstrate that even when sequences are initialized to random strings of bits that can only change in a bitwise manner, the sequences nevertheless self-organize into near-periodic patterns that resemble real firefly flash sequences. We observe that the competing pressures to minimize predation risk and confusion between species presents a tradeoff, wherein prioritizing discriminability between sequences results in both a higher amount of flashing as well as more complex sequences that contain flash bursts instead of singly periodic flashes. Finally, we explore how the vocabulary generator can be used to "reverse-engineer" a cost function that can capture the relative strength of evolutionary pressures which shaped the flash patterns observed in extant species.

Vocabulary generator

We introduce the "vocabulary generator", an evolutionary method for simulating firefly-like signals (Fig. 2). This method is inspired by the naming game procedure, wherein agents achieve common vocabularies via pairwise interactions (Steels, 1997; Baronchelli et al., 2008). In our procedure, multiple firefly species simultaneously minimize a cost function over a series of epochs, whereby fireflies of the same species iteratively compare their sequences via pairwise interactions to mutually adopt the lower-cost sequence. Unlike in the naming game, sequences in vocabulary generator can mutate, as in some evolutionary language models (Nowak and Krakauer, 1999; Nowak et al., 1999). The objective in the vocabulary generator is for fireflies of the same species to arrive at the same characteristic flash sequence, with that sequence being distinguishable from the sequences adopted by other species.

A flash sequence of length L is defined as a binary string of bits, each with value 1 (*on* or flashing) or 0 (*off* or not flashing):

$$(a_n)_{n=1}^{L}, \;\; a_n \in \{0, 1\}.$$

In defining the cost function, we focus on two aspects likely to shape the evolution of firefly communication: the distinguishability between species-specific patterns, and predation risk. We define the cost $\mathcal{C}$ of a particular sequence as follows:

$$\mathcal{C}((a_n)_{n=1}^{L}) = w_s s + w_p p, \tag{1}$$

where s denotes the average similarity between that sequence and sequences of all other species, and p the predation risk of the sequence. w_s and w_p are weights whose relative values can be adjusted to produce sequences with different characteristics.

We define the similarity between two sequences $(a_m)_{m=1}^{L}$ and $(a_n)_{n=1}^{L}$ as the length of the longest common substring under cyclic permutation, denoted $LCS((a_m)_{m=1}^{L}, (a_n)_{n=1}^{L})$, representing the maximal possible overlap between the sequences. We consider all cyclic permutations in comparing the sequences as in nature, there is no guarantee that two different co-habitant species start flashing their sequences at the same time or at a constant offset. The longest common substring between two sequences is given by

$$LCS((a_m)_{m=1}^{L}, (a_n)_{n=1}^{L}) = \max_{0 \le k \le L-1} f((a_m)_{m=1}^{L} - (a_n \to a_{n+k(\mathrm{mod}\ L)})_{n=1}^{L}), \tag{2}$$

where $a_n \to a_{n+k}$ indicates shifting the nth element of the sequence by k places, and $f(a_n)_{n=1}^{L}$ represents a function that computes the length of the longest consecutive subsequence of zeroes in the sequence $(a_n)_{n=1}^{L}$. For example, $f(1, 0, 0, 0, 1, 0, 1) = 3$. We note that the spatiotemporal resolution and signal processing mechanisms of real fireflies vary from species to species, and the ability of female fireflies to discriminate and respond to simulated male flashes of varying length or period has been explored for various North American species (Lloyd, 1966; Carlson and Copeland, 1985). These can be further quantified in future field experiments in order to inform the similarity computation. Here we ignore spatial considerations such as the

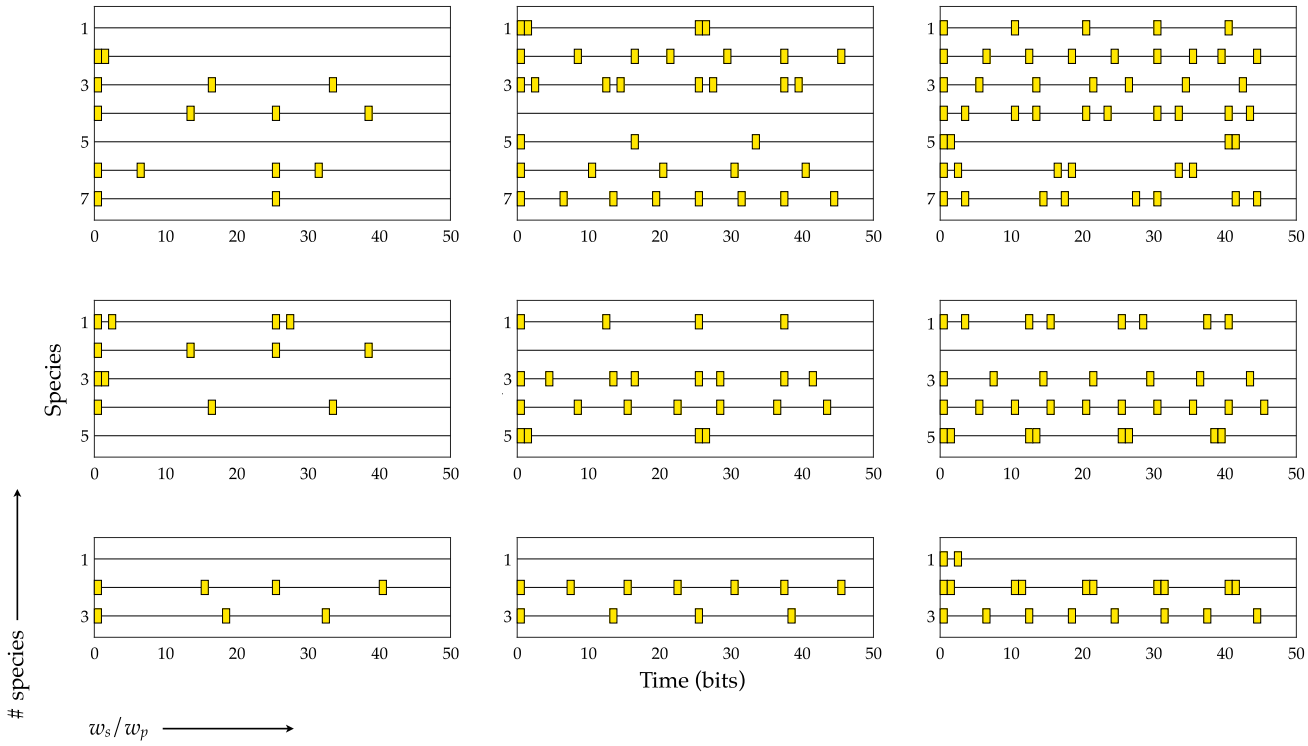


Figure 3: Example ensembles of flash sequences simulated for 3, 5, and 7 species, with weight ratio w_s/w_p values of 0.4 (left column), 1.4 (middle column), and 2.4 (right column). The weight ratio w_s/w_p represents the tradeoff between similarity and predation pressures, and higher values of this ratio produce sequences with a greater number of flashes.

movement of the firefly or the potential attenuation or obscuring of signals by foliage or other obstacles, and we assume that a single bit represents the highest temporal resolution achievable. To compute the similarity term s for a given sequence, the LCS between that sequence and all other sequences of different species is averaged and normalized by the length of the sequence, such that s is valued between 0 and 1.

Moreover, we define the predation risk p as the proportion of *on* bits over a given sequence length, i.e., $p = \sum_{n=1}^{L} a_n/L$. The more a firefly flashes, the higher the predation risk. Likewise, p is valued between 0 and 1. Ultimately, this cost function presents a tradeoff: minimizing predation risk will result in sparser flash sequences, but minimizing mutual similarity may require the presence of more flashes to distinguish between species.

In the vocabulary generator (Fig. 2), we define N_s different species of fireflies, with each species containing N_f individual firefly agents. In population genetics simulations, one way to shorten the time to convergence while keeping the mutation rate constant is to introduce the concept of populations for each species (Gillespie, 2004). Hence, we consider a population of N_f agents for each species rather than a single representative phenotype for that species. Each agent is assigned a binary sequence $(a_n^{\sigma,\phi})_{n=1}^{L}$, initialized to a random string of bits such that $a_n^{\sigma,\phi} \sim \mathrm{Ber}(0.5)$, where $\mathrm{Ber}(p)$ denotes the Bernoulli distribution with probability p, $\sigma \in \{1, \ldots, N_s\}$ denotes the species, and $\phi \in \{1, \ldots, N_f\}$ denotes the agent identity.

Then, the cost function for a sequence $(a_n^{\sigma,\phi})_{n=1}^{L}$ in a system with N_s species and N_f agents per species is given by:

$$\mathcal{C}\left((a_n^{\sigma,\phi})_{n=1}^{L}\right) = \frac{w_s}{(N_s-1)N_f} \sum_{\sigma'=1,\sigma'\neq\sigma}^{N_s} \sum_{\phi=1}^{N_f} LCS((a_n^{\sigma,\phi})_{n=1}^{L}, (a_n^{\sigma',\phi'})_{n=1}^{L}) + \frac{w_p}{L} \sum_{n=1}^{L} a_n^{\sigma,\phi}. \quad (3)$$

The vocabulary generator method (Procedure 1) is illustrated in Fig. 2A and proceeds as follows. The similarity between sequences of different species is determined by computing the LCS (Eq. 2), and the similarity score s for each sequence is determined by averaging over the normalized LCS between that sequence and those of all other species (Fig. 2A, Box 1). Then, for each pair of firefly agents belonging to the same species, their costs are computed using the previously determined similarity score s, following Eq. 3

Procedure 1 Vocabulary generator

Input: number of species N_s, number of agents N_f, length of sequence L, mutation rate μ, epochs to simulate N

for σ = 1 to N_s **do**
  for ϕ = 1 to N_f **do**
    for n = 1 to L **do**
      $a_n \leftarrow$ random integer $\in \{0, 1\}$
for epoch = 1 to N **do**
  for σ = 1 to N_s **do**
    for ϕ = 1 to N_f **do**
      for ϕ' = ϕ to N_f **do**
        $\mathcal{C}_{\sigma,\phi} \leftarrow \mathcal{C}((a_n^{\sigma,\phi})_{n=1}^{L})$
        $\mathcal{C}_{\sigma,\phi'} \leftarrow \mathcal{C}((a_n^{\sigma,\phi'})_{n=1}^{L})$
        if $\mathcal{C}_{\sigma,\phi} > \mathcal{C}_{\sigma,\phi'}$ **then**
          $(a_n^{\sigma,\phi})_{n=1}^{L} \leftarrow (a_n^{\sigma,\phi'})_{n=1}^{L}$
          if random number $\in (0, 1) < \mu$ **then**
            mutate$((a_n^{\sigma,\phi})_{n=1}^{L})$
        else if $\mathcal{C}_{\sigma,\phi} < \mathcal{C}_{\sigma,\phi'}$ **then**
          $(a_n^{\sigma,\phi'})_{n=1}^{L} \leftarrow (a_n^{\sigma,\phi})_{n=1}^{L}$
          if random number $\in (0, 1) < \mu$ **then**
            mutate$((a_n^{\sigma,\phi'})_{n=1}^{L})$
function MUTATE$((a_n)_{n=1}^{L})$
  $r \leftarrow$ random integer $\in \{1, \ldots, L\}$
  if random integer $\in \{0, 1\} = 0$ **then**
    $a_r \leftarrow !a_r$
  else
    $x \leftarrow a_r$
    $y \leftarrow a_{1+r(\mathrm{mod}L)}$
    $a_r \leftarrow y$
    $a_{1+r(\mathrm{mod}L)} \leftarrow x$

(Fig. 2A, Box 2). Then, the agent with the higher-cost sequence adopts the sequence of the other agent. The sequence can also mutate with a specified probability (the mutation rate) by flipping a bit (from *off* to *on* or *on* to *off*) or transposing two adjacent bits (Fig. 2A, Box C). The above steps are then repeated until the average costs of the sequences reach a steady state (Fig. 2B). Moreover, Fig. 2C illustrates the evolution of the sequences from a system of 5 species, initialized to random sequences in epoch 0, and reaching a steady state by epoch 4000.

Examples of sequences simulated with the vocabulary generator are illustrated in Fig. 3. The top row shows sequences generated for ensembles of $N_s = 7$ species with $N_f = 10$ agents; the resulting characteristic sequence for each species is shown. The middle row illustrates ensembles of $N_s = 5$ and the bottom row ensembles of $N_s = 3$, with $N_f = 10$ agents in both cases. The ratio of the cost function weights w_s/w_p increases from left to right, with sequences in the left column simulated using $w_s/w_p = 0.4$, $w_s/w_p = 1.4$ for the middle column, and $w_s/w_p = 2.4$ for the right column.

We observe that when the w_s/w_p ratio is small (Fig. 3, left column), sequences are extremely sparse, with more than one sequence being entirely devoid of flashes. As this ratio increases, the frequency of flashing increases, and the length of flashes may increase as well (Fig. 3, right column).

We emphasize that we do not enforce the sequences to be periodic; they are initialized to random strings of bits and mutated in a bit-by-bit manner (Fig. 2A). Nevertheless, we observe that the resulting sequences (Fig. 3) demonstrate an emergent periodicity. We also observe both singly-periodic and doubly-periodic sequences; moreover, we observe that the lengths of individual flashes in a sequence generally do not vary.

We repeatedly run the vocabulary generator to perform a more extensive parameter sweep over values of the w_s/w_p weight ratio that range from 0.2 to 3 in increments of 0.2, and over system sizes ranging between 2 and 7 species (Fig. 4). We quantify the periodicity in the resulting generated sequences by examining the variance in the spaces, or gaps, between flashes. A sequence is classified as singly periodic if the standard deviation of the gap size is less than 0.2 times the mean gap size of that sequence, and doubly periodic if it is greater. Fig. 4A shows the standard deviation in gap sizes as a function of the number of species and the ratio of the similarity and predation weights. For singly periodic sequences, simply the standard deviation of gap sizes, normalized by the mean gap size, is shown. For doubly periodic sequences, the gap sizes are partitioned into two clusters using k-means clustering, and the standard deviation for each cluster, normalized to the mean of that cluster, is computed; these two values are then averaged and shown in Fig. 4A. We observe that the higher the w_s/w_p ratio, the higher the variation in the gap size, and thus the lower the periodicity.

The cost function (Eq. 1) captures a tradeoff between predation risk and similarity. When the similarity term is more strongly weighted, i.e., when w_s is larger relative to w_p, we observe that sequences include more flashes in order to differentiate themselves, but also become more complex, containing more sequences with flash bursts. We also observe that for a small number of species, most simulated sequences are singly periodic. For higher numbers of species, along with increased values of the w_s/w_p ratio, the higher the prevalence of doubly periodic sequences (Fig. 4B). Increasing the number of flashes in a burst can help to reduce the similarity with other sequences when more species are present.

Brute force validation

To validate our observation of emergent periodicity, we take a brute-force approach by exploring the set of *all* possible binary sequences for a given length. Our objective here is to determine whether sequences that are periodic like firefly flash patterns result in a lower cost (Eq. 1) than those that are not.

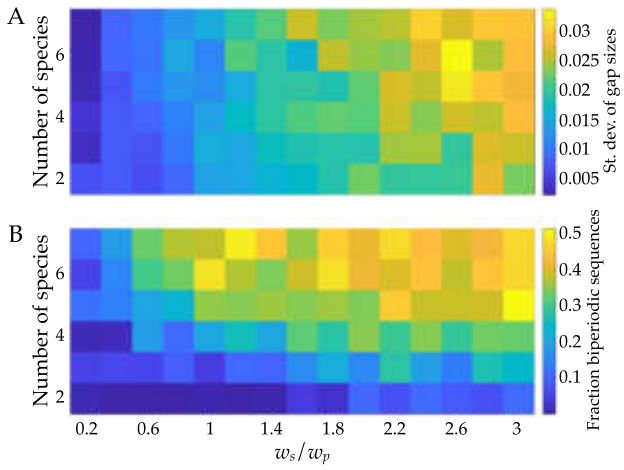


Figure 4: Periodicity in simulated flash sequences. A: Standard deviation in the size of gaps between flashes as a function of the number of species and the ratio of weights w_s/w_p. The lower the standard deviation in gap size, the higher the periodicity. B: The fraction of doubly periodic sequences present in an ensemble of simulated sequences as a function of the number of species and the ratio of weights w_s/w_p.

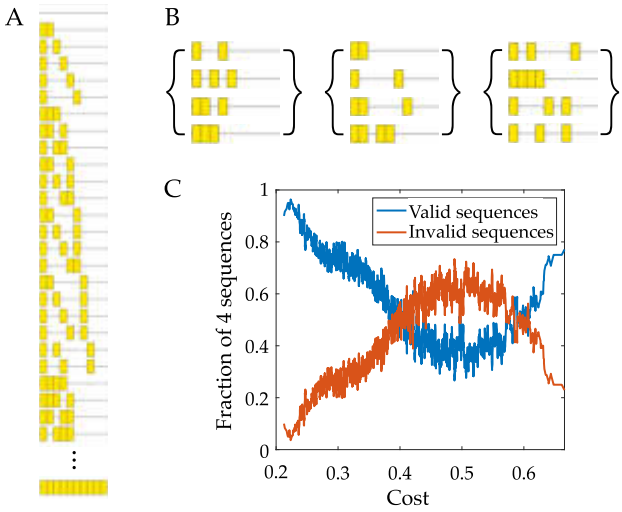


Figure 5: Brute force validation of periodicity. A: 109 10-bit long sequences can be generated which are unique under cyclical permutation; 44 of these sequences are valid firefly sequences, while the rest are invalid as they are neither singly nor doubly periodic. B. Combinations of four sequences are sampled from the set of unique sequences, and the average cost of each combination is computed (Eq. 3, with $N_s = 4$, $N_f = 1$, $L = 10$, $w_s = 1$, and $w_p = 1$). C. The fractions of valid and invalid sequences in each combination is plotted against the cost of that combination. The blue line (fraction of valid sequences) and orange line (fraction of invalid sequences) sum to a constant value of 1.

First, we generate all sequences of a given length that are unique under cyclic permutation. For example, $(1,0,1,0,0,0)$ is equivalent to $(0,1,0,1,0,0)$, $(0,0,1,0,1,0)$, $(0,0,0,1,0,1)$, etc.; and as such only one of these permutations would need to be included. Then, we determine how many of these sequences are valid firefly sequences, i.e., singly or doubly periodic with flashes of equal lengths. For example, $(1,0,0,1,0,0)$ is a valid firefly sequence, while $(1,1,1,0,1,0)$ is not a valid firefly sequence.

As a specific example, we focus on sequences that are 10 bits long: there are 109 unique binary sequences, and 44 of these are valid firefly sequences (Fig. 5A). Then, from this set of unique sequences, we can generate every combination of 4 sequences representing communities of 4 species, which mimic the ensembles of sequences generated with the vocabulary generator (Fig. 5B). For each of these combinations, we compute the average cost with Eq. 3, with $N_s = 4$, $N_f = 1$, $L = 10$, $w_s = 1$, and $w_p = 1$.

We observe that in the combinations with the lowest average cost, a dominant fraction of these are valid sequences (Fig. 5C). However, for combinations with increasing cost, the proportion of invalid sequences increases as well. We note that for very high costs, the majority of sequences in the combinations are valid sequences, but these contain extremely long flashes and are thus energetically costly.

Reverse-engineering cost functions

Lastly, we demonstrate a way in which the weights w_s and w_p in Eq. 1 can be estimated in order to shed light on how real firefly species may have evolved under selection to minimize similarity and predation risk. Our objective is to use the vocabulary generator to simulate ensembles of sequences for various values of the weight ratio w_s/w_p, and compare these simulated sequences with known firefly flash patterns to find the weight ratio which results in simulated sequences that most resemble the known patterns. In doing so, this can shed light on the relative importance of the similarity and predation pressures in shaping real firefly signals.

As an example, we consider six species from the Consangineus clade of *Photinus* (Fig. 6A) (Stanger-Hall and Lloyd, 2014). We use the sequences of *P. macdermotti* and *P. ignitus*, which were among the earliest to speciate, as starting "seed sequences" for the vocabulary generator. That is, we initialize a system of $N_s = 3$ species and N_f agents. The sequences $(a_n^{1,\phi})_{n=1}^L$ for all $\phi = 1,\ldots,N_f$ are initialized to all be equivalent to the known sequence for *P. macdermotti*, and likewise $(a_n^{2,\phi})_{n=1}^L$ for *P. ignitus*. The sequences $(a_n^{3,\phi})_{n=1}^L$ are also all initialized to be that of *P. ignitus*, the most recently speciated species of the two. Then, the vocab-

Procedure 2 Vocabulary generator based on known firefly phylogeny trees

Input: Initial seed sequence(s) $(a_n^{\sigma})_{i=1}^{L}$, number of initial sequences N_s, number of agents N_f, length of sequence L, mutation rate μ, epochs to simulate N, total number of sequences to simulate N_{tot}

for iteration = $N_s + 1$ to N_{tot} **do**
  $N_s \leftarrow N_s + 1$
  for ϕ = 1 to N_f **do**
    $(a_n^{\sigma=N_s,\phi})_{n=1}^{L} \leftarrow (a_n^{\sigma=N_s-1,\phi})_{n=1}^{L}$
  run Procedure 1 where only $(a_n^{\sigma=N_s,\phi})_{n=1}^{L}$ are mutable
  for ϕ = 1 to N_f **do**
    $(a_n^{N_s,\phi})_{n=1}^{L} \leftarrow \text{mode}_{\phi=1,\ldots,N_f}((a_n^{N_s,\phi})_{n=1}^{L})$

ulary generator (Procedure 1) is run for a specified number of epochs, where only the sequences $(a_n^{3,\phi})_{n=1}^{L}$ are allowed to change; the seed sequences $(a_n^{\{1,2\},\phi})_{n=1}^{L}$ are kept fixed and only factor into the similarity computations (Eq. 2) of the cost function (Eq. 3). By the last epoch, the sequences of species $\sigma = N_s = 3$ should have all converged to the same sequence, but are assigned to the mode of all $(a_n^{3,\phi})_{n=1}^{L}$ sequences to ensure that they are identical. Then, a simulated speciation occurs: the number of species in the system is increased by 1, so that $N_s = 4$. The new species' sequences are initialized to those most recently obtained from the vocabulary generator; that is, $(a_n^{N_s,\phi})_{n=1}^{L} := (a_n^{N_s-1,\phi})_{n=1}^{L}$, and the sequences $(a_n^{N_s-1,\phi})_{n=1}^{L}$ are now kept fixed. The vocabulary generator is then run again, the number of species incremented by 1, and the new sequences initialized to the recent outputs, which are now kept fixed. This is repeated until the desired total number of sequences are obtained, specifically six in our example. The full procedure for simulating sequences based on known phylogeny trees is described in Procedure 2.

We repeatedly carry out the above procedure (Procedure 2), sweeping over the weight ratio w_s/w_p in the cost function (Eq. 1) between a range of 0.3 and 1.3 in increments of 0.04. Then, we compare each set of simulated sequences and with the known firefly flash patterns. For each of the known flash patterns, we extract four parameters that describe the sequences: the number of pulses, pulse duration, IPI, and FPI (Fig. 1). We also estimate these parameters for each of the simulated sequences, and compute the average root mean square error between the known and simulated parameters (Fig. 6B). We observe that a weight ratio w_s/w_p between 0.4 and 0.5 produces flash sequences that are most similar to the known firefly patterns (Fig. 6C).

Conclusion and Future Work

In this work, we explore how firefly sequences may have coevolved among sympatric species to increase discriminability in order to alleviate the "cocktail party problem" faced by swarms during their flash-mediated mating rituals. We develop a method, termed the vocabulary generator, to simulate the co-evolution of firefly sequences under pressure to minimize both similarity with other species' signals, and individual predation risk. The resulting simulated sequences are periodic or close to periodic, despite the lack of any constraints pertaining to periodicity. We also observe that combinations of flash sequences that contain periodic sequences can result in a lower average cost than those that contain aperiodic sequences.

We also demonstrate a method in which the vocabulary generator can be used to gauge the relative importance of the selection pressures in the cost function (Eq. 1) in shaping real firefly flash patterns. While we posit here that predation risk and distinguishability play the largest roles in shaping firefly flash patterns, it is possible that other factors may affect communication signals, such as the nature of the habitat and, more significantly, female preference (Lewis and Cratsley, 2008; Stanger-Hall and Lloyd, 2014). Intraspecies variation in male flash patterns has been observed in some species, along with female preference for longer flashes, shorter flashes, or a higher flash rate, depending on the species (Lewis and Cratsley, 2008). Incorporating female responses and the aforementioned factors into the vo-

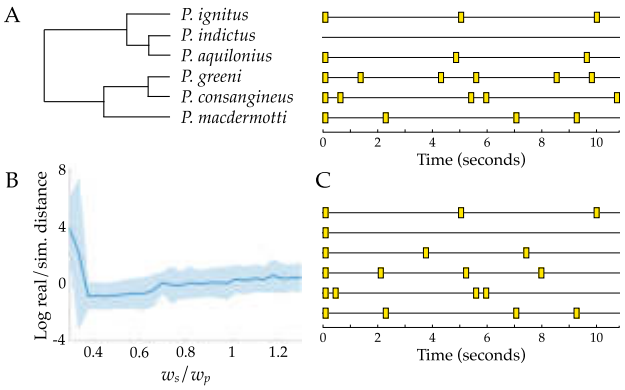


Figure 6: Reverse-engineering the cost function by simulating sequences along the Consangineus clade. A: The phylogenetic relationships of six firefly species and their respective species-specific flash patterns (Stanger-Hall and Lloyd, 2014). B: The logarithm of the average distance between known and simulated flash sequences, as measured by the average root mean squared error in the four flash parameters (number of pulses, pulse duration, IPI, FPI), as a function of the weight ratio w_s/w_p. Shaded region indicates one standard deviation. C: Examples of simulated flash sequences, obtained following Procedure 2 using w_s/w_p = 0.46. Note that the first and last sequence are the two "seed" sequences and are identical to the known sequences for *P. ignitus* and *P. macdermotti*, respectively.

cabulary generator may shed further insight into firefly signal evolution.

Many other animals, including various species of birds, frogs, and other insects, encode species information in their signals, whether visual or acoustic (Ravignani et al., 2014; Garcia et al., 2020; Höbel and Gerhardt, 2003; Amezquita et al., 2011; Ryan and Rand, 1993). For example, Garcia et al. explored the evolution of species-specific woodpecker drumming patterns and quantified how the mutual information content of the signals changed with species radiation, thereby determining the strength of selection for signal diversity (Garcia et al., 2020). Likewise, in future work, a more extensive analysis of firefly flash patterns could be performed in order to similarly quantify selection pressures on firefly signals, including signal diversity but also predation and other factors, by harnessing the known phylogenetic relationships and recorded signals of numerous North American species and more worldwide (Stanger-Hall et al., 2007; Stanger-Hall and Lloyd, 2014).

In developing the vocabulary generator, we discretize flash sequences into series of bits, where we define a single bit to represent the finest temporal resolution achievable by a firefly's internal signal processing capabilities. By defining similarity as the longest common substring, we also assume that the species information is encoded as a consecutive sequence of bits. However, fireflies may determine species information by measuring properties relating to the timing of the flash periods or the number of flashes. Moreover, fireflies may also encode species information in non-temporal properties of their signal, such as their movement patterns during flashing or the wavelength of the light, although these properties appear to be less important than temporal characteristics in species identification (Lloyd, 1966; Lewis et al., 2004). Further field experiments can be performed to explicitly quantify the spatial and temporal resolution of a firefly's signal processing capabilities.

Fireflies, and in particular their synchronization, are increasingly probed as a source of inspiration for swarm robotics (Christensen et al., 2009; Perez Diaz, 2016). Moreover, visible light communication has been explored as a cost-effective method for local, decentralized communication between agents, and firefly signals have also been credited as a source of inspiration in the design of such systems (Maxseiner et al., 2021; Murai et al., 2012; Ito et al., 2018). For instance, an individual robot can be equipped with an LED that can flash according to a pattern similar to that of a firefly's. We propose that the vocabulary generator method can be adapted to generate different sequences for robotic agents. We note that in the cost function (Eq. 1), predation risk is interchangeable with energetic cost, as the corresponding term serves to minimize the total amount of flashing. For example, for a swarm of robots with differentiated tasks, each group can be programmed to communicate according to its own sequence, optimized to be maximally dissimilar from the others and individually energetically efficient. If the robots must differentiate their tasks in real time, the vocabulary generator could be used to generate easily discriminated sequences rather than using ones that are preprogrammed.

Fireflies have long been a source of wonder and inspiration, but their populations are increasingly threatened by deforestation, urbanization, pesticide use, and climate change (Lewis et al., 2020). Recent research has shown that light pollution can interfere with flash signaling, lowering mating success (Lewis et al., 2020; Firebaugh and Haynes, 2016). Understanding the mechanisms behind firefly communication will be invaluable to worldwide conservation efforts.

Acknowledgements

O.P. acknowledges internal funds from the BioFrontiers Institute.

References

Amezquita, A., Flechas, S. V., Lima, A. P., Gasser, H., and Hodl, W. (2011). Acoustic interference and recognition space within a complex assemblage of dendrobatid frogs. *Proc. Natl. Acad. Sci. USA*, 108(41):17058–17063.

Baronchelli, A., Loreto, V., and Steels, L. (2008). In-depth analysis of the Naming Game dynamics: the homogeneous mixing case. *Int. J. Mod. Phys. C*, 19(05).

Bee, M. A. and Micheyl, C. (2008). The "cocktail party problem": What is it? How can it be solved? And why should animal behaviorists study it? *J. Comp. Psychol.*, 122:235–251.

Buck, J. (1988). Synchronous rhythmic flashing of fireflies. II. *Q. Rev. Biol.*, 63:265–289.

Carlson, A. D. and Copeland, J. (1985). Flash communication in fireflies. *Q. Rev. Biol.*, 60(4):415–436.

Christensen, A., O'Grady, R., and Dorigo, M. (2009). From fireflies to fault-tolerant swarms of robots. *IEEE Trans. Evol. Computat.*, 13(4):754–766.

Firebaugh, A. and Haynes, K. J. (2016). Experimental tests of light-pollution impacts on nocturnal insect courtship and dispersal. *Oecologia*, 182:1203–1211.

Garcia, M., Theunissen, F., Sèbe, F., Clavel, J., Ravignani, A., Marin-Cudraz, T., Fuchs, J., and Mathevon, N. (2020). Evolution of communication signals and information during species radiation. *Nat. Commun.*, 11(1):4970.

Gillespie, J. H. (2004). *Population Genetics: A Concise Guide (2nd ed.)*. Johns Hopkins University Press, Baltimore, MD.

Höbel, G. and Gerhardt, H. C. (2003). Reproductive character displacement in the acoustic communication system of green tree frogs (Hyla cinerea). *Evolution*, 57(4):894–904.

Ito, T., Tahara, J., Koike, M., and Zhang, F. (2018). Development of the visible light communication device for swarm using nonlinear synchronization. *Artif. Life Robotics*, 23(1):60–66.

Lewis, S. M. and Cratsley, C. K. (2008). Flash signal evolution, mate choice, and predation in fireflies. *Annu. Rev. Entomol.*, 53:293–321.

Lewis, S. M., Cratsley, C. K., and Demary, K. (2004). Mate recognition and choice in Photinus fireflies. *Ann. Zool. Fennici*, 41(6,):809–821.

Lewis, S. M., Wong, C. H., Owens, A. C. S., Fallon, C., Jepsen, S., Thancharoen, A., Wu, C., De Cock, R., Novák, M., López-Palafox, T., Khoo, V., and Reed, J. M. (2020). A global perspective on firefly extinction threats. *BioScience*, 70:157–167.

Lloyd, J. E. (1957). Aggressive mimicry in Photuris fireflies: Signal repertoires by femmes fatales. *Science*, 187(4175):452–453.

Lloyd, J. E. (1966). Studies on the flash communication system in Photinus fireflies. *Univ. Michigan Mus. Zool. Misc. Pub.*, 130:1–96.

Lloyd, J. E. (1968). A new Photinus firefly, with notes on mating behavior and a possible case of character displacement (Coleoptera: Lampyridae). *Coleopt. Bull.*, 22(1):1–10.

Lloyd, J. E. (1969). Flashes, behavior and additional species of Nearctic Photinus fireflies (Coleoptera: Lampyridae). *Coleopt. Bull.*, 23(2):29–40.

Lloyd, J. E. (1973). Firefly parasites and predators. *Coleopt. Bull.*, 27(2):91–106.

Maxseiner, A. B., Lofaro, D. M., and Sofge, D. A. (2021). Visible light communications with inherent agent localization and simultaneous message receiving capabilities for robotic swarms. In *2021 18th International Conference on Ubiquitous Robots (UR)*, pages 633–639, Gangneung, Korea (South). IEEE.

Moiseff, A. and Copeland, J. (1994). Mechanisms of synchrony in the North American firefly Photinus carolinus (Coleoptera: Lampyridae). *J. Insect Behav.*, 8(3):395–407.

Moiseff, A. and Copeland, J. (2010). Firefly synchrony: A behavioral strategy to minimize visual clutter. *Science*, 329(5988):181–181.

Moiseff, A. and Copeland, J. (2020). Behavioral consequences of sensory system constraints in the firefly Photinus carolinus. *Ecol. Psychol.*, 32(4):143–152.

Murai, R., Sakai, T., Kawano, H., Matsukawa, Y., Kitano, Y., Honda, Y., and Campbell, K. C. (2012). A novel visible light communication system for enhanced control of autonomous delivery robots in a hospital. In *2012 IEEE/SICE International Symposium on System Integration (SII)*, pages 510–516, Fukuoka, Japan. IEEE.

Nowak, M. A. and Krakauer, D. C. (1999). The evolution of language. *Proc. Natl. Acad. Sci. USA*, page 6.

Nowak, M. A., Plotkin, J. B., and Krakauer, D. C. (1999). The Evolutionary Language Game. *J. Theor. Biol.*, 200(2):147–162.

Perez Diaz, F. (2016). *Firefly-Inspired Synchronization in Swarms of Mobile Agents*. PhD thesis, University of Sheffield.

Ravignani, A., Bowling, D. L., and Fitch, W. T. (2014). Chorusing, synchrony, and the evolutionary functions of rhythm. *Front. Psychol.*, 5.

Ryan, M. J. and Rand, A. S. (1993). Species recognition and sexual selection as a unitary problem in animal communication. *Evolution*, 47(2):647–657.

Sarfati, R., Hayes, J. C., and Peleg, O. (2021). Self-organization in natural swarms of Photinus carolinus synchronous fireflies. *Sci. Adv.*, 7:eabg9259.

Stanger-Hall, K. and Lloyd, J. (2014). Flash signal evolution in photinus fireflies: Character displacement and signal exploitation in a visual communication system. *Evolution*, 69:666–682.

Stanger-Hall, K. F., Lloyd, J. E., and Hillis, D. M. (2007). Phylogeny of North American fireflies (Coleoptera: Lampyridae): Implications for the evolution of light signals. *Mol. Phylogenet. Evol.*, 45(1):33–49.

Steels, L. (1997). The synthetic modeling of language origins. *Evoluti. Commun.*, 1:1–34.

Reliably Re-Acting to Partner's Actions with the Social Intrinsic Motivation of Transfer Empowerment

Tessa van der Heiden[1], Herke van Hoof[2], Efstratios Gavves[2] and Christoph Salge[3]

[1]BMW Group
[2]University of Amsterdam
[3] University of Hertfordshire
tessavdheiden@gmail.com

Abstract

We consider multi-agent reinforcement learning (MARL) for cooperative communication and coordination tasks. MARL agents can be brittle because they can overfit their training partners' policies. This overfitting can produce agents that adopt policies that act under the expectation that other agents will act in a certain way rather than react to their actions. Our objective is to bias the learning process towards finding reactive strategies towards other agents' behaviors. Our method, transfer empowerment, measures the potential influence between agents' actions. Results from three simulated cooperation scenarios support our hypothesis that transfer empowerment improves MARL performance. We discuss how transfer empowerment could be a useful principle to guide multi-agent coordination by ensuring reactiveness to one's partner.

Introduction

In this paper we investigate if and how social intrinsic motivation can improve Multi-agent reinforcement learning (MARL). MARL holds considerable promise to help address a variety of cooperative multi-agent problems - both for problem solving and simulation of multi-agent systems. However, one problem with MARL is that agents develop strong policies that are overfitted to their partners' behaviors. Specifically, with centralised training, agents can adopt strategies that expect other agents to act in a certain way rather than reacting to their actions. Such systems are undesirable as they may fail when their partners alter their strategies or have to collaborate with novel partners, either during the learning or deployment phase. Our aim is to avoid this specific lack of robustness and find a guiding principle that makes agents stay reactive to other agents' policy changes.

We want to introduce an additional reward to bias learning towards socially reactive strategies which should fulfil the following constraints: 1) it should, with minimal adaptation, apply to a wide range of problems with different sensor-actuator configurations to preserve the universality of the RL framework, and 2) it should not negatively affect the performance, i.e., once good policies are found, it should not harm exploitation. Fulfilling the above criteria would provide a general-purpose multi-agent learning algorithm for various cooperative tasks - as well as provide insights into general principles that would enable and improve the development of various forms of social interaction.

To address this challenge, we turn towards the idea of using Intrinsic motivation (IM) - a school of computational models (Oudeyer and Kaplan, 2009) that try to capture the essential motivations behind the behavior of (biological) agents - and then use them for behavior generation to obtain plausible and beneficial behavior. The core idea here is to ask if the principles that create single agent behavior can also be used to enhance multi-agent behavior. In this paper specifically, we look at Empowerment, an IM that captures how much an agent is able to affect the world it can itself perceive. Its information-theoretic formulation as the channel capacity between an agent's actions and its own sensors makes it a versatile measure that can be applied to a wide range of models where agent's are defined - satisfying constraint 1. Existing work on coupled empowerment maximisation (Salge and Polani, 2017; Guckelsberger et al., 2018) extends the formalism to a multi-agent setting. In this paper, we focus specifically on the idea of Transfer Empowerment (TE), introduced in those papers, which tries to capture how much one agent can potentially influence the actions of another. We use a slightly modified version of TE, which considers the channel capacity from one agent's actions to another agent's *actions*, rather than to their sensors, as in traditional TE.

Keeping the TE high between two agents means they are in a state were one of them is reliably reacting to the other. Adding this as an additional reward mechanism during training should help to avoid the brittleness of over-fitting we outlined before. We provide here quantitative evidence for our hypothesis that adding transfer empowerment as an additional reward increases upon the performance of state-of-the art MARL methods. Constraint 2 will be evaluated empirically. We also compare this approach to a similar idea of social influence by Jaques et al. (2019). First, we will introduce the concepts of MARL and IM in more detail. We will then define the specific formalism for TE used, and then simulate three increasingly harder, multi-agent, cooperation

scenarios. We will also look at how the better reward was obtained, and discuss the difficult switch from a indexical to an action oriented communication strategy.

Related work

Multi-Agent Reinforcement Learning

There is a large body of research on constructing agents that are robust to their partners. In self-play, for example, agents train against themselves rather than a fixed opponent strategy to prevent developing exploitable strategies (Tesauro, 1994). Population based-training goes one step further by training agents to play against a population of other agents rather than only a copy of itself. For instance, some methods train an ensemble of policies with a variety of collaborators and competitors (Jaderberg et al., 2018; Lowe et al., 2017). By using a whole population rather than only a copy of itself, the agent is forced to deal with a wide variety of potential strategies instead of a single strategy. However, it requires a great deal of engineering because the policy parameters suitable for the previous environment are not necessarily the next stage's best initialization.

Some works combine the minimax framework and MARL to find policies that are robust to opponents with different strategies. Minimax is a concept in game theory that can be applied to find an approach that minimizes the possible loss in a worst-case scenario (Osborne et al., 2004). Li et al. (2019) use it during training to optimize the reward for each agent under the assumption that all other agents act adversarial. We are interested in methods that can deal with perturbations in the training partners' behavior, which differs from dealing with partners with various strategies.

Recent works look at settings in which one RL agent, that is trained separately, must join a group of new agents (Lerer and Peysakhovich, 2018; Tucker et al., 2020; Carroll et al., 2019). For example, Carroll et al. (2019) build a model of the other agents which can be used to learn an approximate best response using RL. Lupu et al. (2021) propose to generate a large number of diverse strategies and then train agents that can adapt to other agents' strategies quickly using meta-learning. A related problem is zero-shot coordination (Hu et al., 2020) in which agents need to cooperate with unseen partners at test time. The focus of our paper is not to perform well with novel partners at test-time or build complex opponent models. Our aim is to train agents together to remain attentive and reactive towards their partners' policies.

Intrinsic Social Motivation

Due to centralized training in MARL, agents might adopt non-reactive strategies that may struggle with other agents' changing behaviors. Social intrinsic motivation can give an additional incentive to find reactive policies towards other agents.

IM in Reinforcement learning (RL) refers to reward functions that allow agents to learn interesting behavior, sometimes in the absence of an environmental reward (Chentanez et al., 2005). Computational models of IM are generally separated into two categories (Baldassarre and Mirolli, 2013), those that focus on curiosity (Burda et al., 2018; Pathak et al., 2017) and exploration (Gregor et al., 2016; Eysenbach et al., 2018), and those that focus on competence and control (Oudeyer and Kaplan, 2009; Karl et al., 2017). The information-theoretic Empowerment formalism (Klyubin et al., 2005) is in the latter category, trying to capture how much an agent is in control of the world it can perceive. Empowerment has produced robust behavior linked to controllability, operationality and self-preservation - in both robots (van der Heiden et al., 2020; Karl et al., 2017; Leu et al., 2013) and simulations (Guckelsberger et al., 2016), with (de Abril and Kanai, 2018) and without (Guckelsberger et al., 2018) reinforcement learning and neural network approximations (Karl et al., 2017).

Empowerment has also been applied to multi-agent simulations, under the term of coupled empowerment maximization (Guckelsberger et al., 2016), in which it was used to produce supportive and antagonistic behavior. Of particular interest is the idea of transfer empowerment - introduced in those two papers - a measure that quantifies concepts such as operational proximity and social influence, and led to behaviours such as collaboration, coordination, and lead-taking (Salge and Polani, 2017).

Similar techniques quantify the interaction between agents for improving coordination between agents. Barton et al. (2018) analyze the degree of dependence between two agents' policies to measure coordination, specifically by using Convergence Cross Mapping (CCM). Strouse et al. (2018) show how agents can share (or hide) intentions by maximizing the mutual information between actions and a categorical goal. One notably relevant work is by Jaques et al. (2019) called social influence, which is the influence of one agent on the policies of other agents, measured by the mutual information between action pairs of distinct agents. Similarly, Mahajan et al. (2019), compute the mutual information between agents' trajectories and a latent variable that captures the joint behavior. Wang et al. (2019) compute the mutual information between the transition dynamics of agents.

In contrast to social influence (SI), transfer empowerment considers the *potential* mutual information or channel capacity. When optimizing for *actual* mutual information, its value is bounded from above by the lowest entropy of both agent's action variables. SI might easily interfere with an exploitation strategy and may need regularization once a good strategy is found. On the other hand, empowerment does not have this limitation and the action sets could have very narrow distributions, while still being reactive.

Model

First, let us define a general model that captures multi-agent scenarios and lets use define transfer empowerment. Let us consider a Dec-POMDP, an extension of the MDP for multi-agent systems, being both decentralized and partially observable (Nair et al., 2003). This means that each of the N agents conditions the choice of its action on its partial observation of the world. It is defined by the following tuple: $\langle S, \boldsymbol{A}, T, O, \boldsymbol{O}, R, N \rangle$. S is the set of states and $\boldsymbol{A} = \times_{i \in [1,..,n]} \boldsymbol{A}^i$ the set of joint actions. At each time step, the state transition function $P(s_{t+1}|s_t, \boldsymbol{a}_t)$ maps the joint action and state to a new state. As the game is partially observable, we have a set of joint local observations, $\boldsymbol{O} = \times_{i \in [1,..,n]} \boldsymbol{O}^i$ and an observation function O. Each agent i selects an action using their local policy $\pi^i(a_t^i|o_t^i)$.

We consider fully cooperative tasks, so agents share a reward $r(s_t, \boldsymbol{a_t})$ which conditions on the joint action and state. The goal is to maximise the expected discounted return $J(\boldsymbol{\pi}) = \mathbb{E}_{\tau \sim \pi}\left[R(\tau)\right] = \mathbb{E}_{\tau \sim \pi}\left[\sum_{t=0}^{T} \gamma^t r_t\right]$, with discount factor $\gamma \in [0, 1]$ and horizon T. The expectation is taken w.r.t. the joint policy $\boldsymbol{\pi} = [\pi^1, \ldots, \pi^N]$ and trajectory $\tau = (\boldsymbol{o}_0, \boldsymbol{a}_0, \ldots, \boldsymbol{o}_T)$.

Methodology

This section describes an additional heuristic that biases the learning process in obtaining policies that are reactive to other agents' actions. First, we introduce our specific version of transfer empowerment, which rewards the idea of an agent being responsive to adaptations in the other's policy. Then we explain how to train agents in a multi-agent environment.

Transfer Empowerment

Consider two agents, j and k, both taking actions and changing the overall state. Each time agent k acts, the state of agent j is modified, and j's policy indirectly conditions on k's actions. The objective of coordination is that by changing the actions of agent k, agent, j also *reliably* adapts its actions. Here we look at transfer empowerment, namely the *potential* causal influence that one agent has on another. It is defined for pairs of agents by the channel capacity between one agent's action a_t^k and another agent's action a_{t+1}^j at subsequent time steps and conditioned on the current state s_t, which can be computed by maximizing the mutual information $\mathcal{I}$ between those values, with regards to ω^k :

$$\mathcal{E}^{k \to j}(s_t) = \max_{\omega^k} \mathcal{I}\left(A_{t+1}^j, A_t^k \middle| s_t\right). \tag{1}$$

Here, $\omega^k(a_t^k|s_t)$ is the *hypothetical* policy of agent k, that takes an action a_t^k after observing state s_t and influencing a_{t+1}^j at a later time step. Note that the policy $\omega^k(a_t^k|s_t)$ that maximises the mutual information is not necessarily used for action generation, but simply to compute the channel capacity by looking at all potential policies for the one with the highest mutual information $\mathcal{I}$.

Our version of transfer empowerment differs slightly from the one introduced by Salge and Polani (2017), as we consider the potential information flow, or channel capacity, from one agent's actions to another agent's *actions* in subsequent time steps. Salge and Polani (2017) on the other hand, consider the transfer empowerment between one agent's action and another agent's *sensor* state. We make this modification to address to challenges already discussed in those earlier papers.

One issues is that transfer empowerment to another agent's sensory state captures the direct influence on the other agent's environment. This influence, or information flow can take two pathways. The influence can either act directly on the world the other agent perceives, or alternatively, the other agent can perceive the action's of the first agent, and react to them, modifying their own Umwelt. The difference is that in the second case the information flows through the second agent, and requires a degree of attention to, and reactivity to the first agent's actions. In the first case, the second agent can be fully passive and just have its perceived world changed by the first agent. Since we wanted to create a motivation for more reactivity, we used action-to-action TE, because the only way to influence another agent's actions is by having them react to what you do, i.e. have information flow through the other agent.

The other challenge of TE is the necessity to define distinct sensor states for both agents - otherwise TE is identical to self-empowerment. If every agent only has sensor access to a limited part of the world, this is straight forward. But in many simpler models the access to the world is absolute for both agents, or limitations end up being somewhat arbitrary, or design choices that influence the final behaviour. Looking at the actions, rather then the sensor states, offers a principled alternative, as action's of different agent are usually distinct.

Transfer empowerment has ties with, but is different from, social influence (Jaques et al., 2019). Social influence is the mutual information between agents' actions. It is high when both action variables have a particular entropy, e.g., policies taking different actions. However, towards the end of the training, a high entropy policy distribution might be suboptimal. Our method, on the other hand, considers the *potential* and not *actual* information flow, so agents only calculate how they *could* influence and react to each other, rather than carrying out its potential. As such, action sets can have very narrow distributions; as long as the system would still be reactive *if*, those actions change. Therefore it does not interfere with obtaining optimal policies.

Multi-Agent Training

Training with transfer empowerment results in joint policies that are reactive to their partner's actions, because for the value to be high, it requires considering the decisions of others. As such, transfer empowerment rewards a very general idea of coordination that requires paying attention to each other, and reliably reacting to a variation in their actions. While empowerment does not measure how this reaction looks, or even if it is good, combined with the actual reward should lead to the selection of a strategy that both solves the problem while also avoiding the brittleness that comes from not being reactive to the information from other agents' policies. Specifically, we will modify the agents' reward function so that it becomes:

$$\tilde{r}(s_t, \boldsymbol{a_t}) = r(s_t, \boldsymbol{a_t}) + \sum_{s_{t+1}} P(s_{t+1}|s_t, \boldsymbol{a_t}) \sum_{j=1}^{N} \mathcal{E}^{-j \to j}(s_{t+1}), \tag{2}$$

where $-j$ means all agents excluding agent j. To simplify notation, we will use j instead of $-j \to j$ in the superscript. The new RL objective becomes:

$$J(\boldsymbol{\pi}) = \mathbb{E}_{\tau \sim \boldsymbol{\pi}} \left[\sum_{t=0}^{T} \gamma^t \tilde{r}(s_t, \boldsymbol{a_t}) \right].$$

This new return motivates the potential influence of information between agents' actions, thereby stimulating them to act informatively and react reliably.

Efficient Implementation

We now introduce an efficient implementation to estimate empowerment. We use a for agent k's action at time t and a' for agent j's action at time $t+1$. Mutual information is defined as:

$$\begin{aligned} \mathcal{I}(A, A'|s) &= \mathrm{KL}(p(a, a'|s) \| p(a|s)p(a'|s)) \\ &= \sum_a \sum_{a'} p(a, a'|s) \ln \frac{p(a, a'|s)}{p(a|s)p(a'|s)}, \end{aligned}$$

where KL is the KL divergence. We can substitute $p(a, a'|s)$ and cancel out terms:

$$\begin{aligned} &\sum_a \sum_{a'} p(a, a'|s) \ln \frac{p(a, a'|s)}{p(a|s)p(a'|s)} \\ &= \sum_a \sum_{a'} p(a, a'|s) \ln \frac{p(a|a', s)\cancel{p(a'|s)}}{p(a|s)\cancel{p(a'|s)}} \\ &= \sum_a \sum_{a'} p(a, a'|s) \ln \frac{p(a|a', s)}{p(a|s)}. \end{aligned}$$

By choosing a variational approximator $q(a|a', s)$, with the property $\mathrm{KL}(p(a|a', s)||q(a|a', s)) \geq 0$, we obtain a lower bound on the mutual information:

$$\begin{aligned} \mathcal{I}(A, A'|s) &\geq \hat{\mathcal{I}}(A, A'|s) \\ &:= \sum_a \sum_{a'} p(a, a'|s) \ln \frac{q(a|a', s)}{p(a|s)} \\ &= \sum_a \sum_{a'} p(a, a'|s) \left(\ln q(a|a', s) - \ln p(a|s) \right) \\ &= \mathbb{E}_{p(a,a'|s)} \left[\ln q(a|a', s) - \ln p(a|s) \right]. \end{aligned}$$

The gradient of the lower bound can be approximated by Monte-Carlo sampling. Furthermore, the overall training procedure can be implemented efficiently when representing the distributions by neural networks and using gradient ascent. So the gradient computed over S samples:

$$\begin{aligned} \nabla_\theta \hat{\mathcal{I}}_\theta(A, A'|s) &= \nabla_\theta \mathbb{E}_{p(a,a'|s)} \left[\ln q_\theta(a|a', s) - \ln \omega_\theta(a|s) \right] \\ &\approx \frac{1}{S} \sum_{m=1}^{S} \nabla_\theta \left(\ln q_\theta(a_m|a'_m, s) - \ln \omega_\theta(a_m|s) \right), \end{aligned}$$

where we substituted $p(a|s)$ with $\omega_\theta(a|s)$ and $q(a|a', s)$ with $q_\theta(a|a', s)$, to denote functions parametrized by θ.

Partial Observable

The objective in the previous section was to estimate the empowerment value for a particular state s. However, our main goal is to train policies to be reactive towards the actions of their partners. Let a policy for agent j be π^j_χ with parameters χ. As each policy is conditioned on its local observations, the lower bound on mutual information for agent j becomes:

$$\begin{aligned} &\hat{\mathcal{I}}^j_{\theta,\chi}(A, A'|\boldsymbol{o}) = \\ &\quad \mathbb{E}_{\boldsymbol{o'} \sim p_\nu, a^j \sim \pi^j_\chi, a^k \sim \omega^k_\theta} \left[\ln q_\theta(\boldsymbol{a}|\boldsymbol{o}, \boldsymbol{o'}, a'^j) - \ln \boldsymbol{\omega}^{-j}_\theta(\boldsymbol{a}^{-j}|\boldsymbol{o}^{-j}) \right], \end{aligned} \tag{3}$$

where samples are generated by a learned transition model $\boldsymbol{o'} \sim p_\nu(\boldsymbol{o'}|\boldsymbol{o}, \boldsymbol{a})$. The actions are selected by the target policy $a^j \sim \pi^j_\chi(a^j|o^j)$ and behavior policy $a^k \sim \omega^k_\theta(a^k|o^k)$ and the joint action is $\boldsymbol{a} = (a^1, \ldots, a^j, \ldots, a^N)$ where $a^j \sim \pi_\chi$, $a^k \sim \omega_\theta$ and $k \neq j$.

Notice that the actions come from $\boldsymbol{\omega}_\theta$ and π_χ. The former is the joint behavior policy and the latter is the target policy of agent j. The behavioral policy is only used to train agent j's policy with empowerment but will not generate extrinsic environmental rewards.

This training procedure has two interesting properties. First, it estimates a state's empowerment value. This is done by increasing the diversity of agents' actions while ensuring that these are retrievable from agent j's actions. Actions

that affect j's policy, e.g., informative, are chosen more often than those with a lower effect. Second, it trains agent j's policy to be reactive towards the actions of its partners, because we compute the gradient of mutual information w.r.t. χ to directly optimize π_χ. We provide the description of the full algorithm in the Appendix, which also describes how our method applies to settings with more than 2 agents.

Altogether, empowerment prefers states that allow for information flow between agents, altering policies to be more responsive. We will experimentally verify this in the next section.

Experimental Results

We adopt the simulator developed for testing multi-agent reinforcement learning algorithms [1] that allows creating cooperative and competitive environments. Agents have a continuous observation space and a discrete actions space.

Scenarios

We use a cooperative two-dimensional environment consisting of two agents. The (disembodied) speaker agent can, each time step, choose a communication action that is broadcast to the listener. The listener can choose from 5 physical actions, moving up, down, left and right, or doing nothing. The environment contains a series of L, randomly placed, landmarks. Only the speaker has a signal informing it which of the L landmarks is the target. The objective for the randomly placed listener is to reach the target landmark by decoding the speaker's message. The speaker can send a symbol chosen from a set of C distinct symbols. The team reward is the negative squared distance between the listener and the target landmark, which is given out every time step. The game ends after 100 time steps. To perform well the listener has to quickly move onto the landmarks. We developed three tasks in the environment with increasing difficulty. Fig. 2 visualises the tasks.

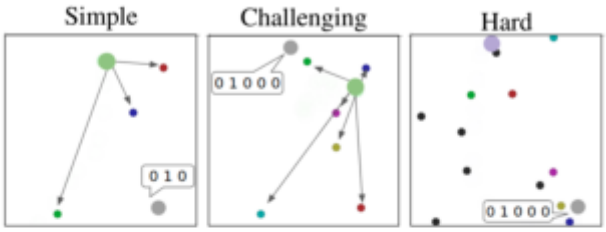


Figure 2: Visualizations of the three tasks. Small dark circles indicate landmarks and obstacles, and the big circles are the listener and speaker. The listener observes the relative distance to the landmarks indicated by the arrows. The speaker's massages are one-hot vectors displayed by the speaker boxes.

Simple The number of symbols $K = |C| = 3$ equals the number of landmarks $L = 3$. The speaker observes the color of the target landmark, while the listener sees a distance vector pointing to each colored landmark.

This scenario could be solved well by an indexical communication strategy, where the speaker simply has to consistently assign a symbol to each landmark color, and then simply relay the information to the speaker, who then has to minimise the distance to the landmark of that color.

Challenging The second task involves more landmarks $L = 6$ than distinct symbols $K = |C| = 5$. The speaker observes the target *position* and the listener's position, while the listener observes the landmarks' positions and the messages sent by the speaker. Here, an *action-oriented strategy*, e.g., indicating movement direction, is likely optimal because the speaker cannot use each symbol for a landmark

[1] https://github.com/openai/multiagent-particle-envs

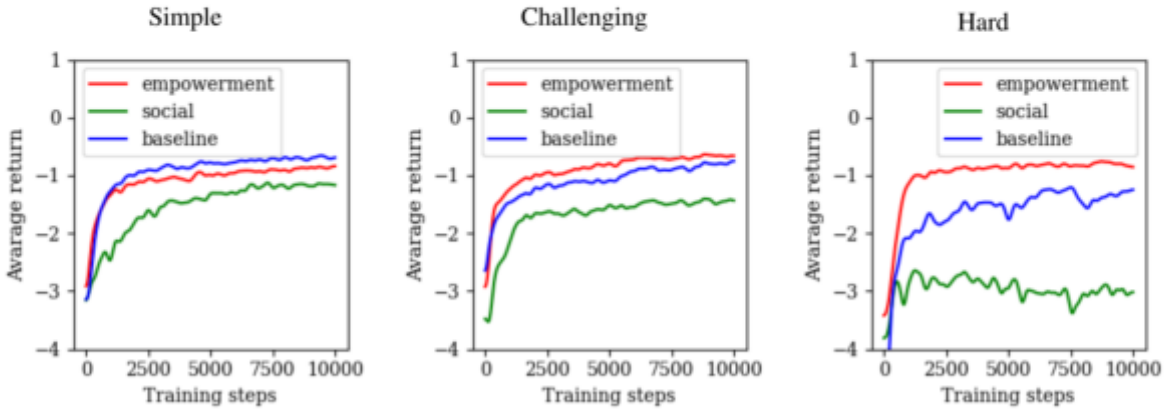


Figure 1: Learning curves for the the three tasks. The rewards are averaged over the steps in an episode to obtain the return. The returns are averaged over three training runs.

uniquely, nor do the landmarks have any identifying features that are easy to community, i.e. they are not colored anymore. Using symbols to direct the listener now requires the speaker to observe and react to the listener's position with an updated signal, and put more cognitive demands on the speaker, who could simply relay its internal signal in the simple scenario.

Hard The last task adds $M = 6$ obstacles, and the reward includes a penalty if the listener hits an obstacle. Furthermore, the landmarks' positions are now unobserved by the listener. These two features increase the difficulty because a higher precision is required. First, the listener has to avoid obstacles, and second, the listener is even more dependent on the speaker because it does not see the landmarks.

Reward The reward function is determined from the position, $\boldsymbol{p} = [p_x, p_y]$, of the listener (agent 1) $\boldsymbol{p}^1$, target $\boldsymbol{p}^g$ and obstacles $\boldsymbol{p}^o$. The state is defined as $s_t = [\boldsymbol{p}_t^1, \boldsymbol{m}_t, \boldsymbol{p}^g, \boldsymbol{p}^{o,1}, \ldots, \boldsymbol{p}^{o,M}, \boldsymbol{p}^{l,1}, \ldots, \boldsymbol{p}^{l,L}]$ for M obstacles, and L landmarks. The reward function is:

$$r(s_t, \boldsymbol{a}_t) = -\|\boldsymbol{p}_{t+1}^1 - \boldsymbol{p}^g\| + \text{penalty} \tag{4}$$

$$\text{penalty} = \begin{cases} -1 & \text{if} \quad \exists j \in [1, ..., M] : \|\boldsymbol{p}_{t+1}^1 - \boldsymbol{p}^{o,j}\| < 0.15 \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

The observation for the speaker and listener $o_t^0 = [\boldsymbol{p}_t^1, \boldsymbol{p}^g, \boldsymbol{p}^{l,1}, \ldots, \boldsymbol{p}^{l,L}]$ and $o_t^1 = [\boldsymbol{v}_t^1, \boldsymbol{m}_t^0]$, respectively. v is the velocity and m the message, represented by a one-hot vector. The listener's action is a force vector $a^1 = [f_x, f_y]$, while the speaker's action is a message $a^0 = [c^0, \ldots, c^K]$ with vocabulary size K. The listener's position is updated according to the following equation:

$$\begin{bmatrix} \boldsymbol{p} \\ \boldsymbol{v} \\ \dot{\boldsymbol{v}} \end{bmatrix}_{t+1} = \begin{bmatrix} \boldsymbol{p} + \boldsymbol{v}\Delta t \\ \zeta \boldsymbol{v} + \dot{\boldsymbol{v}}\Delta t \\ \frac{\boldsymbol{u}}{\text{mass}} \end{bmatrix}_t \tag{6}$$

with damping coefficient $\zeta = 0.5$ and mass $= 1$. The speaker's messages, will be added to the state at the next time-step: $\boldsymbol{m}_t = \boldsymbol{a}_{t-1}^0$. As is common when working with policies parameterised by neural networks, the actions are one-hot vectors, obtained by Gumbel-Softmax function (Murphy, 2022). For example, the actions of the speaker are converted into

$$\text{one-hot}(\boldsymbol{a^0}) = [\mathbb{I}(a_1^0 = \max(\boldsymbol{a^0})), \ldots, \mathbb{I}(a_K^0 = \max(\boldsymbol{a^0}))].$$

Comparison and Implementation details

We compare our method with MADDPG (Lowe et al., 2017) (baseline) and social influence (Jaques et al., 2019) (social infl.). Our method (empowerment) is built on top of the MADDPG, a centralized actor-critic method. Social influence is a decentralized method. The agents' policies are parameterized by a two-layer ReLU MLP with 64 units per layer. The messages sent between agents are soft approximations to discrete messages, calculated using the Gumbel Softmax estimator. All models are trained for 10k episodes, of which an episode consists of 25 interactions. Source code can be found at https://github.com/tessavdheiden/social_empowerment.

Results and Discussion

Learning Curves

Our two main hypothesis are that adding TE to MARL produces faster adaptation (needs less training steps), and achieves better, overall results. To answer both of the questions, we compare the learning curves, over 10k training steps, averaged over three runs, for both the MARL baseline, and with the addition of TE (empowerment) and Social influence (SI) (social). Figure 1 shows the averaged return after a given number of training steps. A higher score is better, it shows that the listener is closer to the target. Since the listener starts away from the target a score of 0 is impossible, all scores are negative.

The learning speed seems to be comparable between models in difference scenarios, i.e. it takes about the same time for the different algorithm to reach their peak, final performance. Only SI seems to learn slower in both the simple and challenging task. Performances seem to mostly stabilise after some point, so we can also take a closer look at the performances of the trained agents after 10k training steps.

Final Scores

Details of the results are presented in Table 1. It shows the average distance and the percentage of collisions with an obstacle for the final agents, computed for 100 episodes.

The baseline obtains the top performance, with lowest distance of 0.221, in the simple task, requiring a simple, lexical strategy. Here empowerment performs worse with 0.414, with social influence performing even worse with

Table 1: The values show the average distance between the listener and target landmark and the percentage of collisions with obstacles. The results are computed for 100 episodes after training with 10k episodes.

	Simple	Challenging	Hard	
	Average distance	Average distance	Average distance	Obstacle hit
basel.	0.221	0.440	0.520	0.603
SI	0.716	0.949	1.076	0.220
TE	0.414	0.460	0.440	0.266

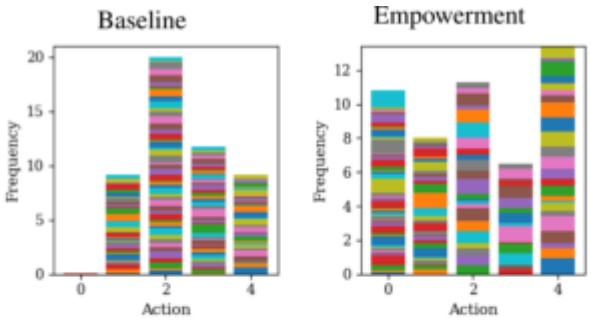


Figure 3: The action distributions for the listener in the hard task for both the baseline and empowerment. The colors indicate each a different episode. 0 is the wait action, 1 - 4 are cardinal accelerations.

0.716. This is an indication of the idea that an added incentive might get in the way of exploiting an easy to find, simple strategy.

In the challenging task, which requires an action-oriented strategy empowerment seems to perform similar to the baseline, while it clearly outperforms the baseline for the hard task, both in terms of average distance, and in terms of obstacles hit. The difference in hit obstacles is particularly large, indicating that TE helps with the higher reactivity required within an episode to navigate around the obstacles. Social influence seems to struggle with both tasks, again likely due to an interference between the added reward and the best exploitation strategy.

In contrast, the better performance of empowerment in the hard challenge, compared to the baseline, must be due to empowerment helping to discover a better overall strategy - as the baseline implementation would be fully capable of producing a strategy identical to the one performance by the TE framework, had it discovered it.

To illuminate this difference, we can take a look at the action distribution for the speaker and listener agents over several episodes, using the agents after 10k training episodes. Fig. 3 shows how often the five available actions were used. Action 0 for the listener is the waiting action - and we see that this one is not used by the baseline. We speculate that it might be difficult for the listener to learn when to use this action, as it is detrimental in most cases. The bias towards reactivity induced by TE might help to keep this rare action as an option - following a symbol by the speaker that might become a "stop" signal.

We can also take three trained listener agents and compare what they will do when we provide them with a fixed speaker signal over several time steps, to figure out what those symbols directed them to do. Fig. 4 shows the trajectories resulting from this, with each color denoting a different forces symbol by the speaker. The baseline has a relatively good separation into cardinal actions, but transfer empowerment

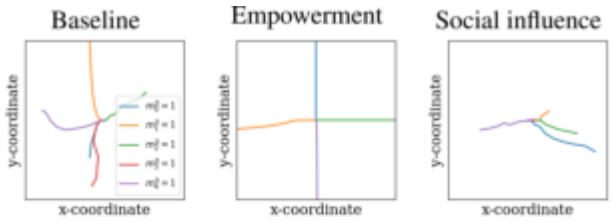


Figure 4: The listener's positions plotted for 10 time steps, given a speaker's message $\boldsymbol{m}^0$, a one-hot vector. The subscript denotes the component in $\boldsymbol{m}^0$ that is equal to a 1.

leads to nearly perfect control by the speaker over the actions of the listener. Note that one signal results in the wait action, leading to no visible trajectory for empowerment.

Conclusions and Future Work

Overall, adding transfer empowerment to MARL seems to improve the overall performance level of cooperative agents - particularly for harder tasks that rely on an action-oriented communication strategy. This seems to indicate that TE helps the learning process to find better solutions to converge on - which remain undiscovered by the baseline MARL with similar training time - while also not getting in the way of exploitation to much. An immediate open question, a direction for future work, is of course the question of generalizability of this approach to different scenarios. Other exciting research directions are scenarios with partners unseen at training time, moving in the direction of one-shot adaptation to partners, and scenarios with competitive, or cooperative-competitive mixed scenarios. Using TE to bias systems towards control, or information hiding to find optimal solutions.

We also showed how an efficient computation of empowerment could be combined with RL for the MARL framework, opening the door for more complex scenarios such as humans interacting with robots. In general, the results in this study are promising for the overall agenda to develop a framework of social intrinsic motivations based on empowerment (or similar measures) to bias an agent towards general social concepts, such as reliable reactivity, or lead-following. The fact that it is based on similar, single-agent intrinsic motivations is also interesting, as it might offer insights into how to transition from single to social agent behavior with only gradual adaptation.

References

Baldassarre, G. and Mirolli, M. (2013). *Intrinsically motivated learning in natural and artificial systems*. Springer.

Barton, S. L., Waytowich, N. R., Zaroukian, E., and Asher, D. E. (2018). Measuring collaborative emergent behav-

ior in multi-agent reinforcement learning. In *International Conference on Human Systems Engineering and Design: Future Trends and Applications*, pages 422–427. Springer.

Burda, Y., Edwards, H., Pathak, D., Storkey, A., Darrell, T., and Efros, A. A. (2018). Large-scale study of curiosity-driven learning. *arXiv preprint arXiv:1808.04355*.

Carroll, M., Shah, R., Ho, M. K., Griffiths, T., Seshia, S., Abbeel, P., and Dragan, A. (2019). On the utility of learning about humans for human-ai coordination. *Advances in neural information processing systems*, 32.

Chentanez, N., Barto, A. G., and Singh, S. P. (2005). Intrinsically motivated reinforcement learning. In *Advances in neural information processing systems*, pages 1281–1288.

de Abril, I. M. and Kanai, R. (2018). A unified strategy for implementing curiosity and empowerment driven reinforcement learning. *arXiv preprint arXiv:1806.06505*.

Eysenbach, B., Gupta, A., Ibarz, J., and Levine, S. (2018). Diversity is all you need: Learning skills without a reward function. *arXiv preprint arXiv:1802.06070*.

Gregor, K., Rezende, D. J., and Wierstra, D. (2016). Variational intrinsic control. *arXiv preprint arXiv:1611.07507*.

Guckelsberger, C., Salge, C., and Colton, S. (2016). Intrinsically motivated general companion npcs via coupled empowerment maximisation. In *2016 IEEE Conference on Computational Intelligence and Games (CIG)*, pages 1–8. IEEE.

Guckelsberger, C., Salge, C., and Togelius, J. (2018). New and surprising ways to be mean. adversarial npcs with coupled empowerment minimisation. *arXiv preprint arXiv:1806.01387*.

Hu, H., Lerer, A., Peysakhovich, A., and Foerster, J. (2020). " other-play" for zero-shot coordination. *arXiv preprint arXiv:2003.02979*.

Jaderberg, M., Czarnecki, W., Dunning, I., Marris, L., Lever, G., Castaneda, A., et al. (2018). Human-level performance in first-person multiplayer games with population-based deep reinforcement learning. arxiv. *arXiv preprint arXiv:1807.01281*.

Jaques, N., Lazaridou, A., Hughes, E., Gulcehre, C., Ortega, P., Strouse, D., Leibo, J. Z., and De Freitas, N. (2019). Social influence as intrinsic motivation for multi-agent deep reinforcement learning. In *International Conference on Machine Learning*, pages 3040–3049. PMLR.

Karl, M., Soelch, M., Bayer, J., and Van der Smagt, P. (2016). Deep variational bayes filters: Unsupervised learning of state space models from raw data. *arXiv preprint arXiv:1605.06432*.

Karl, M., Soelch, M., Becker-Ehmck, P., Benbouzid, D., van der Smagt, P., and Bayer, J. (2017). Unsupervised real-time control through variational empowerment. *arXiv preprint arXiv:1710.05101*.

Klyubin, A. S., Polani, D., and Nehaniv, C. L. (2005). All else being equal be empowered. In *European Conference on Artificial Life*, pages 744–753. Springer.

Lerer, A. and Peysakhovich, A. (2018). Learning social conventions in markov games. *arXiv preprint arXiv:1806.10071*.

Leu, A., Ristić-Durrant, D., Slavnić, S., Glackin, C., Salge, C., Polani, D., Badii, A., Khan, A., and Raval, R. (2013). Corbys cognitive control architecture for robotic follower. In *Proceedings of the 2013 IEEE/SICE International Symposium on System Integration*, pages 394–399. IEEE.

Li, S., Wu, Y., Cui, X., Dong, H., Fang, F., and Russell, S. (2019). Robust multi-agent reinforcement learning via minimax deep deterministic policy gradient. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 4213–4220.

Lowe, R., Wu, Y. I., Tamar, A., Harb, J., Abbeel, O. P., and Mordatch, I. (2017). Multi-agent actor-critic for mixed cooperative-competitive environments. In *Advances in neural information processing systems*, pages 6379–6390.

Lupu, A., Cui, B., Hu, H., and Foerster, J. (2021). Trajectory diversity for zero-shot coordination. In *International Conference on Machine Learning*, pages 7204–7213. PMLR.

Mahajan, A., Rashid, T., Samvelyan, M., and Whiteson, S. (2019). Maven: Multi-agent variational exploration. In *Advances in Neural Information Processing Systems*, pages 7613–7624.

Murphy, K. P. (2022). *Probabilistic Machine Learning: An introduction*. MIT Press.

Nair, R., Tambe, M., Yokoo, M., Pynadath, D., and Marsella, S. (2003). Taming decentralized pomdps: Towards efficient policy computation for multiagent settings. In *IJCAI*, volume 3, pages 705–711. Citeseer.

Osborne, M. J. et al. (2004). *An introduction to game theory*, volume 3. Oxford university press New York.

Oudeyer, P.-Y. and Kaplan, F. (2009). What is intrinsic motivation? a typology of computational approaches. *Frontiers in neurorobotics*, 1:6.

Pathak, D., Agrawal, P., Efros, A. A., and Darrell, T. (2017). Curiosity-driven exploration by self-supervised prediction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 16–17.

Salge, C. and Polani, D. (2017). Empowerment as replacement for the three laws of robotics. *Frontiers in Robotics and AI*, 4:25.

Strouse, D., Kleiman-Weiner, M., Tenenbaum, J., Botvinick, M., and Schwab, D. J. (2018). Learning to share and hide intentions using information regularization. In *Advances in Neural Information Processing Systems*, pages 10249–10259.

Tesauro, G. (1994). Td-gammon, a self-teaching backgammon program, achieves master-level play. *Neural computation*, 6(2):215–219.

Tucker, M., Zhou, Y., and Shah, J. (2020). Adversarially guided self-play for adopting social conventions. *arXiv preprint arXiv:2001.05994*.

van der Heiden, T., Weiss, C., Shankar, N. N., Gavves, E., and van Hoof, H. (2020). Social navigation with human empowerment driven reinforcement learning. *arXiv preprint arXiv:2003.08158*.

Wang, T., Wang, J., Wu, Y., and Zhang, C. (2019). Influence-based multi-agent exploration. *arXiv preprint arXiv:1910.05512*.

Appendix

Algorithm 1 explains how we train with empowerment. We omit super- and subscripts denoting time, agent and batch indices whenever clear from the context.

Algorithm 1 Training joint policy $\boldsymbol{\pi}_\theta$ with empowerment

Require: Initialisation of networks $\boldsymbol{\pi}_\chi$, $\boldsymbol{Q}_\psi$, $\boldsymbol{\omega}_\theta$, q_θ and p_ν, and target networks $\boldsymbol{\pi}_\phi$, $\boldsymbol{Q}_\zeta$.
  for each episode **do**
    for each time step **do**
      $\boldsymbol{o} = O(s)$, $\boldsymbol{a} \sim \boldsymbol{\pi}(\boldsymbol{a}|\boldsymbol{o})$, $s' \sim f(s, \boldsymbol{a})$
      $\tau = \tau \cup \{(\boldsymbol{o}, \boldsymbol{a}, \boldsymbol{o}', r, \hat{\mathcal{I}}, y)\}$
    end for
    $\mathcal{D} = \mathcal{D} \cup \tau$
    for each agent i **do**
      sample minibatch with S tuples from $\mathcal{D}$
      $\hat{\mathcal{I}}_{\chi^i,\theta^i}(\boldsymbol{o}) = \texttt{computeLowerBound}(\boldsymbol{o}, \boldsymbol{\pi}_\chi, \boldsymbol{\omega}_\theta, q_\theta, p_\nu)$ ▷ Equation 3
      $y = r + \hat{\mathcal{I}}_{\chi^i,\theta^i}(\boldsymbol{o}) + \gamma Q^i_\zeta(\boldsymbol{o}, \boldsymbol{a})$
      $\mathcal{L}(\psi^i) = \frac{1}{S}\sum_j (y - Q^i_\psi(\boldsymbol{o}_j, \boldsymbol{a}_j))^2|_{a^i_j \sim \pi^i_\phi}$
      $\mathcal{L}(\chi^i) = -\frac{1}{S}\sum_j Q^i_\psi(\boldsymbol{o}_j, \boldsymbol{a}_j)|_{a^i_j \sim \pi^i_\chi}$
      $\texttt{updateCritic}(\mathcal{L}(\psi^i), \psi^i)$ ▷ See (Lowe et al., 2017)
      $\texttt{updateActor}(\mathcal{L}(\chi^i), \chi^i)$
      $\texttt{gradientAscent}(\hat{\mathcal{I}}_{\chi^i,\theta^i}, \theta^i, \chi^i)$
      $\texttt{maxLogLikelihood}(\nu, \boldsymbol{o}, \boldsymbol{a}, \boldsymbol{o}')$ ▷ See (Karl et al., 2016)
      $\texttt{updateTargets}(\phi^i, \zeta^i, \psi^i, \chi^i)$
    end for
  end for

$\boldsymbol{\pi}$	Joint policy with N components $[\pi^1, \ldots, \pi^N]$.
$O(s)$	Deterministic observation function $\boldsymbol{o} = (o^1, \ldots, o^N) = O(s)$.
$\boldsymbol{a}, \boldsymbol{o}$	Joint action and observation $(a^1, \ldots, a^N)$, $(o^1, \ldots, o^N)$.
$\boldsymbol{a}^{-i}, \boldsymbol{o}^{-i}$	Joint action and observation excluding those of agent i.
$J^i(\boldsymbol{\pi})$	Expected return of agent i induced by joint policy $\boldsymbol{\pi}$.
$Q^i(\boldsymbol{o}, \boldsymbol{a})$	Centralised critic of local policy π^i.
$\mathcal{E}^{-i \to i}$	Transfer empowerment from all agents' actions, excluding agent i, towards agent i's action.
$\hat{\mathcal{E}}$	Lower bound on empowerment by employing variational approximation.
$\hat{\mathcal{I}}_\theta$	Lower bound on mutual information computed by θ-parameterized neural networks.

Table 2: Notation

Exploration and exploitation of the adjacent possible space for open-endedness

Mikihiro Suda[1], Takumi Saito[1], Mizuki Oka[1]

[1]University of Tsukuba, Tsukuba, Ibaraki 305-8577 Japan

suda@websci.cs.tsukuba.ac.jp

Introduction

Understanding the mechanisms that create open-ended evolution is considered a grand challenge and a key task in the field of artificial life (Bedau et al., 2000). The source of creating open-endedness is the continuous creation of novel forms. Novel forms are created and evolve as they gain population - understanding this mechanism is key to understanding open-ended evolution. In several models of artificial life, "novelty" has been considered a mutation and has often been assumed to occur randomly. However, recent studies have proposed a model in which novelty does not occur randomly, but rather toward adjacent possible spaces (Tria et al., 2014). The effectiveness of the model is confirmed by the fact that the model simulates the behavior of people with a high degree of accuracy when compared to empirical data constituted from behavioral data on the Internet. The model reveals that the balance between exploration, exploitation, and the search strategy for possible adjacent spaces determines how the network grows.

In this study, we use this agent-based model to analyze how network characteristics differ depending on exploration, exploitation, as well as the search strategy for possible adjacent spaces. The objective is to identify the interactions between nodes that are necessary for the novelty to gain attention and population when they are added to the network.

Agent-based model based on adjacent possible

The idea of *Adjacent Possible* was originally theorized by Stuart Kauffman to explain the evolution of molecules and organisms (Kauffman, 1993). Adjacent possible space refers to the space of possibilities that are one step away from what actually exists and will become reality in the near future. A similar concept can be found in the protein space theory proposed in Smith (1970). The protein space theory argues that gene evolution occurs through the accumulation of minute changes in existing genes and the changes must occur under the restriction that genes can form phenotypes. Kauffman extended and generalized this theory to apply not only to the evolution of genes but also to the evolution of human relationships and many other areas.

Ubaldi et al. (2021) extended Polya's urn model, which incorporates the concept of an adjacency possible space, to an agent-based model to generate a social network. The model generates a network in which agents interact based on two parameters, ρ and ν, and a strategy s, where ρ determines the strength of exploitation, ν determines the strength of exploration, and s is a strategy for exploring Adjacent Possible space.

Each agent has its own urn and is assigned an ID. We select one agent from the environment according to the size of the agent's urn. This agent is the starting point of the interaction, the *caller agent*. Next, one agent (the *called agent*) is drawn from the caller agent's urn to be the interaction partner, and the caller agent's ID is added to the called agent's urn by ρ. This operation gives the model the property of preferential attachment. ρ indicates the strength of exploitation, as it increases the probability of interacting again with a partner with whom one has already interacted.

Next, $\nu + 1$ number of agents are selected from the caller agent's urn according to strategy s and added to the called agent's urn. The operation is performed for called agents. The task of selecting $\nu + 1$ agents from the opponent's agent implies a search for possible adjacent spaces. ν indicates the strength of exploration. How $\nu + 1$ number of agents are selected from the interacting partner agents is defined by the strategy s. Various strategies can be specified, such as selecting randomly (s = *RND*), selecting randomly according to population (s = *WSW*) or exchanging the most recently interacted agents (s = *SSW* or *ASW*). See the original paper for details on the model (Ubaldi et al., 2021).

It has been reported that this model can reproduce real-world data with high accuracy when the parameters are set appropriately. For example, the Twitter mention network has $\rho = 5$, $\nu = 5$ and strategy $s =$ *WSW* can successfully reproduce its dynamics. Similarly, the American physical society co-authorship network has $\rho = 6$, $\nu = 15$ and strategy $s =$ *SSW*. The mobile phone network has $\rho = 21$, $\nu = 7$ and strategy $s =$ *ASW*.

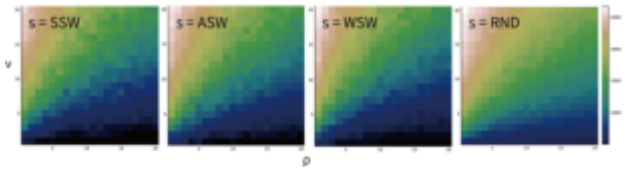


Figure 1: Network size according to different ρ, ν and strategy s.

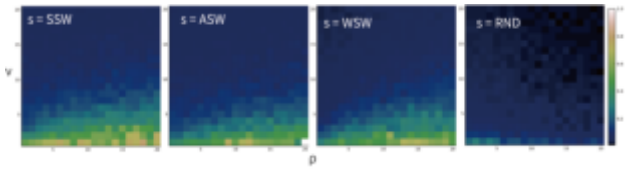


Figure 2: Network cluster according to different ρ, ν and strategy s.

Experiment

We simulated the agent-based model described above and analyzed how the network changes when ρ, ν, and strategy s are changed.

Figure 1 shows the results of how the network size changes. Here, SSW, ASW is the strategy of exchanging the most recently interacted agents, WSW is the strategy of randomly selecting weights on previously chosen agents, *RND* is the strategy of exchanging agents completely at random. If $R = \rho/\nu$, we observe the property that the smaller R is, the larger the size of the network. Observing by strategy, *RND* tends to increase the size of the network compared to other strategies.

Figure 2 shows how the cluster coefficients change. It is observed that the cluster coefficients also correlate with R similar to the network size, but the correlation seems to be smaller. It is also observed that the cluster coefficients in strategy *RND* are very small.

The higher the number of agents randomly selected from the possible adjacency space, the higher the probability of selecting previously less selected agents. Thereby, the network size increases because it acquires many nodes with a small number of edges and the cluster coefficients become smaller. Given open-ended evolution, it is suggested that the value of exploration (ν) is greater than the value of exploitation (ρ) and that randomly selecting agents from the adjacency possible space is important.

Next, with ρ and ν fixed at constant values, we analyzed how the frequency of selection of new nodes (novelty) as they appear differs for different strategies. Figure 3 shows the results for different strategies (*ASW*, *SSW*, *WWS*, *RND*) in the Twitter mentions network ($\rho = 5, \nu = 5$). The selection count of nodes, born at the time signified by the x-axis, is selected at the time on the y-axis. A brighter color indicates a higher selection count. In the figure, a blank space

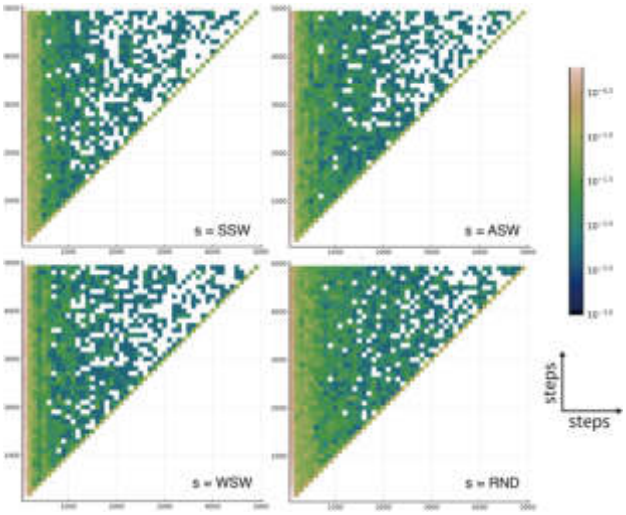


Figure 3: Selection count of nodes, born at the time signified by the x-axis, selected at the time on the y-axis. A brighter color indicates a higher selection count. Twitter mention network ($\rho = 5, \nu = 5$) with four different strategies.

indicates a zero selection. The entire trend is that nodes that have been present since the early steps of network growth are selected more frequently, indicating preferential selection. Compared to *WWS* and *RND*, *ASW* and *SSW* have a stronger preferential attachment effect, and it can be observed that the newer the node is, the less often it is selected. The result that randomly selecting agents to recommend interaction partners contributes to network development is interesting because it is the antithesis of the common social networking service recommendation system, which tends to recommend well-known users.

Conclusion

In this study, we analyzed the nature of interactions to achieve an open-ended network using an agent-based model that incorporates the concept of Adjacency Possible space. The results show that in terms of open-ended evolution, the value of exploration is greater than exploitation and that the more random the search for possible adjacent spaces, the more likely new nodes will be accessed.

It was assumed that all the nodes take the same strategy; however, this is unrealistic. It is common for users to have different ideas about who to recommend to other users. In our future work, we will explore how the obtained population differs when each node has a different strategy, and what kind of network growth will be observed when each node evolves its strategy when obtaining the population.

References

Bedau, M. A., McCaskill, J. S., Packard, N. H., Rasmussen, S., Adami, C., Green, D. G., Ikegami, T., Kaneko, K., and Ray.,

T. S. (2000). Open problems in artificial life. *Artificial Life*, 6(4):363–376.

Kauffman, S. (1993). *The Origins of Order: Self-organization and Selection in Evolution*. Oxford University Press.

Smith, J. M. (1970). Natural Selection and the Concept of a Protein Space. *Nature*, 225(5232):563–564.

Tria, F., Loreto, V., and Servedio, V. D. P. (2014). The dynamics of correlated novelties. *Scientific Reports*, 4(1):5890.

Ubaldi, E., Burioni, R., Loreto, V., and Tria, F. (2021). Emergence and evolution of social networks through exploration of the adjacent possible space. *Communications Physics*, 4(28):2399–3650.

DIAS: A Domain-Independent Alife-Based Problem-Solving System

Babak Hodjat[1], Hormoz Shahrzad[1,2], and Risto Miikkulainen[1,2]
[1]Cognizant AI Labs; [2]The University of Texas at Austin

Abstract

A domain-independent problem-solving system based on principles of Artificial Life is introduced. In this system, DIAS, the input and output dimensions of the domain are laid out in a spatial medium. A population of actors, each seeing only part of this medium, solves problems collectively in it. The process is independent of the domain and can be implemented through different kinds of actors. Through a set of experiments on various problem domains, DIAS is shown able to solve problems with different dimensionality and complexity, to require no hyperparameter tuning for new problems, and to exhibit lifelong learning, i.e. adapt rapidly to run-time changes in the problem domain, and do it better than a standard non-collective approach. DIAS therefore demonstrates a role for Alife in building scalable, general, and adaptive problem-solving systems.

Introduction

Ecosystems in nature consist of diverse organisms each with a generic goal to survive. Survival may require different strategies and actions at different times. Emergent behavior from the collective actions of these organisms then makes it possible for the ecosystem as a whole to adapt to a changing world, i.e. solve new problems as they appear.

Such continual adaptation is often necessary for artificial agents in the real world as well. As a matter of fact, the field of reinforcement learning was initially motivated by such problems: The agent needs to learn while performing the task. While many offline extensions now exist, minimizing regret and finding solutions in one continuous run makes sense in many domains. For instance, there are domains where the fundamentals of the domain are subject to rapid and unexpected change, such as trading in the stock market, and control systems for functions that exhibit chaotic behavior. Similarly in many game-playing domains opponents improve and change their strategies as they play. There are also domains where numerous similar problems need to be solved and there is little time to adapt to each one, such as trading systems with a changing portfolio of instruments, financial predictions for multiple businesses/units, optimizing multiple industrial production systems, optimizing growth recipes for multiple different plants, and optimizing designs of multiple websites.

However, current Artificial Intelligence (AI) systems are not adaptive in this manner. They are strongly tuned to each particular problem, and adapting to changes in it and to new problems requires much domain-specific tuning and tailoring.

The natural ecosystem approach suggests a possible solution: Separate the AI from the domain. A number of benefits could result: First, the AI may be improved in the abstract; it is possible to compare versions of it independently of domains. Second, the AI may more easily be designed to be robust against changes in the domain, or even switches between domains. Third, it may be designed to transfer knowledge from one domain to the next. Fourth, it may be easier to make it robust to noise, task variation, and unexpected effects, and to changes to the action space and state space.

This paper aims at designing such a problem-solving system and demonstrating its feasibility in a number of benchmark examples. In this Domain Independent Alife-based Problem Solving System (DIAS), a population of actors cooperate in a spatial medium to solve the current problem, and continue doing so over the span of several changing problems. The experiments will demonstrate that

- The behaviors of each actor are independent from the problem definition;
- Solutions emerge continually from collective behavior of the actors;
- The actor behavior and algorithms can be improved independently of the domains;
- DIAS scales to problems with different dimensionality and complexity;
- Very little or no hyperparameter tuning is required between problems;
- DIAS can adapt to a changing problem domain, implementing lifelong learning; and
- Collective problem-solving provides an advantage in scaling and adaptation.

DIAS can thus be seen as a promising starting point for scalable, general, and adaptive problem solving, based on principles of Artificial Life.

Related Work

In most population-based problem-solving approaches, such as Genetic Algorithms (GA; Mitchell, 1996; Eiben and Smith, 2015), Particle Swarm Optimization (Sengupta et al., 2018; Rodriguez and Reggia, 2004), and Estimation of Distribution Algorithms (Krejca and Witt, 2020), each population member

is itself a candidate solution to the problem. In contrast in DIAS, the entire population together represents the solution.

Much recent work in Artificial Life concentrates on exploring how fundamentals of biological life, such as reproduction functions, hyper-structures, and higher order species, evolved (Gershenson et al., 2018). However, some Alife work also focuses on potential robustness in problem solving (Hodjat and Shahrzad, 1994). For instance, in Robust First Computing as defined by Ackley and Small (2014), there is no global synchronization, perfect reliability, free communication, or excess dimensionality. DIAS complies to these principles as well. While it does impose periodic boundary conditions, these boundaries can expand or retract depending on the dimensionality of the problem.

This approach is most closely related to Swarm Intelligence systems (Bansal et al., 2019), such as Ant Colony Optimization (Deng et al., 2019). The main difference is that the problem domain is independent from the environment in which the actors survive, i.e. the ecosystem, and a common mapping is provided from the problem domain to the ecosystem. This approach allows for any change in the problem domain to be transparent to the DIAS process, which makes it possible to change and switch domains without reprogramming or restarting the actor population.

Several other differences from prior work result from this separation between actors and problem domains. First, the algorithms that the actors run can be selected and improved independently of the domain and need not be determined a priori. Second, the fitness function for the actors, as well as the mapping between the domain reward function and the actors' reward function, is predefined and standardized, and need not be modified to suit a given problem domain. Third, the actors' state and action spaces are fixed regardless of the problem domain. Fourth, there is no enforced communication mechanism among the actors. While the actors do have the facility to communicate point-to-point and communication might emerge if needed, it is not a precondition to problem solving.

In terms of prior work in the broader field of Universal AI and Domain Independence (Hutter, 2000), most approaches are limited to search heuristics, such as extensions to the A* algorithm (Stern, 2019). Such approaches still require domain knowledge such as the goal state, state transition operators, and costs. While efficient, these approaches lack robustness, and are designed to work on a single domain at a time. They do not do well if the domain changes during the optimization process. In the case of domain-independent planning systems (Della Penna et al., 2009), the elaborate step of modeling the problem domain is still required. Depending on the manner by which such modeling is done, the system will have different performance. In this sense DIAS aims at more general domain-independent problem solving than prior approaches.

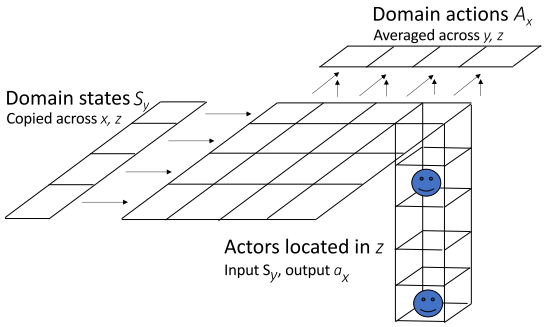


Figure 1: General design of a DIAS system. Actors exist on a three-dimensional grid where x-locations represent the elements of the domain-action vector and y-locations represent the elements of the domain-state vector. The z-locations form a space that the actors can occupy and use for messaging. The grid thus maps the domain space to an actor space where problems can be solved in a domain-independent manner.

Method

A population of independent actors is set up with the goal of surviving in a common environment called a *geo*. The input and output dimensions of the domain are laid out across the geo. Each actor sees only part of the geo, which requires that they cooperate in discovering collective solutions. This design separates the problem-solving process from the domain, allowing different kinds of actors to implement it, and makes it scalable and general. The population adapts to new problems through evolutionary optimization, driven by credit assignment through a contribution measure.

Geo

Actors are placed on a grid called geo (Fig. 1). The dimensions of the grid correspond to the dimensions of the domain-action space (along the x-axis) and the domain-state space (along the y-axis). More specifically, domain action is a vector $\mathbf{A}$; each element A_x of this vector is mapped to a different x-location. Similarly, domain state is a vector $\mathbf{S}$, and its elements S_y are mapped to different y-locations in the geo. There can be multiple actors for each (x, y)-location of the grid. These actors live in different locations of the z dimension. Each (x, y, z) location may contain an actor, as well as a domain-action suggestion and a message, both of which can be overwritten by the actor in that location.

Actors

An actor is a decision-making unit taking an actor-state vector $\boldsymbol{\sigma}$ as its input and issuing an actor-action vector $\boldsymbol{\alpha}$ as its output at each domain time step. All actors operate in the same actor-state and actor-action spaces, regardless of the domain. Each actor is located in a particular (x, y, z) location in the geo grid and can move to a geographically adjacent location. Each actor is also linked to a *linked location* (x', y', z')

elsewhere in the geo. This link allows an actor to take into account relationships between two domain-action elements (A_x and $A_{x'}$) and two domain-state elements (S_y and S'_y) and to communicate with other actors via messages. Thus, it focuses on a part of the domain, and constitutes a part of a collective solution.

The actor-action vectors $\boldsymbol{\alpha}$ consist of the following actions:

- Write a domain-action suggestion a_x in the current location in the geo;
- Write a message in the current location in the geo;
- Write actor's reproduction eligibility;
- Move to a geographically adjacent geo location;
- Change the coordinates of the linked location.
- NOP

The actor-state vectors $\boldsymbol{\sigma}$ consist of the following data:

- Energy e: real ≥ 0;
- Age: integer ≥ 0;
- Reproduction eligibility: True/False;
- Coordinates in the current location: integer $x, y, z \geq 0$;
- Message in the current location: [0..1];
- Domain-action suggestion a_x in current location: [0..1];
- Domain-state value S_y in the current location: [0..1];
- Coordinates in the linked location: integer $x', y', z' \geq 0$;
- Message in the linked location: [0..1];
- Domain-action suggestion $a_{x'}$ in linked location: [0..1];
- Domain-state value $S_{y'}$ in the linked location: [0..1].

Depending on the actor type, actors may choose to keep a history of actor states and refer to it in their decision making.

Problem-solving Process

Algorithm 1 outlines the DIAS problem-solving process. It proceeds through time intervals (in the main while loop). Each interval is one attempt to solve the problem, i.e. a fitness evaluation of the current system. Each attempt consists of a number of interactions with the domain (in the inner while loop) until the domain issues a terminate signal and returns a domain fitness. The credit for this fitness is assigned to individual actors and used to remove bad actors from the population and to create new ones through reproduction.

More specifically, during each domain time step t, the current domain-state vector $\mathbf{S}$ is first loaded into the geo (Step 2.1): Each (x, y, z) location is updated with the domain-state element S_y. Each actor then takes its current actor state $\boldsymbol{\sigma}$ as input and issues an actor action $\boldsymbol{\alpha}$ as its output (Step 2.2). As a result of this process, some actors will write a domain-action suggestion a_x in their location. A domain-action vector $\mathbf{A}$ is then created (Step 2.3): The suggestions a_x are averaged across all locations with the same x to form its elements A_x. If no a_x were written, $A_x(t-1)$ is used (with $A_x(-1) = 0$). The resulting action vector $\mathbf{A}$ is passed to the domain, which executes it, resulting in a new domain state (Step 2.4).

Algorithm 1: The DIAS problem-solving process

```
Initialize_population; solved=False; interval=0
while interval < maxinterval & ¬ solved do
    1. Initialize_domain; terminated=False; t=0
    2. while t < maxt & ¬ terminated do
        2.1 Load S
        2.2 for each actor do
            input σ
            output α
        2.3 for each x do
            Average a_x
        2.4 Execute A
        2.5 t++
    3. Obtain F
    4. if ¬ solved then
        4.1 for each actor do
            Calculate f
            Calculate Δe
            if e = 0 then
                Remove_from_population
        4.2 Reproduce
        4.3 interval++
```

Actors start the problem-solving process with an initial allotment of energy. After each interval (i.e. domain evaluation), this energy is updated based on how well the actor contributed to the performance of the system during the evaluation (Step 4.1). First, the domain fitness F is converted into domain impact M, i.e. normalized within $[0..1]$ based on max and min fitness values observed in the past R evaluations:

$$M = (F - F_{\min_R})/(F_{\max_R} - F_{\min_R}). \quad (1)$$

Thus, even though F is likely to increase significantly during the problem-solving process, the entire range $[0..1]$ is utilized for M, making it easier to identify promising behavior.

Second, the contribution of the actor to M is measured as the alignment of the actor's domain-action suggestions a_x with the actual action elements A_x issued to the domain during the entire time interval. In the current implementation, this contribution c is

$$c = 1 - \min_{t=0..T}(|A_x(t) - a_x(t)|), \quad (2)$$

where T is the termination time; thus $c \in [0..1]$. The energy update Δe, consists of a fixed cost h and a reward that depends on the impact and the actor's contribution to it. If none of the actor's actions were 'write $a_x(t)$', i.e. the actor did not contribute to the impact,

$$\Delta e = h(M - 1), \quad (3)$$

that is, the energy will decrease inversely proportional to impact. In contrast, if the actor issues one or more such 'write'

actions during the interval,

$$\Delta e = h(cM(1-c)(1-M)-1). \quad (4)$$

In this case, the energy will also decrease (unless M and c are both either 0 or 1) but the relationship is more complex: It decreases less for actors that contribute to good outcomes (i.e. M and c are both high), and for actors that do not contribute to bad outcomes (i.e. the M and c are both low). Thus, regardless of outcomes, each actor receives proper credit for the impact. Overall, energy is a measure of the credit each actor deserves for both leading the system to success as well as keeping it away from failure. If an actor's energy drops to or below zero, the actor is removed from the geo.

For example, if the domain is a reinforcement learning game, like CartPole, each time interval consists of a number of left and right domain actions until the pole drops, or the time limit is reached (e.g. 200 domain time steps). At this point, the domain issues a termination signal, and the fitness F is returned as the number of time steps the pole stayed up. That fitness is scaled to $M \in [0..1]$ using the max and min F during the $R = 60,000$ previous attempts. If M is high, actors that wrote a_x values consistently with A_x, i.e. suggested left or right at least once when those actions were actually issued to the domain, have a high contribution c, and therefore a small decrease Δe. Similarly, if the system did not perform well, actors that suggested left(right) when the system issued right(left), have a low contribution c and receive a small decrease Δe. Otherwise the Δe is large; such actors lose energy fast and are soon eliminated.

After each time interval, a number of new actors are generated through reproduction (Step 4.2). Two parents are selected from the existing population within each (x, y) column, assuming the total energy in the column is below a threshold $E_{\max}$. If it is not, the agents are already very good, and evolution focuses on columns elsewhere where progress can still be made, or alternative solutions can be found. In addition, a parent actor needs to meet a maturity age requirement, i.e. it must have been in the system for more than V time intervals and not reproduced for V time intervals. The actor also needs to have reproduction eligibility in its state set to True.

Provided all the above conditions are met, a proportionate selection process is carried out based on actor fitness f, calculated as follows. First, the impact variable M is discretized into L levels: $M = \{b_0, b_1.., b_{L-1}\}$. Then, for each of these levels b_i, the probability p_i that the actor's action suggestions align with the actual actions when $M = b_i$ is estimated as

$$p_i = P(c = 1 | M = b_i), \quad (5)$$

where c measures this alignment according to Eq. 2. The same window of R past intervals is used for this estimation as for determining the max and min M for scaling the impact values. Finally, actor fitness f is calculated as alignment-weighted average of the different impact levels b_i:

$$f = \sum_{i=0}^{L} p_i b_i. \quad (6)$$

Thus, f is the assignment of credit for M to individual actors. Note that while energy measures consistent performance, actor fitness measures average performance. Energy is thus most useful in discarding actors and actor fitness in selecting parents.

Once the parents are selected, crossover and mutation are used to generate offspring actors. What is crossed over and mutated depends on the encoding of the actor type; regardless, each offspring's behavior, as well as its linked-location coordinates, is a result of crossover and mutation. Each pair of parents generates two offspring, whose location is determined randomly in the same (x, y) column as the parents.

Note that the parents are not removed from the population during reproduction, but instead, energy is used as basis for removal. In this manner, the population can shrink and grow, which is useful for lifelong learning. It allows reproduction to focus on solving the current problem, while removal retains individuals that are useful in the long term. Such populations can better adapt to new problems and re-adapt to old ones.

Energy, age, and actor fitness for all actors in an (x, y) column need to be available before reproduction can be done, so computations within the column must be synchronized in Step 4.2. However, the system is otherwise asynchronous across the x and y dimensions, making it possible to parallelize the computations in Steps 2 and 4. Thereby, the system scales to high-dimensional domains in constant time.

Actor Types

The current version of DIAS employs five different actor types:

- Random: Selects its next action randomly, providing a baseline for the comparisons;
- Robot: Selects its next action based on human-defined preprogrammed rules designed for specific problem domains, providing a performance ceiling;
- Bandit: Selects its next action using a basic multi-armed bandit algorithm (not including $\boldsymbol{\sigma}$ as context);
- Q-Learning: Learns to select its next action using temporal differences; and
- Rule-set Evolution: Evolves a set of rules to select its next action.

Simple Q-learning (Watkins and Dayan, 1992) was implemented based on the actor's state/action history, with the actor's energy difference from the prior time interval taken as the reward for the current interval. Because the dimensionality of the state/action space is limited by design, this method is a possible reinforcement learning approach.

Rule-set evolution (Hodjat et al., 2018) was implemented based on rule sets that consist of a default rule and at least one conditioned rule. Each conditioned rule consists of a conjunction of one or more conditions, and an action that is returned if the conditions are satisfied. Conditions consist of a first and second term being compared, each with a coefficient that is evolved. An argument is also evolved for the action. Evolution selects the terms in the conditions from the actor-state space, and the action from the actor-action space. Rules are evaluated in order, and shortcut upon reaching the first to be satisfied. If none of the rules are satisfied, the default action is returned.

These actor types were evaluated in several standard benchmarks tasks experimentally, as will be described next.

Experiments

DIAS was evaluated in a number of benchmark problems to demonstrate the unique aspects of the approach. The system was shown scalable, general, and adaptable. The dynamics of the problem-solving process were characterized and shown to be the source of these abilities.

Test Domains

In the n-XOR domain, the outputs of n independent XOR gates, each receiving their own input, need to be predicted simultaneously. In order to make the domain a realistic proxy for real-world problems, 10% noise is added to the XOR outputs. While a single XOR (or 1-XOR) problem can be solved by a single actor, solving $n > 1$ of them simultaneously requires a division of labor over the population. The different XOR input elements are in different y-locations and the different predicted outputs in different x-locations. With $n > 1$, no actor can see or act upon the entire problem. Instead, emergent coordination is required to find behaviors that collectively solve all XORs. Increasing n makes the problems exponentially more difficult (i.e. the chance of solving all n XORs by luck is reduced exponentially with n).

The first set of experiments in the n-XOR domain show that the DIAS design scales to problems of different dimensionality and complexity. The second set shows that DIAS can adapt to the different n-XOR problems online, i.e. to exhibit lifelong learning.

Experiments were also run on a number of OpenAI Gym games, including CartPole, MountainCar, Acrobot, and LunarLander. The same experimental setup was used across all of them without any hyperparameter tuning. The OpenAI Gym domains thus show that DIAS is a general problem-solving approach, requiring little or no parameter tuning when applied to new problems.

Experimental Setup

Each experiment consists of 10 independent runs of up to 200,000 time intervals. For each domain, the number of x-locations is set to the number of domain actions, and the number of y-locations to the number of domain states (1, 2 for 1-XOR; 2, 4 for 2-XOR; 3, 6 for 3-XOR; 2, 4 for CartPole; 3, 2 for MountainCar; 3, 6 for Acrobot; and 3, 6 for LunarLander). The number of z-locations is constant at 100 in all experiments. The initial population for each (x, y) location is set to 20 actors, placed randomly in z. Each Q-learning actor is initialized with random Q-values, and each rule-set actor with a random default rule. The robot and bandit actors have no random parameters, i.e. they are all identical.

The range R used for scaling domain fitnesses to impact values was 60,000 intervals, and the impact M was discretized into 21 levels $\{0, 0.05, .., 0.95, 1\}$ in calculating actor fitness. Each actor started with an initial energy of 100 units, with a fixed cost $h = 2$ units at each time interval. The energy threshold $E_{\max}$ for reproduction in each (x, y) column was set to the initial energy, i.e. 20 * 100 = 2000 (note that while each actor's energy decreases over time, population growth can increase total energy). Reproduction eligibility was set to True at birth, and the reproduction maturity requirement V to 20. Small variations to this setup lead to similar results. In contrast, each of the main design choices of DIAS is important for its performance, as verified in extensive preliminary experiments.

Each experiment can result in one of three end states: (1) the actor population solves the problem; (2) all actors run out of energy before solving the problem and the actor population goes extinct; and (3) the actor population survives but has not solved the problem within the maximum number of time intervals. In practice, it is possible to restart the population if it goes extinct or does not make progress in F after a certain period of time. Restarts were not implemented in the experiments in order to evaluate performance more clearly.

For comparison, direct evolution of rule sets (DE) was also implemented in the DIAS framework. The setup is otherwise identical, but a DE actor receives the entire domain state vector $\mathbf{S}$ as its input and generates the entire domain action vector $\mathbf{A}$ as its output. DE therefore does not take advantage of collective problem solving. A population of 100 DE actors is evolved for up to 100,000 time intervals through a GA with F as the individual fitness, tournament selection, 25% elitism, and the same crossover and mutation operators as in DIAS.

Comparing Actor Types

The five actor types described above were each tested in preliminary experiments on 1-XOR, using the same settings. These results demonstrate that collective behavior resulting from the DIAS framework can successfully solve these domains.

The Robot actor specifically written for 1-XOR solves it from the first time interval. Similarly, a custom-designed Robot actor is always successful in the CartPole domain. On the other hand, Random, Bandit, and Simple Q-Learning were not able to solve 1-XOR at all: Each attempt leads to extinction in under 350 time intervals. While it is possible that these

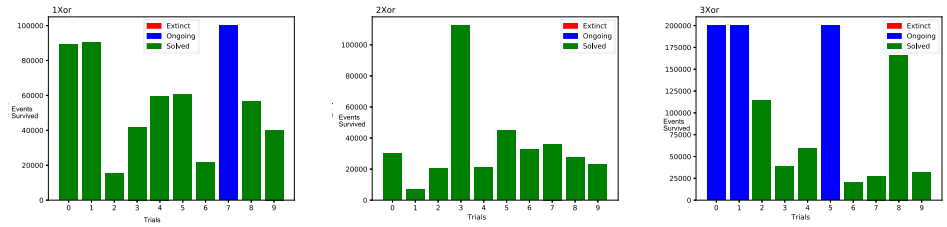


Figure 2: Number of time intervals needed to solve the 1-XOR (left), 2-XOR (middle) and 3-XOR (right) problems in 10 independent runs. No runs lead to extinction (they would have been shown with red bars), though some do not completely solve the problem within the allotted 200,000 time intervals (these runs are shown with blue bars). These experiments show that the DIAS framework scales naturally to problems with increasing dimensionality and complexity.

```
ancestor_count=1065
action counts={decrease_linked_location_y: 55, write: 19}
total_potential_contribution_count=19
total_contribution_count=19
impact_contribution_probabilities=1.0:1.0
state={energy: 66.0; age: 74;
       reproduction_eligibility: True;
       own_location_coordinates 0, 1, 18;
       own_location_message: 0;
       own_location_domain_action: None;
       own_location_domain_state: 0;
       linked_location_coordinates: 0, 1, 19;
       linked_location_message: 0;
       linked_location_domain_action: None;
       linked_location_domain_state: 0;
Rule1<49>: (0.21*y <= 0.62*linked_location_domain_state)
            --> decrease_linked_location_y(0.10)
Rule2<0>:  (0.9*own_location_domain_state
             > 0.15*reproduction_eligibility)&
           (0.21*y <= 0.62*linked_location_domain_state)&
           (0.21*y <  0.62*linked_location_domain_state)
           --> decrease_linked_location_y(0.10)
Rule3<6>:  (0.90*own_location_domain_state
             > 0.15*reproduction_eligibility)
           --> decrease_linked_location_y(0.10)
Rule4<0>:  (0.21*y < 0.62*linked_location_domain_state)
           --> decrease_linked_location_y(0.10)
Deflt<19>: --> write(0.93)
```

Figure 3: An example actor that solves the 1-XOR problem, consisting of a number of metrics, current state, and a set of rules. The 'write' action writes its argument in the own_location_domain_action field as the actor's suggested domain action a_x. Even though the rules explicitly describe the actor's behavior, it is not possible to tell from this one actor what the solution to the complete problem is. The actor does not see the whole problem or determine the outcome alone: The population as a whole collectively solves it.

actors could solve simpler problems, the search space for 1-XOR is apparently already too large for them. Therefore, the main experiments focus on the Rule-set Evolution actor type.

Scaling to Problems of Varying Complexity

The first set of main experiments showed that the DIAS population solves n-XOR with n =1, 2, and 3 reliably (Fig. 2). Even with 10 percent reward noise, the system is resilient and the population collectively achieves the best possible reward, even if it is not constant over time. In comparison, while DE solved the 1-XOR in less than 10,000 time intervals in nine of 10 runs, only three runs solved the 2-XOR and none solved the 3-XOR within 100,000 time intervals. These results show that DIAS provides an advantage in scaling to problems with higher dimensionality and complexity.

The success was due to emergent collaborative behavior of the actor population. This result can be seen by analyzing the rule sets that evolved, for example that of the actor from a population that solved the 1-XOR problem, shown in Fig. 3. This actor is number 1065 in its lineage. It has contributed to the domain action 19 times, and all 19 times, its contribution has been in line with the domain action issued. Therefore, the vector of alignment probabilities p_i at each impact level i has only one element: The probability is 1.0 for the impact level of 1.0. Its current state is high in energy for its age, suggesting that it has contributed well. Its current linked location has null values in message, domain-action, and domain-state fields.

In terms of rules, the second and fourth are redundant, and never fired (redundancy is common in evolution because it makes the search more robust). Rule 1 fired 49 times, Rule 3 six times, and the default rule 19 times. Rules 1 and 3 perform a search for a linked location that has a large enough domain-state value: They decrease the y-coordinate of the linked location whenever they fire. If such a location is found (Rule 1), and its own domain-state value is high enough (Rule 3), 0.93 is written as its suggested domain action a_x (Default rule). An $a_x > 0.5$ denotes a prediction that the XOR output is 1, while $a_x \leq 0.5$ suggests that it is 0; therefore, this actor contributes to predicting XOR output 1. Other actors are required to generate the proper domain actions in other cases. Thus, problem solving is collective: Several actors need to perform compatible subtasks in order to form the whole solution.

Solving Different Kinds of Problems

The second set of main experiments was designed to demonstrate the generality of DIAS, i.e. that it can solve a number of different problems out of the box, with no change to its settings. CartPole, MountainCar, Acrobot, and LunarLander of OpenAI Gym were used in this role because they represent a variety of well-known reinforcement-learning problems.

DIAS was indeed able to solve each of these problems without any customization, and with the same settings as the

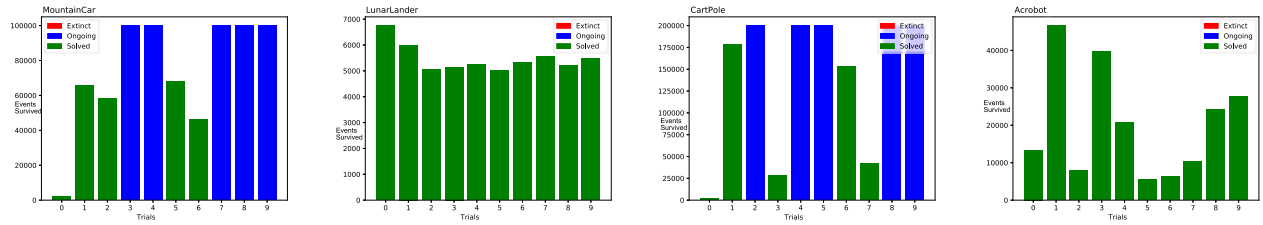


Figure 4: Solving different kinds of problems in the OpenAIGym domain. Results of 10 runs in the MountainCar (left), LunarLander (second from left), CartPole (second from right), and Acrobot (right) problems are shown. Again, no runs resulted in extinction, although some MountainCar and Cartpole runs did not completely solve the problem within the allotted maximum number of time intervals. Notably, DIAS solves all these problems, as well as all other domains in the paper, with the same hyperparameter and experimental settings, demonstrating the generality of the approach.

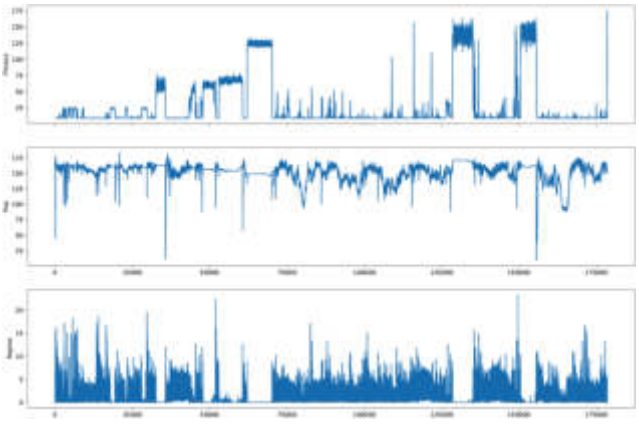


Figure 5: Population dynamics in a sample run of the CartPole problem, showing progression of domain fitness F (the number of time steps the pole stays upright; top), number of live actors (middle), and the number of reproductions (bottom) at each time interval. The reproductions drop and the population becomes relatively stable during periods when the ecosystem finds a peak in F; however, these peaks are unstable and the population eventually moves on to explore other solutions. Such dynamics make it possible to not only find solutions to the current problem, but to also adapt rapidly to changing domains and new problems.

n-XOR problems (Fig. 4). A histogram of the population dynamics as the ecosystem evolves to a solution is shown in Fig. for the CartPole problem. The system gradually finds higher domain fitness peaks, and every time it does so, the number of reproductions drop and the population stabilizes. In this manner, DIAS is trying out different equilibria, eventually finding one that implements the best solution.

Adapting to Changing Problems

A third set of experiments were run in the n-XOR domain to demonstrate the system's ability to switch between domains mid-run. The run starts by solving the 1-XOR problem; then the problem switches to 2-XOR, 3-XOR, and back to 1-XOR again. Note that the max domain fitness level also changes mid-run as problems are switched. These switches require the geo to expand and retract, as the dimension of x (i.e. number of domain actions) and y (number of domain states) are different between problems. This change, however, does not affect the actors, whose action and state spaces remain the same. When retracting, actors in locations that no longer exist are removed from the system. When expanding, new actors are created in locations (i, j, k) with $i > x$ and/or $j > y$ by duplicating the actor in location $(i \bmod x, j \bmod y, k)$, if any.

The results of 10 such runs are shown in Fig. 6. In seven of these runs, DIAS was able to solve the entire sequence of problems. Most interestingly, the time it needed for subsequent problems often became shorter. For example Run 1 took 55,574 time intervals to solve the 1-XOR problem, another 35,363 to solve the 2-XOR, and 36,690 more to solve the 3-XOR. Then, switching back to the 1-XOR problem, a solution was found within a mere 51 time intervals. These results demonstrate that DIAS is able to adapt to new problems quickly, retain information from earlier problems, and utilize it in later problems.

In contrast, while DE solved the 1-XOR fast in the beginning and end of each sequence, none of its 10 runs were able to adapt to 2-XOR and 3-XOR mid-run. Also, it did not solve the second 1-XOR any faster than the first one.

These experiments thus show that the collective problem solving in DIAS is essential for solving new problems continuously as they appear, and for retaining the ability to solve earlier problems. In this sense, it demonstrates an essential ability for continual, or lifelong, learning. It also demonstrates the potential for curriculum learning for more complex problems: The same population can be set to solve domains that get more complex with time. Such an approach may have a better chance of solving the most complex problems than one where they are tackled directly from the beginning.

Discussion and Future Work

The experimental results with DIAS are promising: They demonstrate that the same system, with no hyperparameter tuning or domain-dependent tweaks, can solve a variety of

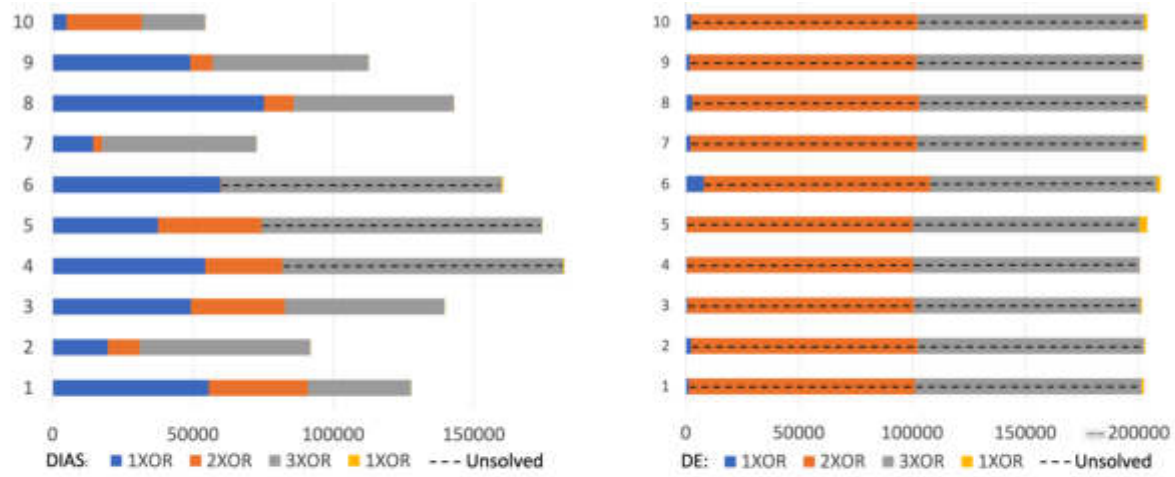


Figure 6: Adapting to changing problems. Ten runs of DIAS (left) and DE (right) are shown where the problem switched from 1-XOR to 2-XOR, 3-XOR, and back to the 1-XOR as soon as the problem was solved or 100,000 time intervals passed (dashed line). DIAS was able to adapt to new problems quickly, solve new problems quicker, and particularly quickly when returning to 1-XOR. In contrast, while DE solved the first 1-XOR quickly, it was not able to adapt to 2-XOR nor 3-XOR mid-run, and it did not solve the second 1-XOR faster than the first. Thus, collective problem solving in DIAS provides a significant advantage in adapting to new problems, i.e. in lifelong learning.

domains, ranging from classification to reinforcement learning. The results also demonstrate ability to switch domains in the middle of the problem-solving process, and potential benefits of doing so as part of curriculum learning. The system is robust to noise, as well as changes to its domain-action space and domain-state space mid-run.

The most important contribution of this work is the introduction of a common mapping between a domain and an ecosystem of actors. This mapping includes a translation of the state and action spaces, as well as a translation of domain rewards to the actors contributing (or not contributing) to a solution. It is this mapping that makes collective problem solving effective in DIAS. With this mapping, changes to the domain have no effect on the survival task that the actors in the ecosystem are solving. As a result, the same DIAS system can solve problems of varying dimensionality and complexity, solve different kinds of problems, and solve new problems as they appear, and do it better than DE can.

In this process, interesting collective behavior analogous to biological ecosystems can be observed. Most problems are being solved through emergent cooperation among actors (i.e. when x and/or y-dimensionality > 1). Problem solving is also continuous: The system regulates its population, stabilizing it as better solutions are found. Because of this cooperative and continual adaptation, it is difficult to compare the experimental results to those of other learning systems. Solving problems of varying scales, different problems, and tracking changes in the domain generally requires domain-dependent set up, discovered through manual trial and error. A compelling direction for the future is to design benchmarks for domain-independent learning, making such comparisons possible and encouraging further work in this area.

In the future, a parallel implementation of DIAS should speed up and scale up problem-solving. It would be possible to run DIAS with larger search spaces in reasonable time. For high-dimensional domain-state and domain-action spaces, it may also be possible to fold the axes of the geo so that a single (x, y) location can refer to more than one state or action in the domain space. This generalization, of course, would come at the expense of larger actor-action and actor-state space because each location would now have more than one value for domain state and action, but it could make it faster with high-dimensional domains. Another potential improvement is to design more actor types. While rule-set evolution performed well, it is a very general method, and it may be possible to design other methods that more rapidly and consistently adapt to specific problem domains as part of the DIAS framework.

Conclusion

DIAS is a domain-independent problem-solving system that can address problems with varying dimensionality and complexity, solve different problems with little or no hyperparameter tuning, and adapt to changes in the domain, thus implementing lifelong learning. These abilities are based on artificial-life principles, i.e. collective behavior of a population of actors in a spatially organized geo, which forms a domain-independent problem-solving medium. Experiments with DIAS demonstrate an advantage over a direct problem-solving approach, thus providing a promising foundation for scalable, general, and adaptive problem solving in the future.

References

Ackley, D. and Small, T. (2014). Indefinitely scalable computing= artificial life engineering. In *ALIFE 14: The Fourteenth International Conference on the Synthesis and Simulation of Living Systems*, pages 606–613. MIT Press.

Bansal, J. C., Singh, P. K., and Pal, N. R. (2019). *Evolutionary and swarm intelligence algorithms*, volume 779. Springer.

Della Penna, G., Magazzeni, D., Mercorio, F., and Intrigila, B. (2009). UPMurphi: A tool for universal planning on PDDL+ problems. In *Proc. International Conference on Automated Planning and Scheduling*. 19:106–113.

Deng, W., Xu, J., and Zhao, H. (2019). An improved ant colony optimization algorithm based on hybrid strategies for scheduling problem. *IEEE Access*, 7:20281–20292.

Eiben, A. E. and Smith, J. E. (2015). *Introduction to Evolutionary Computing*. Springer.

Gershenson, C., Trianni, V., Werfel, J., and Sayama, H. (2018). Self-organization and artificial life: A review. *arXiv:1804.01144*.

Hodjat, B. and Shahrzad, H. (1994). Introducing a dynamic problem solving scheme based on a learning algorithm in artificial life environments. *Proceedings of 1994 IEEE International Conference on Neural Networks*, 4:2333–2338.

Hodjat, B., Shahrzad, H., Miikkulainen, R., Murray, L., and Holmes, C. (2018). PRETSL: Distributed probabilistic rule evolution for time-series classification. In *Genetic Programming Theory and Practice XIV*, pages 139–148. Springer.

Hutter, M. (2000). A theory of universal artificial intelligence based on algorithmic complexity. *arXiv:cs-ai-0004001*.

Krejca, M. S. and Witt, C. (2020). Theory of estimation-of-distribution algorithms. In *Theory of Evolutionary Computation*, pages 405–442. Springer.

Mitchell, M. (1996). *An Introduction to Genetic Algorithms*. MIT Press, Cambridge, MA.

Rodriguez, A. and Reggia, J. A. (2004). Extending Self-Organizing Particle Systems to Problem Solving. *Artif. Life*, 10:379–395.

Sengupta, S., Basak, S., and Peters, R. (2018). Particle swarm optimization: A survey of historical and recent developments with hybridization perspectives. *Machine Learning and Knowledge Extraction*, 1(1):157–191.

Stern, R. (2019). Domain-dependent and domain-independent problem solving techniques. In *IJCAI*, pages 6411–6415.

Watkins, C. J. and Dayan, P. (1992). Q-learning. *Machine learning*, 8(3):279–292.

Toward automatic generation of diverse congestion control algorithms through co-evolution with simulation environments

Teruto Endo, Hirotake Abe, Mizuki Oka

University of Tsukuba, Tsukuba, Ibaraki 305-8573 Japan

mizuki@cs.tsukuba.ac.jp

Abstract

Congestion control algorithms are used to help prevent congestion from occurring on the Internet. However, a definitive congestion control algorithm has yet to be developed. There are three reasons for this: First, the environment and usage of the Internet continue to evolve over time. Second, it is not clear what congestion control algorithms will be required as the environment evolves. Third, there is a limit to the number of the congestion control algorithms that can be developed by researchers. This paper proposes a method for automatically generating diverse congestion control algorithms and optimizing them in various environments by co-evolving network simulations as environments and congestion control algorithms as agents. In experiments conducted using co-evolution, although the algorithms generated were not on par with conventional practical congestion control algorithms, the intent of the procedures in the algorithms was interpretable from a human perspective. Furthermore, our results verify that it is possible to automatically discover a suitable environment for the evolution of a congestion control algorithm.

Introduction

Congestion is the excessive temporal flow of data over a network. On the Internet, many users constantly compete for communication bandwidth. This causes congestion, which may result in reduced communication speed and connection problems for the service. To prevent Internet congestion, congestion control algorithms are used to adjust the data transmission rate.

A congestion control algorithm determines whether congestion is occurring and accordingly decides whether to increase or decrease the amount of data sent. The two main types of congestion control algorithms are loss-based and delay-based algorithms (Al-Saadi et al., 2019). Loss-based algorithms determine whether congestion is occurring based on the packet loss, whereas delay-based algorithms make the determination based on the round-trip time of packets.

Research on congestion control algorithms has been ongoing for over 40 years; however, no definitive congestion control algorithm has yet been developed. This is because Internet usage and connection patterns change over time, and it is unclear what algorithms are necessary when the Internet structure or bandwidth drastically changes. Existing congestion control algorithms are built using a rule-based approach, which makes it difficult for them to respond to changes in the environment (Li et al., 2019). In addition, the number of the congestion control algorithms that can be developed by researchers and the number of environments in which they can be tested are limited.

One possible approach to address these issues is to use program synthesis to automatically build algorithms without human intervention (Gulwani et al., 2017). In this regard, an interesting study was conducted by Lones (2020), who used genetic programming to automatically generate an optimization algorithm. The study highlights the possibility that only a limited portion of the design space of optimization algorithms has been explored because of biases in human thinking. In fact, by using genetic programming, they succeeded in generating an algorithm that has both superior performance and different characteristics from existing human-made optimization algorithms. A similar challenge exists for congestion control algorithms.

In machine learning, which has made remarkable progress as represented by deep learning, a method called AutoML-Zero has been proposed to build algorithms from scratch using evolutionary methods (Real et al., 2020). Although AutoML-Zero has not yet discovered any new valuable algorithms, it has rediscovered useful machine learning methods, such as backpropagation. This result, which automatically builds on existing methods from scratch, suggests the possibility of automatically discovering useful undiscovered algorithms in the future.

This attempt to automatically generate algorithms using evolutionary methods without human intervention has the potential to devise algorithms that have not yet been discovered. This indicates that there may be useful algorithms existing outside the space already heuristically explored by humans. Adding diversity to the solution, rather than simply searching for a single optimal solution, will lead to the discovery of a global optimal solution. Recently, an algorithm called the quality–diversity algorithm, which attempts

to improve both the quality of the solution (the fitness of the solution) and the diversity of the solution (the types of solutions), has been studied (Pugh et al., 2016). The results show that improving the diversity of solutions rather than searching for a single optimal value can lead to the discovery of better solutions.

The paired open-ended trailblazer (POET) (Wang et al., 2019), which co-evolves the agent and environment, has been attracting attention as a further development of the quality–diversity algorithm. POET learns by pairing an agent with an environment generated by mutation, and optimizes the agent for each paired environment. In addition, high-performance agents are transferred to other environments. In this manner, a new environment is automatically generated at the appropriate time, and agents are optimized in the new environment as well to discover solutions (agents) to unknown problems (environments). By co-evolving agents and environments, we can obtain useful environments and agents that cannot be obtained without co-evolution. However, most applications of the quality–diversity algorithm are in games and robotics. Therefore, further development can be expected by demonstrating that it can be applied to other fields.

Therefore, this study examines congestion control algorithms as a new application of the quality–diversity algorithm and agent–environment co-evolution. As no attempt has been made to automatically generate congestion control algorithms using evolutionary algorithms, it is hoped that the application of the quality–diversity algorithm and POET to the automatic generation of congestion control algorithms will lead to the discovery of new and useful congestion control algorithms. Specifically, we aim to automatically construct a variety of congestion control algorithms and simulation environments without human intervention.

Agent and Environment

In this study, the congestion control algorithms are the agents, and the network simulations are the environments. This section describes how the congestion control algorithms are generated and how the environments are controlled.

Controlling the environments

To co-evolve environments and agents, it is necessary to control the difficulty of the environments. In this study, the throughput is used as an indicator of the difficulty of the environment. It is necessary to design parameters such that the higher the throughput, the lower the difficulty; conversely, the lower the throughput, the higher the difficulty. Therefore, we use two types of parameters to control the difficulty level: the amount of cross-traffic generated and the packet loss rate.

Cross-traffic is traffic that interferes with the main communication. For example, it communicates on a communication path that overlaps with the communication path between the main server and clients. When the communication paths overlap, it puts pressure on the bandwidth. In such a case, if the communication data are not properly controlled, congestion will occur and throughput will decrease. Therefore, the amount of cross-traffic generated is varied to control the difficulty of the environment.

There are two ways to control the amount of cross-traffic generated. The first is to vary the amount of data communication between the server and client that generates the cross-traffic, and the second is to vary the number of nodes that generate the cross-traffic.

Figure 1 is a conceptual diagram of a network topology in which cross-traffic can occur. There are three types of nodes: servers, clients, and relay nodes. Relay nodes only relay data between main traffic and cross-traffic, and do not generate any traffic. The red nodes indicate nodes that generate main traffic, and the blue nodes indicate nodes that generate cross-traffic. The server that measures the throughput is called the main server. The client that sends data to the main server is called the main client.

When the second parameter, packet loss, occurs, data must be retransmitted and received. Consequently, it takes time to retransmit packets, resulting in a decrease in throughput. Packet loss can be caused by disconnection in the case of wired networks or radio interference in the case of wireless networks. This is particularly likely to occur in wireless applications. Therefore, packet loss rate is considered an effective parameter for controlling the difficulty of the environment. Packet loss during an experiment occurs at all nodes in the network topology.

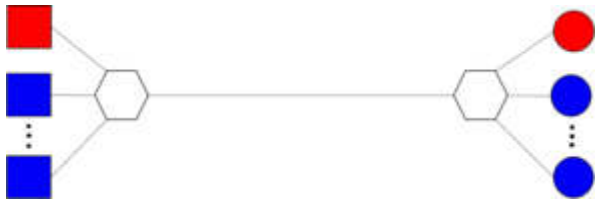


Figure 1: A network topology in which cross-traffic can occur.

We use ns3gym (Gawłowicz and Zubow, 2019) for network simulation, which is our environment. ns3gym is a framework that enables ns3 [1], a network simulator often used in network research, to be used in OpenAI Gym (Brockman et al., 2016), which is a library that provides an environment for reinforcement learning.

Basic congestion control algorithms

To automatically generate a congestion control algorithm, it is necessary to determine the base of the generated algorithm. We use the additive increase and multiplicative

[1] https://www.nsnam.org/

decrease (AIMD) algorithm, a relatively simple but widely used classical congestion control algorithm, as a reference to create the base algorithm (Peterson and Davie, 2011).

The AIMD algorithm can be implemented in a relatively simple manner. Therefore, instead of using complex congestion control algorithms, this research also uses simple basic arithmetic operations and conditional branching, such as AIMD, as the basis for the design of the algorithms.

For AIMD, the congestion window size is determined by Equation 1.

$$x(t+1) = \begin{cases} x(t) + a & (\text{no congestion detected}) \\ x(t) \times b & (\text{congestion detected}), \end{cases} \quad (1)$$

where $x(t)$ represents the congestion window size at time t. AIMD is a very simple method that increases the congestion window size by addition and decreases it by multiplication. If congestion is not detected, the congestion window size is increased via parameter a ($a > 0$). If congestion is detected, the congestion window size is decreased via parameter b ($0 < b < 1$).

Many algorithms using the AIMD scheme have been proposed, but one of the most popular is NewReno (Gurtov et al., 2012). NewReno increases the congestion window size linearly if no congestion is detected and halves the congestion window size if congestion is detected. The formula for updating the congestion window size for NewReno is given by Equation 2.

$$x(t+1) = \begin{cases} x(t) + 1 & (\text{no congestion detected}) \\ \frac{x(t)}{2} & (\text{congestion detected}) \end{cases} \quad (2)$$

Automatic generation of congestion control algorithms

We use grammatical evolution (O'Neill and Ryan, 2001) to generate the congestion control algorithms. Grammatical evolution allows for a relatively free design of the generated code and functions, and because the generated code is grammatically guaranteed, it can be naturally applied to implementations that require conditional branching, such as congestion control algorithms. To run the grammatical evolution, we use PonyGE2 (Fenton et al., 2017), a Python 3 implementation.

The fitness is defined by the average throughput, as shown in Equation 3.

$$Fitness = \frac{1}{N} \sum_{t} throughput_t, \quad (3)$$

where $throughput_t$ represents the throughput measured at time t and N represents the number of times the throughput is measured. The throughput is measured using the main traffic server at regular intervals. The server records the amount of data received at regular intervals and calculates it using Equation 4.

$$throughput_t = \frac{(B_t - B_{t-interval}) \times 8}{interval} \, [bps], \quad (4)$$

where B_t indicates the number of bytes received by the server at time t and $interval$ indicates the interval in seconds at which the server records the amount of data received. It is multiplied by eight to convert the byte unit to bit unit. By maximizing the fitness, it is possible to develop the congestion control algorithm with a higher throughput.

Figure 2 illustrates the grammar used to generate the congestion control algorithm. The grammar is defined such that it can generate an algorithm that can be implemented using four arithmetic operations and conditional branching, such as AIMD. The "obs" in the grammar indicates the information observed during the simulation. The "OPEN," "DISORDER," "RECOVER," and "LOSS" in lines 1 through 7 of the grammar represent the following four congestion states:

- OPEN: Normal state
- DISORDER: State in which a duplicate ACK is received
- RECOVER: Duplicate ACK is received three times (stronger suspicion of congestion occurring)
- LOSS: State in which ACK timeout is detected (transmission is lost owing to congestion)

"OPEN" indicates that congestion is not present, "DISORDER" and "RECOVER" indicate that congestion may be present, and "LOSS" indicates that communication has been lost owing to congestion. In addition, ACK is a packet that indicates to the data sender that the data were received correctly. The grammar shown in Figure 2 defines four methods for updating the congestion window size (new_cwnd) for the four different congestion states. These update methods are defined by the observational information of the environment, the four arithmetic operators, and conditional branching. The larger the value of the congestion window size, the more data that can be transmitted. To improve the fitness, we need to evolve the algorithm to adjust the congestion window size to prevent congestion and communicate more data.

Co-evolution of simulation environments and algorithms

Our aim is to automatically generate congestion control algorithms in a variety of simulation environments through co-evolution. We use POET to co-evolve the simulation environments and congestion control algorithms. POET is characterized by the fact that it generates new environments that are appropriate for the evolution of individuals. In addition to the automatic generation of new algorithms, POET is expected to automatically generate environments that are optimal for evolution, and discover environments that will generate interesting individuals.

To use POET, we need to determine the minimum and maximum values of the fitness that will be used as a criterion to decide whether or not to perform mutation of the environment. It is necessary to set a criterion value that is

```
<algorithm> ::= if state == OPEN:
                    <code1>
                elif state == DISORDER:
                    <code2>
                elif state == RECOVER:
                    <code2>
                elif state == LOSS:
                    <code2>
                else:
                    <code2>
<code1> ::= new_cwnd = <update>
          | if <condition>:
                new_cwnd = <update>
            else:
                new_cwnd = <update>
<code2> ::= new_cwnd = <update>
            new_ssthresh = <update>
          | if <condition>:
                new_cwnd = <update>
                new_ssthresh = <update>
            else:
                new_cwnd = <update>
                new_ssthresh = <update>
<condition> ::= obs[<obs_index>]<comp_op>obs
  [<obs_index>]
<update> ::= <update><arith_op><update>
           | obs[<obs_index>]
           | <num>
<obs_index> ::=
  0|1|2|3|4|5|6|7|8|9|10|11|13|
<comp_op> ::= <|>|<=|>=|==|!=
<arith_op> ::= +|-|*|/|%
<num> ::= 1|2|3|4|5|6|7|8|9
```

Figure 2: A grammar for generating congestion control algorithms in Grammatical Evolution

neither too easy nor too difficult for the agent. However, it is difficult to determine criterion values in advance. Therefore, we set 10% of the bandwidth in the experiment as the minimum value and 80% as the maximum value. In addition, the search space for combinations of algorithms that can be generated by grammatical evolution and environments that can be generated by network simulation is huge. To reduce the search space, we first evolve a congestion control algorithm using a quality–diversity algorithm and use it in the initial population set of POET.

For the quality–diversity algorithm, we use CMA-ME (Fontaine et al., 2020), coupled with an improvement emitter to improve the fitness of the individuals. Thus, individuals are generated by CMA-ME, and the genetic information of the generated individuals is replaced with code using the grammar defined by BNF. In the quality–diversity algorithm, it is necessary to define a behavior descriptor (BD) that represents the characteristics of the solution. We use the mean and standard deviation of the congestion window size as the BDs. The mean value of the congestion window size is an indicator of the trend in the size of the congestion window set by the algorithm. The standard deviation of the congestion window size is an indicator of the degree of variation in the congestion window size. For example, if the mean value of the congestion window size is high and the standard deviation is low, the algorithm is considered to be a congestion control algorithm with the characteristic of selecting a consistently large congestion window size. We used the pyribs (Tjanaka et al., 2021) library to run CMA-ME.

Experimental Evaluation

We conducted two experiments to evaluate the feasibility of the proposed method.

First, we evolved congestion control algorithms using the quality–diversity algorithm for use as the initial population set for co-evolution. We analyzed the generated set of algorithms, the characteristics of the algorithms with a high degree of fitness, and the algorithms that are frequently generated. Next, we examined whether useful simulation environments and congestion control algorithms could be generated automatically through co-evolution using POET.

Automatic generation experiments using the Quality–Diversity algorithm

We automatically generated congestion control algorithms by combining CMA-ME and grammatical evolution. The simulation environment was configured as follows: a dumbbell-shaped network topology with two nodes at each end, as shown in Figure 1; a packet loss rate of 0.0; a bandwidth of 10 Mbps; a delay of 45 s; and a cross-traffic data transmission rate of 6 Mbps. The simulation time was set to 30 s.

The mean (hereinafter referred to as BD1) and standard deviation (denoted BD2) of the congestion window size were used as the BD for CMA-ME. Because BD needs to be discretized in CMA-ME, the minimum and maximum values of BD1 were set to 0 and 3×10^5, the maximum and minimum values of BD2 were set to 0 and 1.5×10^5, and BD1 and BD2 were divided into 50 equally spaced parts. In the experiment, 48 individuals were generated in each iteration of CMA-ME, and 5600 iterations were performed.

Analysis of the relationship between behavior descriptor and fitness

We analyzed the relationship between the BDs and fitness to determine which features of the algorithm adapted to the environment. A heat map showing the relationship between the BDs and fitness is given in Figure 3. The horizontal axis shows the mean value of the congestion window size, which is the first BD, and the vertical axis shows the standard deviation of the congestion window size, which is the second BD. The fitness is expressed by the brightness of the color,

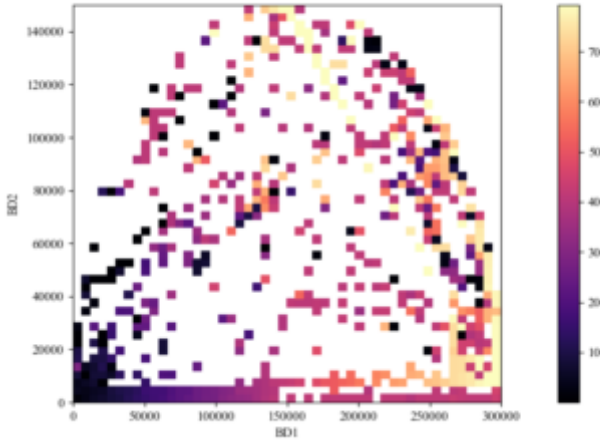


Figure 3: Heat map showing the relationship between BD and fitness (horizontal axis (BD1): mean value of the congestion window size; vertical axis (BD2): standard deviation of the congestion window size).

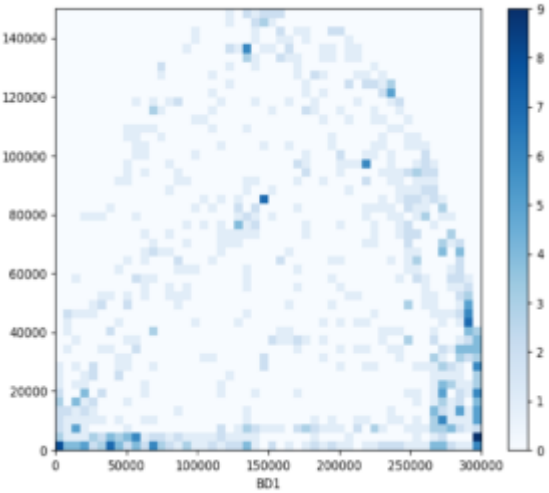


Figure 4: Heat map showing the relationship between BDs and the number of updates for each cell (horizontal axis (BD1): mean value of the congestion window size; vertical axis (BD2): standard deviation of the congestion window size).

with lighter colors indicating higher adaptation and darker colors indicating lower adaptation.

Considering the regions with high adaptability, we find that algorithms with high adaptability are distributed on the lower righthand side of the heat map and the upper center of the heat map. The lower righthand side of the heat map shows the area where the congestion window size tends to be always large. The upper center of the heat map is a region where the value of the congestion window size frequently switches between the upper limit of the congestion window size and zero. This indicates that in the environment, setting the congestion window size to a large value results in high throughput.

Analysis of the number of updates for each cell in the archive

Next, we visualized the relationship between the number of updates of each cell in the archive and BDs and analyzed the characteristics of the individuals that were likely to be generated. A heat map showing the relationship between the number of updates for each cell and the BDs is shown in Figure 4. The horizontal and vertical axes are the same as those shown in Figure 3. The number of updates is represented by the brightness of the color, with brighter colors indicating fewer updates and darker colors indicating more updates.

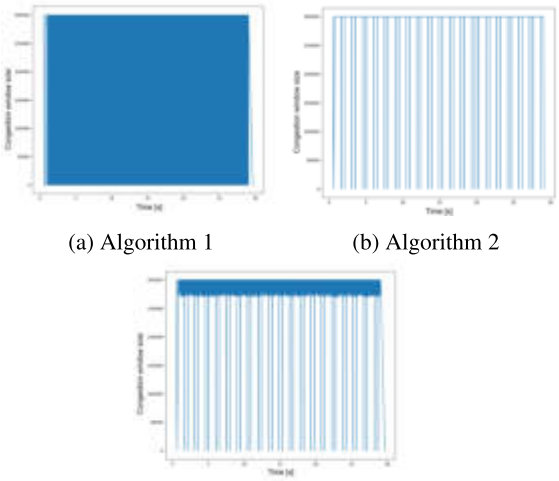
The regions with the highest number of updates are identified in the lower right and lower left of the heat map. These are the areas where algorithms that barely change the size of the congestion window are distributed. In other words, the most frequent updates were for simple algorithms that did not significantly change the congestion window size. This result indicates that relatively simple algorithms tend to be generated.

Analysis of generated algorithms

We investigated how algorithms with a high fitness value control congestion window size. First, we visualized the time variation of the congestion window size using the generated algorithms. Among the generated algorithms, the top three individuals with the highest fitness were Algorithms 1, 2, and 3, and the time variation of their congestion window size is shown in Figures 5a, 5b, and 5c, respectively.

Note that the fitness of all of these algorithms was 7,928 kbps. Algorithm 1 (Figure 5a) sets the congestion window size alternately between the upper limit and zero within a very short time interval. Algorithm 2 (Figure 5b) sets the upper limit and zero for the congestion window size to longer time intervals than those in Algorithm 1. Algorithm 3 (Figure 5c) varies the congestion window size between 250,000 and 300,000, while alternating between the upper limit and zero.

We verified in detail the control method of Algorithm 3 (Fig. 5c) from the generated code because it had a more complicated control method than the other two algorithms. Part of the code for Algorithm 3 is shown in Figure 6. There are two noteworthy points. The first is the setting of the congestion window size when a duplicate ACK is received in Line 2. Here, the congestion window size was set to the current slow-start threshold (obs[4] in the code). The second is



(a) Algorithm 1 (b) Algorithm 2

(c) Algorithm 3

Figure 5: Time variation of congestion window size for the three algorithms with the highest fitness value.

the setting of the slow-start threshold when a duplicate ACK is received three times in Line 6. Here, the new slow-start threshold was set to a value that was manipulated from the current slow-start threshold. Such processing has also been observed in existing algorithms. From these findings, it is clear that part of the operation of the congestion control algorithm in Algorithm 3 is processed in a human-explainable manner. Therefore, we decided to include Algorithm 3 in the initial population set of the co-evolution.

Co-evolution of simulation environments and congestion control algorithms

We used POET to co-evolve a network simulation environment and a congestion control algorithm to verify whether a useful simulation environment and congestion control algorithm can be generated automatically. To increase the efficiency of evolution, we initialized half of the individuals using Algorithm 3 (Figure 5c) and the other half randomly.

A dumbbell network topology was used for the network simulation environment, similar to the experiments with the quality–diversity algorithm. We used the number of dumbbell node pairs, packet loss rate, and cross-traffic data transmission rate as three parameters to control the environment. The possible values to be selected for each parameter are as follows: 2, 3, 4, or 5 number of node pairs, packet loss rates at 0.0, 0.001, 0.01, or 0.1, and data transmission rates 6, 7, 8, or 9 Mbps. We set the number of dumbbell node pairs in the initial environment to two, the packet loss ratio to 0.0, and the cross-traffic data transmission rate to 6 Mbps. As parameters common to all environments, we set the bandwidth to 10 Mbps and delay to 45 ms. Environmental mutations were performed every 30 iterations and individual transfers every 10 iterations, for a final total of 303 iterations.

```
1 elif state == DISORDER:
2     new_cWnd = obs[4]
3     new_ssThresh = 5
4 elif state == RECOVER:
5     new_cWnd = obs[8]
6     new_ssThresh = 5 * obs[4]
```

Figure 6: Part of the code for Algorithm 3.

Results of co-evolution experiments

Fourteen pairs of network simulator environments and congestion control algorithms were generated. Among the environments, we found an environment that appeared to have prompted the evolution of the congestion control algorithm. The environment was generated during the 239th POET iteration and had the following parameters: packet loss rate, 0.001; dumbbell node pair count, 4; cross-traffic data transmission rate, 8 Mbps (this environment is referred to as Environment A). A graph showing the highest fitness of the algorithm for each generation is represented by the blue line in Figure 7. It can be observed that the fitness gradually increased with each generation. However, except for the pair of Environment A, no increase in fitness was observed and the fitness was constant from the initial generation. These results suggest that an environment with a packet loss rate of 0.001, dumbbell node pair count of 4, and cross-traffic data transmission rate of 8 Mbps is conducive to the evolution of congestion control algorithms.

However, we had to ascertain whether the set of algorithms in environment A changed fitness in other environments. Therefore, we evaluated the algorithms generated in Environment A in other environments to verify whether Environment A promoted evolution. The evaluation environment was not the 10 Mbps bandwidth used up to this point, but a bandwidth of 1 Gbps, assuming the current Internet environment. The other environmental parameters were identical to those of Environment A. Owing to the time constraints of the experiment, we evaluated the algorithms for the generations (1, 2, 11, 12, 57, 58, 61, and 62) in which there were changes in fitness in the 10 Mbps environment. Another purpose of the evaluation in a 1 Gbps bandwidth environment was to verify that the generated algorithms have high fitness in a modern environment.

Looking at the change in the highest value of the fitness of the algorithm generated in environment A at 1 Gbps, as shown by the orange line in Figure 7, there was no change in the fitness observed in environment A in the 1 Gbps environment. In other words, the differences in congestion control algorithms as captured in Environment A were not captured in the 1 Gbps environment. An environment in which differences in algorithms cannot be evaluated is inappropriate for algorithm evolution. This suggests that environment A is suitable for the evolution of paired algorithms.

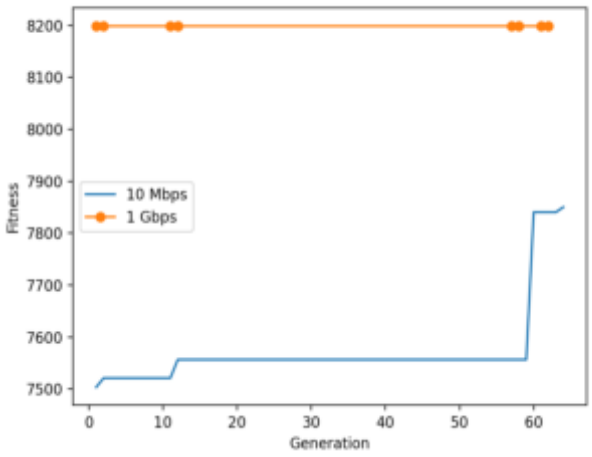


Figure 7: Change in maximum fitness for each generation in environments with bandwidths of 10 Mbps (blue line) and 1 Gbps (orange line). The environmental parameter settings were as follows: packet loss ratio of 0.001, 4 pairs of dumbbell nodes, and cross-traffic data transfer rate of 8 Mbps. (Owing to experimental time constraints, only the generation of algorithms that showed a change in fitness at 10 Mbps was evaluated at 1 Gbps.)

However, no congestion control algorithm was generated in Environment A, which performed interesting processing. The generated congestion control algorithm is a congestion control algorithm with very simple control, similar to the algorithm generated using CMA-ME. For example, algorithms that alternately select an upper limit and zero for the congestion window size (Figure5a, 5b) or algorithms that only set a small congestion window size. However, from Figure7, it is possible that further evolution would produce a congestion control algorithm that performs interesting processing, because the fitness increase was observed even after the last 60 generations.

Discussion

In this study, we were not able to automatically generate a useful congestion control algorithm that is as good as or better than existing algorithms. There are two possible challenges in this approach.

The first is the need to re-examine grammar. In the current grammar, using the quality–diversity algorithm and POET, many of the algorithms generated were very simple. As can be seen in Figure 4, which shows the number of updates for each cell in the BD space in the quality–diversity algorithm, many congestion control algorithms were generated that set very small or large values for the congestion window size. From this, it can be said that the current grammar easily generates simple congestion control algorithms. By improving the grammar, we believe that it will be possible to generate useful congestion control algorithms with a higher probability. One way to improve the grammar is to make the patterns of programs that can be generated more limited than they are now.

Another issue is that the fitness used in this study may be insufficient for evaluating the performance of the algorithm. It is clear from the results that Algorithms 1, 2, and 3 each perform different congestion control. However, the average throughput, which indicates the fitness of these three algorithms, was 7,928 kbps for all three algorithms. Given that the bandwidth of the experimental environment was 10 Mbps, this means that all algorithms performed very well. In other words, it is unlikely that the fitness could be improved further. This is undesirable from the perspective of the objective of this study, which is the automatic generation of new congestion control algorithms. This is because the current index shows high fitness for simple algorithms, which may have a negative impact on the evolution of more complex algorithms, such as existing congestion control algorithms. We believe that it is possible to facilitate the generation of algorithms with the same complexity as existing congestion control algorithms by defining adaptations that can differentiate Algorithm 3, which is a relatively complex control compared to Algorithms 1 and 2.

Conclusion

Using the quality–diversity algorithm, we generated an algorithm that, while not as good as existing congestion control algorithms, can be interpreted by a human observer as to the intent of the process. This provides a foundation for the automatic generation of new and useful congestion control algorithms.

We also showed that by co-evolving the congestion control algorithm and the network simulation environment, it is possible to automatically discover useful environments for evolving the congestion control algorithm. This result indicates that a single environment alone is insufficient to promote evolution and suggests the effectiveness of the agent–environment co-evolution method.

In future work, we will examine the definition of fitness and improve the grammar used to generate congestion control algorithms. Then, we will perform automatic generation of congestion control algorithms to verify whether useful algorithms can be generated. Moreover, one of the advantages of environment generation in co-evolution is the possibility of automatically discovering environments that can yield useful insights. One environmental control parameter that may provide useful insights is network topology. The dumbbell network topology has been used frequently for many years in congestion control research because it is a simplified representation of a real-world situation where congestion is likely to occur. However, the dumbbell network topology is designed by humans, and there is no guarantee that there is

no network topology that can provide useful knowledge that has not yet been discovered. Therefore, we believe that the use of various automatically generated network topologies as environments may provide new insights.

References

Al-Saadi, R., Armitage, G., But, J., and Branch, P. (2019). A survey of delay-based and hybrid tcp congestion control algorithms. *IEEE Communications Surveys Tutorials*, 21(4):3609–3638.

Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., and Zaremba, W. (2016). Openai gym. *CoRR*.

Fenton, M., McDermott, J., Fagan, D., Forstenlechner, S., Hemberg, E., and O'Neill, M. (2017). Ponyge2: Grammatical evolution in python. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, pages 1194–1201.

Fontaine, M. C., Togelius, J., Nikolaidis, S., and Hoover, A. K. (2020). Covariance matrix adaptation for the rapid illumination of behavior space. In *Proceedings of the 2020 Genetic and Evolutionary Computation Conference*, GECCO '20, page 94–102, New York, NY, USA. Association for Computing Machinery.

Gawłowicz, P. and Zubow, A. (2019). Ns-3 meets openai gym: The playground for machine learning in networking research. In *ACM International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems (MSWiM)*, pages 113—120.

Gulwani, S., Polozov, O., and Singh, R. (2017). Program synthesis. *Foundations and Trends® in Programming Languages*, 4(1-2):1–119.

Gurtov, A., Henderson, T., Floyd, S., and Nishida, Y. (2012). The NewReno Modification to TCP's Fast Recovery Algorithm. RFC 6582.

Li, W., Zhou, F., Chowdhury, K. R., and Meleis, W. (2019). Qtcp: Adaptive congestion control with reinforcement learning. *IEEE Transactions on Network Science and Engineering*, 6(3):445–458.

Lones, M. A. (2020). Optimising optimisers with push gp. *Genetic Programming. EuroGP 2020. Lecture Notes in Computer Science*.

O'Neill, M. and Ryan, C. (2001). Grammatical evolution. *IEEE Transactions on Evolutionary Computation*, 5(4):349–358.

Peterson, L. L. and Davie, B. S. (2011). *Computer Networks, Fifth Edition: A Systems Approach*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 5th edition.

Pugh, J. K., Soros, L. B., and Stanley, K. O. (2016). Quality diversity: A new frontier for evolutionary computation. *Frontiers in Robotics and AI*, 3:40.

Real, E., Liang, C., So, D. R., and Le, Q. V. (2020). Automl-zero: Evolving machine learning algorithms from scratch. *arXiv preprint arXiv:2003.03384*.

Tjanaka, B., Fontaine, M. C., Zhang, Y., Sommerer, S., Dennler, N., and Nikolaidis, S. (2021). pyribs: A bare-bones python library for quality diversity optimization. https://github.com/icaros-usc/pyribs.

Wang, R., Lehman, J., Clune, J., and Stanley, K. O. (2019). Poet: Open-ended coevolution of environments and their optimized solutions. In *Proceedings of the Genetic and Evolutionary Computation Conference*, GECCO '19, page 142–151, New York, NY, USA. Association for Computing Machinery.

Towards an FPGA Accelerator for Markov Brains

Arend Hintze[2,3] and Jory Schossau[1,2]

[1]Michigan State University, Department of Computer Science and Engineering, East Lansing, 48824, United States of America
[2]Michigan State University, BEACON Center for the Study of Evolution in Action, East Lansing, 48824, United States of America
[3]Dalarna University, Department of MicroData Analytics, Dalarna, 79188, Sweden
ahz@du.se

Abstract

The success of deep learning is to some degree based on our ability to train models quickly using GPU or TPU hardware accelerators. Markov Brains, which are also a form of neural networks, could benefit from such an acceleration as well. However, Markov Brains are optimized using genetic algorithms, which present an even higher demand on the acceleration hardware: Not only inputs to the network and its outputs need to be communicated but new network configurations have to be loaded and tested repeatedly in large numbers. FPGAs are a natural substrate to implement Markov Brains, who are already made from deterministic logic gates. Here a Markov Brain hardware accelerator is implemented and tested, showing that Markov Brains can be computed within a single clock cycle, the ultimate hardware acceleration. However, how current FPGA design and supporting development toolchains are limiting factors, and if there is a future size speed trade-off are explored here as well.

Introduction

Deep neural networks (DNN) together with deep learning are the most common technologies for the study and implementation of complex automation in the field of artificial intelligence. A DNN is usually optimized using backpropagation, but DNN have a predefined network structure. A more flexible option is neuroevolution that has been shown to be competitive to deep learning (Such et al., 2018; Real et al., 2020). A specific architecture amenable to neuroevolution is Markov Brains (MBs) (Marstaller et al., 2013) that is a form of recurrent neural network — similar to Cartesian Genetic Programming (Miller and Harding, 2009) — where computational units read from inputs and hidden states, and write their computational results into hidden states and outputs. Even the computational functions of MBs are not predefined, but optimized using a genetic algorithm (Hintze et al., 2017) . Markov Brains are particularly well-suited for embodied agent-based control and navigation tasks (Kvam et al., 2015; Edlund et al., 2011; Schossau et al., 2015; Marstaller et al., 2013) producing efficient digital logic circuitry tractable for information and representation analysis (Kirkpatrick and Hintze, 2020, 2019; Hintze et al., 2018; Albantakis et al., 2014) . Markov Brain technology remains yet underutilized compared to the widespread popularity of DNNs, despite a history of utility for research and having promising industrial application because of its ability to include arbitrary computational substrates (Hintze et al., 2019) (the Evolutionary Buffet Method), and for its natural resulting efficient circuit design. The use of Markov Brains is even well-supported by the Modular Agent Based Evolution Framework (MABE) (Bohm et al., 2017), which allows practitioners to easily implement high-performance agent-based tasks and other plugin modules. Even this convenient framework has only a modest worldwide user community. We assume that a key factor responsible for the lack of traction is the requirement of a genetic algorithm (GA), because any GA-based algorithm comes with numerous limitations that cannot be changed (Sivanandam and Deepa, 2008). Similar arguments were also made about training neural networks, but they are now one of the key technologies in the artificial intelligence domain (Soria-Frisch, 2017). So, we have to ask "what made deep learning so successful?" and "can this be applied to Markov Brain technology?"

Deep learning benefited from various technical improvements specific to neural networks and their training (LeCun et al., 2015; Wason, 2018). While those advancements are technology-specific, there were four other improvements that launched DNNs to popularity that could be applied to Markov Brains:

1. Cross-platform Support
2. Educational Resources
3. Large Free Datasets
4. Parallelization

TensorFlow (Abadi et al., 2016) provides the back-end for other end user-friendly Python libraries such as PyTorch (Paszke et al., 2019) and Keras (Chollet et al., 2015). Next, we detail the advancements of these enabling technologies for DNNs in each of the 4 areas above, and how and if MABE for Markov Brains can or has made these advancements.

Cross-Platform Support TensorFlow et al. are available for all major operating systems: Windows, Linux, and OSX. The MABE framework also supports these systems, and various build environments and integrated development environments each system might provide.

Educational Resources The DNN python tooling efforts provide quality educational resources through excellent documentation and numerous free tutorials. In the same venue, but of course on a much smaller scale, MABE provides documentation and user support. The extension of this effort is ongoing (ALIFE Conference 2018 Discussion on common research software, ALIFE Conference 2020 MABE Tutorial).

Large Free Datasets ImageNet (Deng et al., 2009) is only one of many examples how vast data sets for training deep learning have been made accessible. Similarly, websites like `Kaggle.com` even provide deep learning challenges further disseminating deep learning technology. MABE uses genetic algorithms, and in this domain of GAs we find similar endeavors. OpenAI Gym (Brockman et al., 2016) — while primarily a tool for reinforcement learning, provides tasks for neuroevolution. The annual GECCO Humies awards is another competitive event — Human-Competitive Results Produced by Genetic and Evolutionary Computation.

Parallelization GPU computing has been a necessary boon to DNN technology, enabling large-scale parallelization using Single Instruction, Multiple Data processing (Takizawa et al., 2009; Lahabar et al., 2008). The area of parallelization is ripe for Markov Brain advancement, to mitigate limitations of the GA and to capitalize on the digital logic gate intrinsics of the default substrate. The next section will discuss this in more detail.

For Markov Brains, the first three topics are already addressed by the community. This leaves parallelization as a fourth possible option for improvement. GPU computing plays a critical role in training deep neural networks as they can perform the required matrix operations in parallel. While a linear algorithm performing a matrix multiplication of a vector scales with $O(n^\alpha)$ (with $\alpha > 2$) doing the same in parallel reduces this to a linear complexity $O(n\alpha)$ (with $\alpha > 1.0$). It is this complexity reduction that leads to a much faster computation of deep learning algorithms when using GPU or TPU hardware. A Markov Brain uses a state vector, which is comprised of new sensor inputs, hidden states, and motor outputs. A set of computational operations now defines the next state of this vector (see Figure 1). While in most implementations (known to the authors) the operations of this set are computed sequentially, they are still applied in parallel. This is easiest imagined when considering the case where deterministic logic gates are used as computational operators. When two operators write into the same node their input is summed using a logical OR operation. An alternative is to have each computational operator to write into only one node, which allows also for more complex operators, such as they are found in genetic programming (Koza and Poli, 2005) to work in parallel. For that reason, a Markov Brain architecture where only deterministic logic operators each writing into separate nodes is considered here.

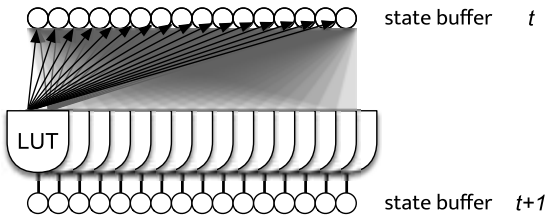


Figure 1: Illustration of a Markov Brain. Logic Gates, here depicted as and LookUp Table (LUT) gates with two inputs, can read from any node (bit) from the state buffer at time point t. Each gate then updates one state of the buffer at time point $t + 1$. For an implementation in an FPGA it is not only crucial to define the logic of each gate, but to also in principle allow each gate to connect to each state.

In this configuration, the computational complexity is constant regardless of the size of the Markov Brain $O = \alpha$. In other words, the time it needs to compute a single update of a Markov Brain — when using a parallel implementation — is theoretically independent of its size. While this might be obvious, there is currently no hardware that allows us to run a Markov Brain in such a way that its computation takes only a single clock cycle. Theoretically, because other computational constraints might limit the efficiency gain due to parallel execution, such as *Amdahl's Law* (Amdahl, 1967; Gustafson, 1988). Those issues can stem from computational overhead required to coordinate parallel execution of code. Regardless, it should thus be possible to take advantage of parallel processing to accelerate Markov Brain evaluation. Field Programmable Gate Arrays (FPGA) seem like the perfect solution. After all, a Markov Brain made from deterministic logic gates is nothing else than a network of logic gates, and so an FPGA should be the ideal solution (see Figure 1, the Markov Brain is already essentially an FPGA). Consequently, here we survey how current FPGA hardware supports building a Markov Brain accelerator, and how contemporary hardware solutions limit this endeavor.

GPU vs. FPGA One obvious alternative to an FPGA might be a GPU, where one could take advantage of its many parallel kernels allowing each one to compute a logic gate

of the Markov Brain. One argument against this approach is the better scalability and power-efficiency of the FPGA (Qasaimeh et al., 2019). Another problem arises from the strict scalar mapping of memory to kernels. Kernels can in principle perform random memory access, meaning for a Markov Brain implementation that the inputs for each gate can be accessed from any kernel. However, the design of a GPU prohibits true random access, and our experience using a GPU in this way shows that throughput is indeed significantly reduced due to cache localization and synchronization . Lastly, using a GPU would not provide the great ratio of transistors to Markov Brain logic gates that an FPGA would provide. For the FPGA a single LUT can be used to implement a single logic gate from the Markov Brain. In the case of the GPU, thousands of logic gates are needed for each of the kernels to compute a single logic function, which is inefficient. However, this allows each kernel to compute any of the other possible more complex functions Markov Brains can take advantage of (Hintze et al., 2019). Regardless, here we will focus on a possible solution using FPGAs.

In the following, the process of implementing an FPGA hardware accelerator parallelizing the execution for Markov Brains will be described and evaluated. However, we do not consider just any solution, but only those that follow the above introduced requirements for open science and accessibility to the three major compute platforms. Furthermore, it is not enough to attain speed of execution, but we must also be able to quickly interface with a GA to allow timely evaluation of candidate solutions. In the following Materials and Methods section, the technological choices will not only be reported, but the criteria that lead to them discussed.

Materials and Methods

General Structure

The typical domain of Markov Brains is agent-based control tasks in virtual environments, even though other applications can easily be imagined. MABE and other software that seeks to optimize Markov Brains using a genetic algorithm, thus needs to implement a perception-action cycle. This schema assumes a separation between world and brain. Input enters the brain via sensors, computation based on this sensor data and hidden states is performed, updating hidden states and creating output signals. Those output signals lead to actions in the environment, which in turn generate new sensor inputs, and so forth.

In a computational evolutionary model the performance of each individual of a population of agent controllers needs to be evaluated. Sometimes even in the presence of each other (Olson et al., 2013). There are numerous strategies to accelerate this process. First of all, if agents do not need to interact, their evaluation can be run in parallel using simple multi-threading. Physics, if needed, can be accelerated using modern physics libraries who take advantage of GPUs or other hardware accelerators. Lastly, in the case where the brain of the agent is, for example, a recurrent neural network again Tensorflow run on a GPU or TPU can be used. Even the selection and mutation of solutions can be accelerated using FPGAs Torquato and Fernandes (2019).

Assuming the above, any hardware accelerator regardless of computational neural substrate needs to communicate three types of data: the network structure (once), inputs, and outputs. While the network structure only needs to be conveyed at the very beginning of evaluation, every iteration the sensor inputs need to be relayed from the host (CPU) to the accelerator device, and the computational results of motor outputs need to be relayed back from the device to the host. There are a variety of proposals in ongoing research to solve the communications aspect of these issues for DNN systems with GPU devices (Espeholt et al., 2020; Lan et al., 2021; Dalton et al., 2020; Petrenko et al., 2021; Makoviychuk et al., 2021). However, these problems apply to all optimization problems of this nature with accelerator devices, so we will focus specifically on the acceleration of Markov Brains and not the communication aspects or how all of the previous technologies might interact (see Figure 2).

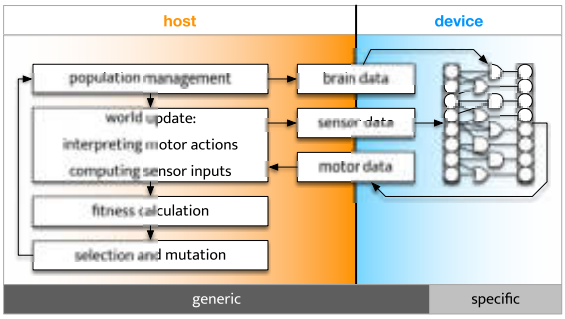


Figure 2: Illustration of the computational evolutionary modeling process. On the left side of the figure all steps that are generic to evolving any computational controller, including the communication part between host and device, and on the right side only the Markov Brain acceleration that is specifically studied here.

.

It is important to mention that all communication aspects can be further accelerated by merging host and device. For example, systems on a Chip (SoC) devices can provide such capabilities — assuming they solve the acceleration of Markov Brains.

Limitation of Existing FPGAs: Operating Systems & Encrypted Configuration

The two largest FPGA manufacturers are AMD (formerly Xilinx) and Intel (formerly Altera), and each provides a plethora of FPGAs together with proprietary design software (Vivado and Quartus) running only on Windows and Linux (Wikipedia, 2022). Neither development software nor

FPGA hardware seems to be produced for OSX on Apple hardware. Both companies fail to meet our requirements because they provide only proprietary development software, and enterprise solutions cost a licensing fee. Furthermore, the proprietary nature of the software makes the bit-stream packages that ultimately program the FPGA effectively encrypted — meaning it is not clear how they actually encode the FPGA configuration, only that the bit stream implements the previously described logic but not exactly how. While this is irrelevant for most industry applications, it matters when it comes to evaluating the actual configuration of the FPGA, as well as in the context of transparent AI (Castelvecchi, 2016).

The above considerations leave Yosys (Wolf et al., 2013) as the only development toolchain to run on all three operating systems. It is also open source, and the translation from Verilog to bitstream is known. This limits the number of FPGAs that can be programmed in this fashion to only a small few, mostly from Lattice Semiconductors. Here, an Icestick 40 HX1K (Evaluation Kit) was used. The Icestudio visual programming environment (icestudio.io) based on the Yosys development toolchain was used to design and program the FPGA.

One cannot avoid wondering "why are both major FPGA manufacturers using proprietary designs, especially if this increases the time it takes to compile the bitstream?" Aside from selling the FPGA hardware, there are recurring licensing fees for enterprise-only features of the development software. Revenue from licensing might be why FPGAs do not find a wider application. At the same time, this model allows companies to sell more than the relatively cheap hardware, especially if this hardware can be reconfigured into any other hardware.

Limitations of Existing FPGAs: Reprogramming Speed

The typical design flow for an FPGA starts with the user defining the desired logic layout. This is done using scripting languages such as Verilog or VHDL (Bailey, 2003), or using even higher level syntax such as C. Alternatively, graphical user interfaces (such as Vivado, Quartus, or Icestudio) allow an easier arrangement of modular components and how they connect to the FPGAs pins, clocks, or other devices such as memory controllers. Then a synthesis step is needed to create a bitstream file which carries the information necessary to configure the FPGA according to the previously defined layout. This bitstream is then transferred to the FPGA effectively reprogramming the device.

This process is a serious limitation when it comes to using FPGAs to accelerate a genetic algorithm. In a typical evolutionary run, a minimum of 100 solutions is tested over many generations. This would mean that the synthesis and reprogramming step, which takes seconds to minutes, is run 10 million times for a single experiment taking 100.000 generations. It is obvious that adding 100 CPU days (assuming around 1 second for synthesis and reprogramming) is preventative costly. The alternative would be to synthesize or optimize the bitstream directly, which for the most modern FPGAs is impossible because this synthesis step is proprietary. This problem is already known in other contexts (Leong et al., 2001; Skliarova and de Brito Ferrari, 2004; Chakraborty et al., 2013). Even if the bitstream is synthesized directly, it also does not remove the reprogramming step.

First generation FPGAs used fuses which were "burnt" during programming, effectively soldering them and allowing only for a single programming step. Then came EPROM and EEPROM-based configuration, where the bitstream is loaded into the EPROM, which in turn defines the FPGA's logic. This is a very tedious and time-consuming process, mostly replaced by SRAM-based configuration (Cofer and Harding, 2006). Here, an external none volatile memory storage like flash memory or an SD-card is used that configures on board SRAM, but technically allows fast reprogramming. While this two-step process is convenient because the bitstream is stored persistently on flash memory and could be used to configure the FPGA, it also prevents this configuration from being used for a GA. Non-volatile flash memory can only be reprogrammed up to a million times: an insufficient quantity for this application. Interfacing with the SRAM directly from the host may present an alternative, but the authors could not find such an FPGA, perhaps because such SRAM is distributed across the FPGA making access inconvenient.

A promising alternative could be partially reprogrammable FPGAs. An early version of this allowed the FPGA IO pins to connect back to its own JTAG programmer (Paulsson et al., 2007). While a creative solution, it seems that it did not find widespread application. A more modern version allowing the FPGA to be reprogrammed from within the FPGA is via an Internal Configuration Access Port (ICAP) (Maxfield, 2015), such as in a Xilinx virtex-4 (Ebrahim et al., 2014). This can be done reasonably quickly (800 Mhz) (Duhem et al., 2011), and also on a SoC (System on a Chip) (Al Kadi et al., 2013), allowing further speed improvements by also moving the GA and possible physics calculations all onto the same chip. In both cases the bitstreams were designed beforehand, and this disallows dynamically generated bitstreams during the application as would be required in a GA. Fortunately, a similar discussion is found in the context of fault tolerance of reconfigurable hardware (Garcia et al., 2006). Here, the idea is to reconfigure (reroute) a section of a chip that experienced a fault — a promising line of research, but such technology is not available yet.

With all the above, we find that a fast and quickly reprogrammable FPGA satisfying our design requirements does not exist, but that there have been small steps to solve some

of the intermediary problems. The best candidate at time of writing is a Zynq 7000 SoC chip, and one could configure it to support Markov Brains. However, this would not allow changing the configuration easily or in a transparent and open fashion. Furthermore, this would create a standalone system requiring a unique set of assembly instructions, instead of interacting with the three major operating systems (Win, *nix, OSX) and their well-developed software tooling.

Ironically, we must conclude that even though FPGAs are theoretically a great solution, practically, a different FPGA design optimized for open and fast reprogrammability would be needed. Consequently, here we use an FPGA to implement such a fast reprogrammable FPGA. In one sense this is incredibly wasteful, because we use universal logic gates to implement universal logic gates on a higher level[1]. However, this trade-off allows us to implement a fast reprogrammable FPGA using existing hardware options.

Reprogrammable Markov Brain implementation

The architecture of this implementation focuses on fast evaluation of Markov Brains together with the necessary code to load the data to configure the Markov Brain, to get input states from the host, and to return outputs to the host. This host-device communication can be implemented using a wide array of channels, such as a network adapter, SPI, PCI, or UART/USB. In a SoC one could even map the memory of the embedded host directly to the memory configuring the Markov Brain. Here, due to the hardware limitations of the IceStick-40 HX1K, the UART is the only choice. Observe, that the communication channel between host and device is not the focus of this investigation and could be separately optimized.

Since communication is in most cases serial, the Markov Brain can be configured while the data for that configuration arrives sequentially. Once the Markov Brain has been configured, only data for inputs needs to be received, and outputs returned. See Figure 3 for the general layout of this implementation, consisting of a Communication Controller (that also programs the Markov Brain), UART for IO, and the reprogrammable universal logic gates for the Markov Brain.

The part that implements the Markov Brain is a set of universal logic gates — equivalent to LookUp Tables in circuit design nomenclature (LUTs). Each of them is created as a block storing its own logic and which of the specific bits from the state buffer each gate needs to read. The outputs of all LUTs are pooled and returned to the communication controller. This implementation allows the configured Markov Brain to perform the computation updating the state buffer from $t \rightarrow t+1$ within a single clock tick. If hidden states should be made persistent, then only the new input states are updated by the Communication Controller. Similarly, only

[1]It also reminds us of Christopher Nolan's 2010 movie Inception

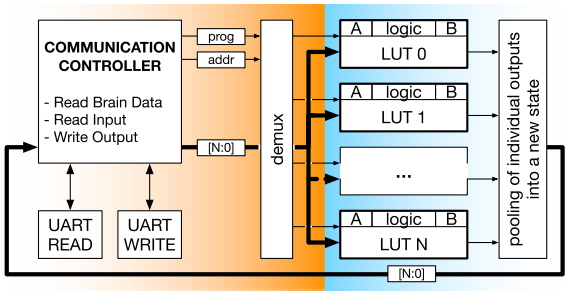


Figure 3: Schematic of the FPGA solution for a fast reprogrammable Markov Brain. In orange all necessary communication and configuration logic, in blue the actual Markov Brain.

.

output states can be communicated back to the host, again reducing the bandwidth for the communication. Here the entire buffer is communicated back and forth for verification purposes.

The current implementation allows Markov Brains with 16 LUTs and consequently a state buffer of 16 binary nodes to be implemented. This implementation also assumes that each node of the state buffer is updated by a single logic gate. If for writing, gates should arbitrarily be connected to states at $t+1$, then their addresses need to be stored as well. Outputs of gates will then not be single bits but a bus as wide as the state buffer. Performing an OR operation on all gate outputs would then be applied. While, without optimizing this process, requiring a serious amount of additional logic gates, it would not slow the computation because only an OR operation is needed. This kind of mapping has not been evaluated here.

Active Categorical Perception Task

An Active Categorical Perception Task (ACP) is one of the most-used tasks when evolving of artificial agents controlled by Markov Brains (Marstaller et al., 2013; Schossau et al., 2015; Hintze et al., 2018). In this task an agent with 4 sensors arranged as pairs of 2 separated by a wide gap (equivalent to 2 sensors) must catch or avoid blocks that are always moving roughly towards it. This "game" is played on a toroidal grid, 16 steps wide and 20 high. Blocks have a size of 2 or 4 and move to the left or right 1 step every time they advance lower. When blocks reach the bottom, the 6 blocks wide agent needs to intercept the 2 wide blocks to catch them, while blocks of size 4 need to be avoided. Agents are evaluated on all 64 combinations of blocks, directions, and possible start locations of the block while starting the agent always at column 0. The toroidal shape of the grid allows the agent leaving to either side to return from the other.

The fitness (see Equation 1 for the agent depends on the

number of properly caught and avoided blocks ('C' for correct outcomes, 'I' for incorrect ones):

$$W = 1.10^{(C-I)}. \quad (1)$$

A standard Genetic Algorithm was used, starting with a randomly generated population of 100 agents. Markov Brains were encoded as a set of 16 binary logic gates, with each logic gate having two input wires and four bits defining the output of all possible computations. When agents were selected by roulette wheel selection to propagate into the next generation each logic gate had a 1% chance to change its wiring and logic. Furthermore, each gate had a 1% chance to randomly copy the wiring and logic of another randomly selected gate, effectively implementing a gene duplication. This evolutionary optimization was carried out in MABE (Bohm et al., 2017).

Results

The accuracy of the implementation was evaluated by using a previously evolved agent with perfect performance on the active categorical perception task. The FPGA performed flawlessly. A python wrapper communicating with the FPGA also allows simple continual reprogramming during an entire evolutionary run. However, the FPGA used here runs at 12Mhz and is so slow that such a run would have taken too long. However, the proof of principle of evaluating a few generations on the FPGA by evaluating each Markov Brain on the ACP task worked flawlessly.

Next, we evaluate the speed of the FPGA. Both UART Rx and Tx (input and output communication between host and device) was run at 2 million baud, that is 6 clock cycles (`clk` in circuit design) per transferred bit. One whole round of communication starts with sending the character '`P`' (for programming) and sets the FPGA into a mode that the next 32 bytes are read and used to configure the 16 LUTs comprising the Markov Brain (see Figure 4 *command* and *bytes sent*). That is followed by sending the character '`T`' (for transfer) and sets the FPGA into a mode to receive 2 more bytes (16 bit) defining the state the Markov Brain should use to compute the next state. Once that is done, the letter '`R`' (for read) is sent setting the FPGA into a mode that sends back the 2 bytes defining the computed $t+1$ state. Sending a single byte, means sending 10 bits, one leading and one trailing 0. Each bit, at that baud rate requires 6 `clk` ticks. Consequently, we expect 32 times 60 `clk` ($1,920$ in total) ticks for that transmission (see Figure 4 *interval* and *measurements*). When measuring, we find this to take $1,921$ `clk` in over 90% of the $1,000$ measurements performed. The extra `clk` is used to clean up the state of the finite state machine of the communication controller. We measure more ticks (up to $4,500$) in the remaining 10% that we attribute to synchronization issues of the UART communication between the host and the device. This is followed by sending a character '`T`' that sets the FPGA into the state to receive two more bytes (16 bits) defining the current state of the Markov Brain. This should take 3 times 20 `clk` ticks (180). As above, this happens in 90% of $1,000$ measurements. Because the Markov Brain updates the $t+1$ buffer at every tick, once the t state buffer is loaded, the update is complete. This update is specified in Verilog as *wires* to not require additional `clk`. To receive the newly updated state, the character '`R`' is sent taking another 60 `clk`. As above, this can be confirmed in 90% of the $1,000$ measurements. Then, the FPGA sends back the updated state. Taking all `clk` together we expect this to take 32+3+1 times 60 ($2,160$) `clk`s. The extra `clk` for cleaning up the finite state machine in this case is executed in parallel to the next state and does not contribute. If the Markov Brain update would take additional `clk`s, we would measure more before the next state is returned. Alternatively, the signal might be returned, but the Markov Brain did not have a chance to complete the update. Since this does not happen, and we indeed find this process to actually take $2,160$ `clk` ticks in 90% of $1,000$ measurements, then we analytically confirm that this implementation allows the entire Markov Brain to be updated in a single `clk`. Any delays, measured in 10% of the cases must be attributed to the communication channel.

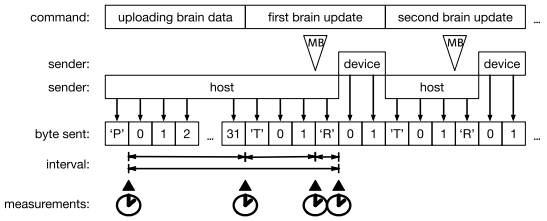


Figure 4: Illustration of the implementation timing. The host sends a '`P`' as the signal to encode the Markov Brain on the FPGA using the next 32 bytes. That is followed by a '`T`' and two more bytes to specify the first state buffer. At the end, the Markov Brain performs the proper update (inverted triangle with 'MB'). Technically, these updates happen every `clk`, but only now are they using the intended data. Typically the host then requests the $t+1$ state by sending an '`R`', which is answered with 2 bytes returning the $t+1$ buffer. After that more data is sent from the host. The clock symbols depict where `clk` counters were reset to time the FPGA. The arrows show the measured intervals.

.

Technically, it is possible that chaining very many logic gates — as it could happen within the LUTs encoding the Markov Brain — leads to signals that need an initial stabilization period. This, and similar phenomena, are summarized under the term "instability" in the FPGA engineering field. This could theoretically lead to faulty signals, and also cause a delay exceeding a single `clk`. FPGAs generally run

slower than modern CPUs thereby mitigating this problem. Additionally, our implementation does not relay the newly computed state back to the host, but waits at least 60 `clk` cycles for a signal from the host to do so. This is more then enough time to account for any instability that may occur. This 60 `clk` delay is of course a constant runtime cost and could be better optimized using different communication channels obviating the need for a delay.

Removing any of the LUTs up to the point where only 1 gate remains did not change the already maximal speed, as we are at the limit fitting on the FPGA used here (900 of the $1,280$ available logic cells). Obviously, removing gates is removing functionality.

Discussion

We showed that an FPGA can be configured such that it uses a single clock cycle (`clk`) to compute the t to $t+1$ update for a Markov Brain using deterministic logic gates. Furthermore, the implementation allows the proper computation of Markov Brains even when integrated into an evolutionary model. However, due to the extremely slow clock of the FPGA (12 Mhz) and the slow UART communication, this implementation does not provide a speed increase compared to the 3.5 Ghz Intel Core i5 host system the device was connected to and compared with.

The first question now is if there would be a benefit by economy of scale, that much larger Markov Brains implemented in this way would still update within a constant single `clk`? The largest currently (2022) available FPGAs have 10 million logic cells (LC), which can all update their outputs within a single `clk`. Speeds of these chips are often lower than that of contemporary CPUs, but speeds of 5Ghz haven been achieved in prototypes already 20 years ago (Clarke, 2001). However, these FPGAs require the conventional toolchain and are not designed to be quickly reprogrammable as defined within this context. Assuming the FPGAs were designed to be quickly reprogrammable — ideally following the design principles laid out here — then they would be able to perform the Markov Brain computation in a single `clk`.

Such reasoning about massively parallel computation in a single `clk` assumes that there is some scaling between the size of the Markov Brain and the required number of logic gates (or transistors needed). So, how does the use of logic cells scale with the size of Markov Brains?

The requirement that needs to be fulfilled here is to allow every logic gate to be able to read from arbitrary locations from the state buffer. Within modern FPGAs the wiring of logic cells to each other is not only proprietary, but also presumably optimized for processing speed, and likely not for reprogrammability. As discussed before, FPGAs are often only reprogrammed during testing and development. The method of implementing this feature, that each logic cell of the Markov Brain can connect to every possible bit in the

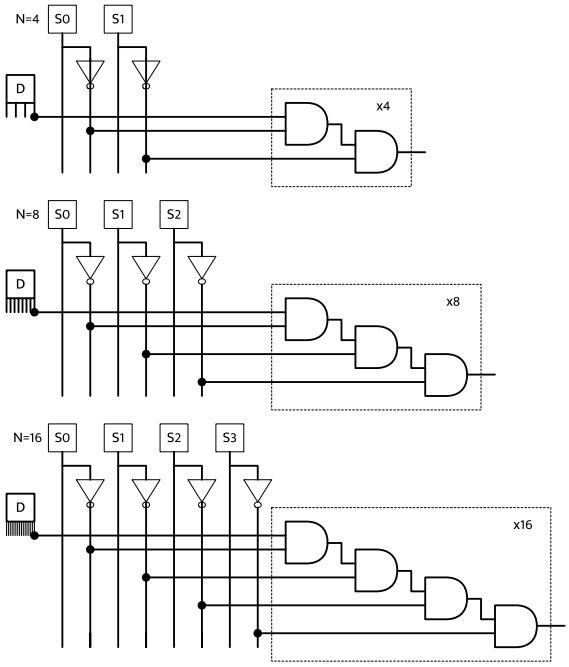


Figure 5: Examples of Demux circuits as they are used here to get the right bit from the current state buffer (D). Depending on the width N of the buffer, $\log_2 N$ signal wires labeled here S_0 to S_3 are needed. For each of the N wires, a set of AND gates is needed shown in the dashed box. The schema is shown for three different sizes of Markov Brains.

.

state buffer, was achieved using a structure similar to a de-multiplexer (see Figure 5). Depending on the size of the state buffer N a set of bits ($S_0 \rightarrow S_{\log_2 N}$)is needed for addressing. These bits specify which bit from the state buffer (D) needs to be selected. The equation to calculate the number of logic gates (L) scales exponentially with the number of bits necessary for addressing:

$$L = S2^S + S. \tag{2}$$

For Markov Brains with $N = 256$ gates, this would result in $500,000$ logic gates needed. While that sounds reasonable, and exceeds what has been used so far, the largest Markov Brain that could be currently supported would be $N = 65,536$, requiring about 70 million logic gates, as much as the largest available CPUs have these days. Observe that under perfect conditions a single CPU performing the same computation would need to retrieve the states from memory, evaluate the logic, and then update the $t+1$ state buffer, each requiring already many `clk`s. Which in serial computation would then be multiplied by N. However, this assumes the above described, and very wasteful use of such a de-multiplexer like arrangement. Fortunately,

there are alternatives for this design, such as transmission gates or combinations and stacking of de-multiplexers. Shift registers could be used for the same purpose, and while they reduce the gate footprint drastically, the require additional `clk`s.

An interesting question is, how FPGAs solve the same problem? After all, are FPGAs not able to connect logic cells arbitrarily? The answer is that FPGAs typically use switch boxes for interconnections, putting a limit on the possible configurations. Configuring these switch boxes is part of the synthesis step of the supporting software. It optimizes the layout of the FPGA. Often gates are not connected in a massive parallel fashion as done here, but in a linear sequence, which hides what is a shortcoming with respect to our requirements.

However, a combination of these methods could be imagined. What remains is that even extremely large Markov Brains can still be accelerated using FPGA technology up to a point. While a different chip design might allow us to to avoid speed/size trade-offs that stem from stereotypical FPGA designs. It seems that the current design, that makes FPGAs great at what they do right now, prevents them from being the perfect Markov Brain accelerator.

A common alternative idea is to use a GPU instead of an FPGA. This approach assumes that one uses each core of the GPU to perform the computation of each computational unit of the Markov Brain. From experience, GPUs work well, when data and processing is aligned. Whereas here, each core must be able to arbitrarily read from a shared state buffer, which drastically slows GPU performance. Secondly, if the Markov Brain is made from deterministic logic gates, a single universal logic gate would be needed to compute this function. Doing the same using a GPU kernel that uses hundreds or thousands of logic gates seems extremely wasteful. However, a GPU kernel could allow the implementation of much more complex computational units. Since GPUs are not optimized for arbitrary random memory access for each kernel in parallel, one could consider a hybrid: Using an FPGA to implement a GPU but with a different data access and pooling architecture, optimized for Markov Brain acceleration — an option we will explore in the future.

Conclusion

Exploring the implementation of a Markov Brain as an FPGA resulted in many interesting insights. First of all one can perform the computation of a Markov Brain made from deterministic logic gates in a single `clk` using an FPGA. However, the fast reprogrammability of such an FPGA was identified as the most important feature when it comes to an actual use within a genetic algorithm. This presents two major shortcomings of current technology: FPGAs are not optimized for the fast and repeated reprogramming that is needed here, and their usually proprietary for-profit development toolchains do not support the spirit of open source or open access. This severely limits the choice of FPGAs and toolchains, and limits a wider application of Markov Brain technology for the major operating systems.

This leaves us with three options: The use of an SoC FPGA, which would allow us to accelerate all computations including the GA and the physics calculation at the same time. However, instead of an accelerator, this would be a "standalone" solution, dependent on current toolchains, limited in the ways described above. It would also not solve the wiring problem discussed above and instead use valuable logic cells to create the inter-connectivity between all gates and the state buffer. Deploying the current solution on a larger FPGA and taking advantage of better host-device communication (for example, a Lattice ECP5 that has an open toolchain) is another viable option. This would also be the simplest way to explore larger Markov Brains, and could answer the question "how long can one maintain a single `clk` update for the entire Markov Brain before speed trade-offs need to be considered?" Lastly, one could explore new FPGA designs that are directly optimized for fast and repeated reprogrammability. They might even only support the Markov Brain paradigm. After all, updating a state buffer using arbitrary logic also allows a user to implement any finite state machine; only limited by the size of the buffer. Instead of running the Markov Brain algorithm on an FPGA, one could run an FPGA on a Markov Brain accelerator — a compelling thought worthy of further exploration.

Acknowledgements

The authors would like to thank David H. Ackley for pointing out that sometimes problems will only be solved by solving them yourself.

References

Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M., et al. (2016). Tensorflow: A system for large-scale machine learning. In *12th {USENIX} symposium on operating systems design and implementation ({OSDI} 16)*, pages 265–283.

Al Kadi, M., Rudolph, P., Gohringer, D., and Hubner, M. (2013). Dynamic and partial reconfiguration of zynq 7000 under linux. In *2013 International Conference on Reconfigurable Computing and FPGAs (ReConFig)*, pages 1–5. IEEE.

Albantakis, L., Hintze, A., Koch, C., Adami, C., and Tononi, G. (2014). Evolution of integrated causal structures in animats exposed to environments of increasing complexity. *PLoS Comput Biol*, 10(12):e1003966.

Amdahl, G. M. (1967). Validity of the single processor approach to achieving large scale computing capabilities. In *Proceedings of the April 18-20, 1967, spring joint computer conference*, pages 483–485.

Bailey, S. (2003). Comparison of vhdl, verilog and systemverilog. *Available for download from www. model. com*, page 29.

Bohm, C., CG, N., and Hintze, A. (2017). MABE (modular agent based evolver): A framework for digital evolution research. *Proceedings of the European Conference of Artificial Life*.

Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., and Zaremba, W. (2016). Openai gym. *arXiv preprint arXiv:1606.01540*.

Castelvecchi, D. (2016). Can we open the black box of ai? *Nature News*, 538(7623):20.

Chakraborty, R. S., Saha, I., Palchaudhuri, A., and Naik, G. K. (2013). Hardware trojan insertion by direct modification of fpga configuration bitstream. *IEEE Design & Test*, 30(2):45–54.

Chollet, F. et al. (2015). Keras. `https://github.com/fchollet/keras`.

Clarke, P. (2001). Sige process pushes reconfigurable fpga to 5 ghz.

Cofer, R. and Harding, B. F. (2006). *Rapid System Prototyping with FPGAs: Accelerating the Design Process*. Elsevier.

Dalton, S. et al. (2020). Accelerating reinforcement learning through gpu atari emulation. *Advances in Neural Information Processing Systems*, 33:19773–19782.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. (2009). Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee.

Duhem, F., Muller, F., and Lorenzini, P. (2011). Farm: Fast reconfiguration manager for reducing reconfiguration time overhead on fpga. In *International Symposium on Applied Reconfigurable Computing*, pages 253–260. Springer.

Ebrahim, A., Arslan, T., and Iturbe, X. (2014). On enhancing the reliability of internal configuration controllers in fpgas. In *2014 NASA/ESA Conference on Adaptive Hardware and Systems (AHS)*, pages 83–88. IEEE.

Edlund, J. A., Chaumont, N., Hintze, A., Koch, C., Tononi, G., and Adami, C. (2011). Integrated information increases with fitness in the evolution of animats. *PLoS Comput Biol*, 7(10):e1002236.

Espeholt, L., Marinier, R., Stanczyk, P., Wang, K., and Michalski, M. (2020). Seed rl: Scalable and efficient deep-rl with accelerated central inference. In *International Conference on Learning Representations*.

Garcia, P., Compton, K., Schulte, M., Blem, E., and Fu, W. (2006). An overview of reconfigurable hardware in embedded systems. *EURASIP Journal on Embedded Systems*, 2006:1–19.

Gustafson, J. L. (1988). Reevaluating amdahl's law. *Communications of the ACM*, 31(5):532–533.

Hintze, A., Edlund, J. A., Olson, R. S., Knoester, D. B., Schossau, J., Albantakis, L., Tehrani-Saleh, A., Kvam, P., Sheneman, L., Goldsby, H., et al. (2017). Markov brains: A technical introduction. *arXiv preprint arXiv:1709.05601*.

Hintze, A., Kirkpatrick, D., and Adami, C. (2018). The structure of evolved representations across different substrates for artificial intelligence. In *Artificial Life Conference Proceedings*, pages 388–395. MIT Press.

Hintze, A., Schossau, J., and Bohm, C. (2019). The evolutionary buffet method. In *Genetic Programming Theory and Practice XVI*, pages 17–36. Springer.

Kirkpatrick, D. and Hintze, A. (2019). The role of ambient noise in the evolution of robust mental representations in cognitive systems. In *to appear in the Proceedings of the Artificial Life Conference 2019*. Cambridge, MA: MIT Press.

Kirkpatrick, D. and Hintze, A. (2020). The evolution of representations in genetic programming trees. In *Genetic Programming Theory and Practice XVII*, pages 121–143. Springer.

Koza, J. R. and Poli, R. (2005). Genetic programming. In *Search methodologies*, pages 127–164. Springer.

Kvam, P., Cesario, J., Schossau, J., Eisthen, H., and Hintze, A. (2015). Computational evolution of decision-making strategies. *arXiv preprint arXiv:1509.05646*.

Lahabar, S., Agrawal, P., and Narayanan, P. (2008). High performance pattern recognition on gpu. *Proceedings of NCVPRIPG*, 2008:154–159.

Lan, T., Srinivasa, S., Wang, H., and Zheng, S. (2021). Warpdrive: Extremely fast end-to-end deep multi-agent reinforcement learning on a gpu. *arXiv preprint arXiv:2108.13976*.

LeCun, Y., Bengio, Y., and Hinton, G. (2015). Deep learning. *nature*, 521(7553):436–444.

Leong, P. H. W., Sham, C.-W., Wong, W., Wong, H., Yuen, W. S., and Leong, M.-P. (2001). A bitstream reconfigurable fpga implementation of the wsat algorithm. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 9(1):197–201.

Makoviychuk, V., Wawrzyniak, L., Guo, Y., Lu, M., Storey, K., Macklin, M., Hoeller, D., Rudin, N., Allshire, A., Handa, A., et al. (2021). Isaac gym: High performance gpu-based physics simulation for robot learning. *arXiv preprint arXiv:2108.10470*.

Marstaller, L., Hintze, A., and Adami, C. (2013). The evolution of representation in simple cognitive networks. *Neural computation*, 25(8):2079–2107.

Maxfield, C. (2015). The mcu guy's introduction to fpgas: Configuration techniques & technologies.

Miller, J. F. and Harding, S. L. (2009). Cartesian genetic programming. In *Proceedings of the 11th annual conference companion on genetic and evolutionary computation conference: late breaking papers*, pages 3489–3512.

Olson, R. S., Hintze, A., Dyer, F. C., Knoester, D. B., and Adami, C. (2013). Predator confusion is sufficient to evolve swarming behaviour. *Journal of The Royal Society Interface*, 10(85):20130305.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai,

J., and Chintala, S. (2019). Pytorch: An imperative style, high-performance deep learning library. In Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., and Garnett, R., editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc.

Paulsson, K., Hubner, M., Auer, G., Dreschmann, M., Chen, L., and Becker, J. (2007). Implementation of a virtual internal configuration access port (jcap) for enabling partial self-reconfiguration on xilinx spartan iii fpgas. In *2007 International Conference on Field Programmable Logic and Applications*, pages 351–356. IEEE.

Petrenko, A., Wijmans, E., Shacklett, B., and Koltun, V. (2021). Megaverse: Simulating embodied agents at one million experiences per second. In *International Conference on Machine Learning*, pages 8556–8566. PMLR.

Qasaimeh, M., Denolf, K., Lo, J., Vissers, K., Zambreno, J., and Jones, P. H. (2019). Comparing energy efficiency of cpu, gpu and fpga implementations for vision kernels. In *2019 IEEE International Conference on Embedded Software and Systems (ICESS)*, pages 1–8.

Real, E., Liang, C., So, D., and Le, Q. (2020). Automl-zero: Evolving machine learning algorithms from scratch. In *International Conference on Machine Learning*, pages 8007–8019. PMLR.

Schossau, J., Adami, C., and Hintze, A. (2015). Information-theoretic neuro-correlates boost evolution of cognitive systems. *Entropy*, 18(1):6.

Sivanandam, S. and Deepa, S. (2008). Genetic algorithms. In *Introduction to genetic algorithms*, pages 15–37. Springer.

Skliarova, I. and de Brito Ferrari, A. (2004). Reconfigurable hardware sat solvers: A survey of systems. *IEEE Transactions on Computers*, 53(11):1449–1461.

Soria-Frisch, A. (2017). 3 practical thoughts on why deep learning performs so well.

Such, F. P., Madhavan, V., Conti, E., Lehman, J., Stanley, K. O., and Clune, J. (2018). Deep neuroevolution: Genetic algorithms are a competitive alternative for training deep neural networks for reinforcement learning.

Takizawa, H., Chida, T., and Kobayashi, H. (2009). Evaluating computational performance of backpropagation learning on graphics hardware. *Electronic Notes in Theoretical Computer Science*, 225:379–389.

Torquato, M. F. and Fernandes, M. A. (2019). High-performance parallel implementation of genetic algorithm on fpga. *Circuits, Systems, and Signal Processing*, 38(9):4014–4039.

Wason, R. (2018). Deep learning: Evolution and expansion. *Cognitive Systems Research*, 52:701–708.

Wikipedia (2022). Field-programmable gate array — Wikipedia, the free encyclopedia. `http://en.wikipedia.org/w/index.php?title=Field-programmable%20gate%20array&oldid=1063351721`. [Online; accessed 06-January-2022].

Wolf, C., Glaser, J., and Kepler, J. (2013). Yosys-a free verilog synthesis suite. In *Proceedings of the 21st Austrian Workshop on Microelectronics (Austrochip)*.

Cost-efficiency of institutional reward and punishment in cooperation dilemmas

Manh Hong Duong[1] and The Anh Han[2]

[1] School of Mathematics, University of Birmingham, B15 2TT, UK. Email: h.duong@bham.ac.uk
[2] School of Computing, Media and Art, Teeside University, TS1 3BX, UK. Email: T.Han@tees.ac.uk

Introduction. A central challenge in biological, computational and social sciences is to understand the evolution of cooperation within populations of self-regarding individuals and mechanisms that promote it (Perc et al., 2017; Yang et al., 2018; Han, 2013). To this extent, various mechanisms have been revealed and studied using methods from evolutionary game theory, statistical physics and agent-based modelling and simulations (Maynard-Smith, 1982; Hofbauer and Sigmund, 1998; Perc et al., 2017). They include both endogenous and exogenous mechanisms such as kin and group selection, direct and indirect reciprocity, spatial networks (Nowak, 2006b), reward and punishment (Sigmund et al., 2001), and pre-commitments (Han et al., 2015). Institutional incentives, positive (reward) and negative (punishment), are among of the most important ones (Sigmund et al., 2001; Van Lange et al., 2014). In institutional incentives, an external decision maker (e.g. institutions such as the United Nations and the European Union) who has a budget to interfere in the population in order to achieve a desirable outcome, for instance to ensure a desired level of cooperation. Providing incentives for promoting cooperation is costly and it is thus important to optimize the cost while ensuring a sufficient level of cooperation (Ostrom, 1990; Chen et al., 2015; Cimpeanu et al., 2021). In the literature, evolution of populations can be studied using either a deterministic approach, which utilizes the continuous replicator dynamics assuming infinite populations, or a stochastic approach, which employs Markov chain for modelling finite populations. For infinite populations, Wang et al. (2019) has recently exploited optimal control theory to provide an analytical solution for cost optimization of institutional incentives. This work therefore does not take into account various stochastic effects of evolutionary dynamics such as mutation and those resulting from behavioural update (Nowak et al., 2004). This might be problematic since undesired behaviours can reoccur over time, via mutation or when incentives were not strong or effective enough in the past. Moreover, a key factor in behavioural update, the intensity of selection (Sigmund, 2010)—which determines how strongly an individual bases her decision to copy another individual's strategy on fitness difference and is absent in the continuous approach—might influence the incentivisation strategy and its cost efficiency as well. For finite populations, so far this problem has been investigated primarily based on agent-based and numerical simulations (Sasaki et al., 2012; Han and Tran-Thanh, 2018; Cimpeanu et al., 2019). In this extended abstract, starting from a finite population framework in (Han and Tran-Thanh, 2018), we summarize a recent publication (Duong and Han, 2021) that provides a *rigorous analysis*, supported by numerical simulations, for this problem and discuss open problems in this emerging research direction.

Models and Methods. We consider a well-mixed, finite population of N self-regarding individuals or players, who interact with each other using one of the following cooperation dilemmas, namely the Donation Game (DG) and the Public Goods Game (PGG). We adopt here the finite population dynamics with the Fermi strategy update rule (Traulsen et al., 2006), stating that a player A with fitness f_A adopts the strategy of another player B with fitness f_B with a probability given by, $P_{A,B} = \left(1 + e^{-\beta(f_B - f_A)}\right)^{-1}$, where β represents the intensity of selection. To reward a cooperator (resp., punish a defector), the institution has to pay (fine) an amount θ (resp., θ) so that the cooperator's (defector's) payoff increases (decreases) by θ. The population dynamics are modelled using an absorbing Markov chain consisting of $(N+1)$ states, $\{S_0, ..., S_N\}$, where S_i represents a population with i C players. S_0 and S_N are absorbing states. Let $U = \{u_{ij}\}_{i,j=1}^{N-1}$ denote the transition matrix between the $N-1$ transient states, $\{S_1, ..., S_{N-1}\}$. The transition probabilities can be defined as follows, for $1 \leq i \leq N-1$:

$$
\begin{aligned}
u_{i,i\pm j} &= 0 \qquad \text{for all } j \geq 2,\\
u_{i,i\pm 1} &= \frac{N-i}{N}\frac{i}{N}\left(1 + e^{\mp\beta[\Pi_C(i) - \Pi_D(i) + \theta]}\right)^{-1},\\
u_{i,i} &= 1 - u_{i,i+1} - u_{i,i-1},
\end{aligned}
$$

where $\Pi_C(i)$ and $\Pi_D(i)$ represent the average payoffs of a C and D player, respectively, in a population with i C players and $(N-i)$ D players. In the DG and the PGG, $\Pi_C(i) - \Pi_D(i)$ is always a negative constant, which is de-

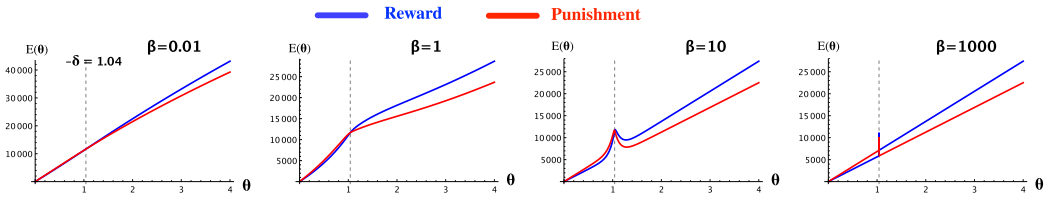


Figure 1: The expected total cost of investment E for reward and punishment, for varying θ and different values of β. Donation game: $b = 2$, $c = 1$; $N = 50$. When $\theta < -\delta$, punishment is more costly than reward, which is reversed when $\theta \geq -\delta$.

noted by $\delta < 0$. The entries n_{ij} of the so-called fundamental matrix $\mathcal{N} = (n_{ij})_{i,j=1}^{N-1} = (I - U)^{-1}$ of the absorbing Markov chain gives the expected number of times the population is in the state S_j if it is started in the transient state S_i (Kemeny and Snell, 1976). As a mutant can randomly occur either at S_0 or S_N, the expected number of visits at state S_i is: $\frac{1}{2}(n_{1i} + n_{N-1,i})$. The frequency of cooperation is given by $\frac{\rho_{D,C}}{\rho_{D,C}+\rho_{C,D}}$, where $\rho_{C,D}$ (resp. $\rho_{D,C}$) is the fixation probabilities of a C (respectively, D) player in a (homogeneous) population of D (respectively, C) players. Hence, this frequency of cooperation can be maximised by maximising

$$\max_{\theta} (\rho_{D,C}/\rho_{C,D}) = \max_{\theta} e^{\beta(N-1)(\delta+\theta)},$$

where the equality is obtained by simplifying the ratio $\rho_{D,C}/\rho_{C,D}$ following an established procedure (Nowak, 2006a). More generally, assuming that we desire to obtain at least an $\omega \in [0, 1]$ fraction of cooperation, i.e. $\frac{\rho_{D,C}}{\rho_{D,C}+\rho_{C,D}} \geq \omega$, then θ needs to satisfy the following lower bound (Han and Tran-Thanh, 2018)

$$\theta \geq \theta_0 = \frac{1}{(N-1)\beta} \log\left(\frac{\omega}{1-\omega}\right) - \delta.$$

Optimization problems. The expected total cost of interference for institutional reward and institutional punishment are respectively

$$E_r(\theta) = \frac{\theta}{2} \sum_{i=1}^{N-1} (n_{1i} + n_{N-1,i}) i,$$

$$E_p(\theta) = \frac{\theta}{2} \sum_{i=1}^{N-1} (n_{1i} + n_{N-1,i})(N - i).$$

In summary, we obtain the following cost-optimization problems of institutional incentives in stochastic finite populations: $\min_{\theta \geq \theta_0} E(\theta)$, where E is either E_r or E_p.

Main results. The main results of Duong and Han (2021) can be summarized as follows.

1. (infinite population limit)

$$\lim_{N \to +\infty} \frac{E(\theta)}{\frac{N^2\theta}{2}(\ln N + \gamma)} = \begin{cases} 1 + e^{-\beta|\theta - c|} & \text{for DG,} \\ 1 + e^{-\beta|\theta - c(1-\frac{r}{n})|} & \text{for PGG,} \end{cases}$$

where $\gamma = 0.5772...$ is the Euler–Mascheroni constant.

2. (weak selection limits) $\lim_{\beta \to 0} E(\theta) = N^2 \theta H_N$, where $H_N = \sum_{i=1}^{N-1} \frac{1}{i}$ is the harmonic number.

3. (strong selection limit of E_r, E_p is similar)

$$\lim_{\beta \to +\infty} E_r(\theta) = \begin{cases} \frac{N^2}{2}\theta\left(\frac{1}{N-1} + H_N\right) & \text{for} \quad \theta < -\delta, \\ N^2 \theta H_N & \text{for} \quad \theta = -\delta, \\ \frac{N^2}{2}\theta\left(1 + H_N\right) & \text{for} \quad \theta > -\delta. \end{cases}$$

4. There exists a threshold value β^* such that $\theta \mapsto E(\theta)$ is non-decreasing for all $\beta \leq \beta^*$ and is non-monotonic when $\beta > \beta^*$. As a consequence, for $\beta \leq \beta^*$

$$\min_{\theta \geq \theta_0} E(\theta) = E(\theta_0).$$

For $\beta > \beta^*$ and N is not too large ($N \leq N_0$ for some N_0), there exist $\theta_1 < \theta_2$ such that, $E(\theta)$ is increasing when $\theta < \theta_1$, decreasing when $\theta_1 < \theta < \theta_2$ and increasing when $\theta > \theta_2$. Thus, for $N \leq N_0$,

$$\min_{\theta \geq \theta_0} E(\theta) = \min\{E(\theta_0), E(\theta_2)\}.$$

5. $E_r(\theta) < E_p(\theta)$ for $\theta < -\delta, E_r(\theta) = E_p(\theta)$ for $\theta = -\delta$ and $E_r(\theta) > E_p(\theta)$ for $\theta > -\delta$.

Figure 1 demonstrates the behaviour of the cost function in different regimes of intensities of selection, when institutional reward is more or less costly than institutional punishment, as well as the phase transitions that occur when β is sufficiently large.

Summary and Outlook. We have summarized a recent theoretical analysis of the problem of optimizing cost of institutional incentives (for both reward and punishment) while guaranteeing a minimum amount of cooperation, in stochastic finite populations. In this context, institutional approaches have been widely adopted to study biological and artificial life systems (Andras et al., 2018; Jones et al., 2013; Smaldino and Lubell, 2014; Perret et al., 2019; Andras, 2020). This analysis provides new, fundamental insights into a cost-efficient design of institution-based solutions for promoting pro-social behaviours in social and artificial systems.

References

Andras, P. (2020). Composition of games as a model for the evolution of social institutions. In *Artificial Life Conference Proceedings*, pages 171–179. MIT Press.

Andras, P., Esterle, L., Guckert, M., Han, T. A., Lewis, P. R., Milanovic, K., Payne, T., Perret, C., Pitt, J., Powers, S. T., et al. (2018). Trusting intelligent machines: Deepening trust within socio-technical systems. *IEEE Technology and Society Magazine*, 37(4):76–83.

Chen, X., Sasaki, T., Brännström, Å., and Dieckmann, U. (2015). First carrot, then stick: how the adaptive hybridization of incentives promotes cooperation. *Journal of The Royal Society Interface*, 12(102):20140935.

Cimpeanu, T., Han, T. A., and Santos, F. C. (2019). Exogenous rewards for promoting cooperation in scale-free networks. In *The 2018 Conference on Artificial Life: A Hybrid of the European Conference on Artificial Life (ECAL) and the International Conference on the Synthesis and Simulation of Living Systems (ALIFE)*, pages 316–323. MIT Press.

Cimpeanu, T., Perret, C., and Han, T. A. (2021). Cost-efficient interventions for promoting fairness in the ultimatum game. *Knowledge-Based Systems*, 233:107545.

Duong, M. H. and Han, T. A. (2021). Cost efficiency of institutional incentives for promoting cooperation in finite populations. *Proceedings of the Royal Society A*, 477(2254):20210568.

Han, T. A. (2013). *Intention Recognition, Commitments and Their Roles in the Evolution of Cooperation: From Artificial Intelligence Techniques to Evolutionary Game Theory Models*, volume 9. Springer SAPERE series.

Han, T. A., Pereira, L. M., and Lenaerts, T. (2015). Avoiding or Restricting Defectors in Public Goods Games? *J. Royal Soc Interface*, 12(103):20141203.

Han, T. A. and Tran-Thanh, L. (2018). Cost-effective external interference for promoting the evolution of cooperation. *Scientific reports*, 8(1):1–9.

Hofbauer, J. and Sigmund, K. (1998). *Evolutionary Games and Population Dynamics*. Cambridge University Press.

Jones, A. J., Artikis, A., and Pitt, J. (2013). The design of intelligent socio-technical systems. *Artificial Intelligence Review*, 39(1):5–20.

Kemeny, J. and Snell, J. (1976). *Finite Markov Chains*. Undergraduate Texts in Mathematics. Springer.

Maynard-Smith, J. (1982). *Evolution and the Theory of Games*. Cambridge University Press, Cambridge.

Nowak, M. A. (2006a). *Evolutionary Dynamics: Exploring the Equations of Life*. Harvard University Press, Cambridge, MA.

Nowak, M. A. (2006b). Five rules for the evolution of cooperation. *Science*, 314(5805):1560.

Nowak, M. A., Sasaki, A., Taylor, C., and Fudenberg, D. (2004). Emergence of cooperation and evolutionary stability in finite populations. *Nature*, 428:646–650.

Ostrom, E. (1990). *Governing the commons: The evolution of institutions for collective action*. Cambridge university press.

Perc, M., Jordan, J. J., Rand, D. G., Wang, Z., Boccaletti, S., and Szolnoki, A. (2017). Statistical physics of human cooperation. *Phys Rep*, 687:1–51.

Perret, C., Hart, E., and Powers, S. T. (2019). Being a leader or being the leader: The evolution of institutionalised hierarchy. In *Artificial Life Conference Proceedings*, pages 171–178. MIT Press.

Sasaki, T., Brännström, Å., Dieckmann, U., and Sigmund, K. (2012). The take-it-or-leave-it option allows small penalties to overcome social dilemmas. *Proceedings of the National Academy of Sciences*, 109(4):1165–1169.

Sigmund, K. (2010). *The Calculus of Selfishness*. Princeton University Press.

Sigmund, K., Hauert, C., and Nowak, M. (2001). Reward and punishment. *P Natl Acad Sci USA*, 98(19):10757–10762.

Smaldino, P. E. and Lubell, M. (2014). Institutions and cooperation in an ecology of games. *Artificial life*, 20(2):207–221.

Traulsen, A., Nowak, M. A., and Pacheco, J. M. (2006). Stochastic dynamics of invasion and fixation. *Phys. Rev. E*, 74:11909.

Van Lange, P. A., Rockenbach, B., and Yamagishi, T. (2014). *Reward and punishment in social dilemmas*. Oxford University Press.

Wang, S., Chen, X., and Szolnoki, A. (2019). Exploring optimal institutional incentives for public cooperation. *Communications in Nonlinear Science and Numerical Simulation*, 79:104914.

Yang, C.-L., Zhang, B., Charness, G., Li, C., and Lien, J. W. (2018). Endogenous rewards promote cooperation. *Proceedings of the National Academy of Sciences*, 115(40):9968–9973.

Pseudo-attractors in Random Boolean Network Models and Single-Cell Data

Marco Villani[1,2], Gianluca D'Addese[1], Stuart A. Kauffman[3] and Roberto Serra[1,2,4]

[1]Department of Physics, Informatics and Mathematics, Modena and Reggio Emilia University, 41125, Modena (IT)
[2]European Centre for Living Technology, 30123, Venice (IT)
[3]Institute for Systems Biology, Seattle, WA 98109, USA
[4]Institute of Advanced Studies, University of Amsterdam, 1012 GC Amsterdam, (NL)
Corresponding author: marco.villani@unimore.it

Abstract

In this extended abstract two novel concepts are defined in the study of Random Boolean Networks, i.e. those of "pseudo-attractors" and "common sea", and it is shown how their analogues can be measured in experimental data on gene expression in single cells.

Introduction

Random Boolean Networks (RBNs for short, see [1] for a recent review) have been widely studied as abstract models of complex systems, thanks to the possibility of tuning their behaviors from ordered to pseudo-chaotic. They are generic models, which can however be used also to describe important biological phenomena, in particular those concerning gene expression (indeed, RBNs were originally introduced [2] as strongly simplified models of gene regulatory networks).

A RBN is a time-discrete, Boolean deterministic dynamical system where the overall state of a given network of N nodes, $X(t+1) \in \{0,1\}^N$, is uniquely determined by the previous state X(t), given the connection topology and the transition functions at each node. Both connections and transitions functions are chosen at random according to some probability distribution. While the success of this model led to the introduction of several variants, here we will consider the "classical" case, where updating is synchronous: dynamical attractors of finite networks of this kind can be either fixed points or limit cycles, but the oscillations of the latter are largely due to the choice of synchronous updating. While this is a clear choice, it limits the biological plausibility of RBNs to describe gene regulatory networks, since it requires simultaneous forgetting of the previous states of all the nodes. Different updating schemes (e.g. asynchronous) have been proposed [1,3,4], but none can claim undisputed plausibility. In particular, cyclic attractors are fragile if the updating scheme is changed, while on the other hand point attractors are conserved. Moreover, cyclic attractors do not seem to be the analogue of the cell cycle, so experimental data on gene expression do not show this type of time dependence.

While different alternatives have been proposed, the usual recipe to interpret the biological significance in multicellular organisms of RBNs' attractors is that of regarding them as the analogue of cell types. We therefore generalize this approach to pseudo-attractors. In real cells, the analogue of the CS is then the set of genes which take the same value in every cell type.

Pseudo-attractors and common sea

As anticipated, the identification of which genes "take the same value" in different cyclic attractors requires some care, since cycles in RBNs depend to a large extent upon the choice of synchronous updating, which does not have a sound biological basis [5]. Synchronous RBNs are Markovian systems, whose state $\boldsymbol{X}(t+1)$ depends upon $\boldsymbol{X}(t)$, forgetting the previous states of all the nodes of the network. The action of a gene on the activation of other genes takes place through the action of its corresponding protein; therefore, the notion of a single time step corresponds to assuming a common decay time of the different regulatory proteins, which is not supported by biological data.

By following [6] we therefore define, for each N-dimensional attractor, a corresponding constant N-dimensional "pseudo-attractor", in which each component assumes the value 1 if its time average in the dynamic attractor is $\geq\theta$ (in the following we suppose θ=0.5) and take the value 0 otherwise. As a consequence, the relationship between dynamical attractors and pseudo-attractors is not injective, and it qualitatively corresponds to a kind of coarse graining in phase space.

The "common sea" (CS) is then defined as the set of nodes which take the same value in all the pseudo-attractors of a given network realization, while the set of all the other nodes is called the "specific part" (SP). Note that the concept of CS differs from existing ones like the "frozen sea" [7] in that it is based on pseudo-attractors (so that also oscillating nodes can belong to it) and it requires that the nodes take the same value in all the pseudo-attractors.

We studied the properties of the CS and the SP by simulating RBNs which belong to different ensembles, generated with different parameter values. The following results have been presented in [6], so we will avoid to continuously refer to it. Most simulations concern dynamically critical networks (i.e. those whose parameters take values which separate the regions of ordered behaviors from the pseudo chaotic ones) which are particularly interesting, for reasons discussed at length in the literature [7,8], which will not be reviewed in this extended abstract.

It turns out that the fraction of nodes belonging to the CS of critical networks increases as the overall size of the network

(N) is increased, and that it comprises the majority of the nodes. This may look surprising but a simplified mean-field calculation shows that it should indeed be expected. An interesting result comes from the comparison of dynamically critical networks with different average number of connections per node (k) and different biases (b) of the Boolean functions. Indeed, dynamical criticality imposes a relationship between k and b, so it is possible to consider ensembles of networks with different pairs of values. Perhaps surprisingly, simulations show that the criticality condition does not suffice to determine the size of the CS. It appears that the larger the bias, the larger the CS.

It is also interesting to observe the internal organization of the common sea and of the specific part. For example, once the CS of a given network realization is identified, we can look at the topology of the network that is composed of its nodes only. If we identify the subparts of the CS with its weakly connected components (WCCs), then there is often (in 70% of the cases) only one subpart per network realization, but in other cases, there are more than one, although one usually finds a dominant subpart that comprises many more nodes than the others. If we perform a similar analysis on the specific part, we often find a more evenly distributed situation, with more fragments of similar size. It should be emphasized that these are not completely independent parts, and that some changes in one WCC (for example, the knock-out of a gene) can affect the values of nodes in other WCCs.

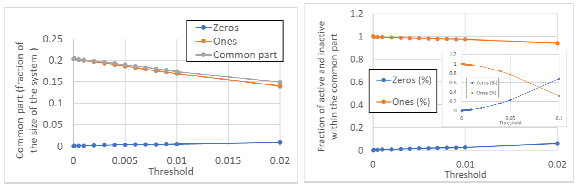
The presence of a large CS obviously limits the maximum possible distance between pairs of attractors, whose distribution turns out to be unimodal.

A Preliminary Look at Single-Cell Data

While we do not aim at an in-depth comparison of the behavior of the models with experimental data, we suggest that looking at experimental data through the lenses of our models can lead to new insights and new questions. To show how this might work, we have performed a preliminary analysis of an important experimental data set concerning the expression levels of human single cells [9]. Although these data are very noisy, since many different exemplars of each type are available, it is possible to aggregate all the contributions into a single profile that then constitutes the "average profile" of the cell type. Since these averages are real valued, binarization is necessary to compare them with pseudo-attractors: a simple way in which binarization can be achieved is by rescaling the values for each gene so to match the [0,1] interval, and by comparing the rescaled values with a fixed threshold ζ.

The approach is simple and it could certainly be refined: however, it is very interesting to observe that the notions of a common sea and specific parts, which have been defined here in a model system, can also be applied to experimental data, as shown e.g. in fig.1.

It remains to investigate possible independent criteria to determine the correct threshold values (possibly different for different genes [10]), an activity that we leave for future work. Here we can note that the approach potentially allows to identify new constraints that simulation models must satisfy in order to correctly interpret experimental data.



(a) (b)

Figure 1 (a) Size of the common part in the Human Cell Landscape data, as the threshold ζ varies. (b) Fraction of active ("Ones") and inactive ("Zeros") genes out of the total of genes belonging to the common sea (in the insert the threshold ζ reaches the value 0.1).

Moreover, the rescaled profiles can be used to compute the distribution of distances (Hamming distances) between cell types – a quantity which can be computed also for our simulated models.

We believe that this kind of comparisons will prove really fruitful to improve both theory and experiment, as it provides new constraints on the acceptable parameters of the model as well as new quantities which are worth measuring. A thorough quantitative comparison between theoretical models and experimental data will be the subject of further work.

Acknowledgments. Useful discussions with Andrea Roli and Michele Braccini are gratefully acknowledged

References

[1] Schwab, J.D.; Kuehlwein, S.D.; Ikonomi, N.; Kuehl, M.; Kestler, H.A. (2020) Concepts in Boolean network modeling: What do they all mean? Comput. Struct. Biotechnol. J., 18:571-582

[2] Kauffman, S.A. (1969) Metabolic Stability and Epigenesis in Randomly Constructed Genetic Nets. Journal of Theoretical Biology 22:437–467

[3] Harvey, I.; Bossomaier, T. (1997) Time out of Joint: Attractors in Asynchronous Random Boolean Networks. In Husbands, P., Harvey, I., editors, *Proceedings of ECAL 97*, pages 67–75. MIT Press, Cambridge, MA

[4] Darabos, C,, Tomassini, M., Giacobini, M. (2009) Dynamics of Noisy and Unperturbed Generalized Boolean Networks. ArXiv 0909.5297v1, DOI:10.1016/j.jtbi.2009.06.027

[5] Gershenson, C. Introduction to Random Boolean Networks. arXiv 2004, arXiv:nlin/0408006

[6] Villani, M.; D'Addese, G.; Kauffman, S.A.; Serra, R. Attractor-Specific and Common Expression Values in Random Boolean Network Models (with a Preliminary Look at Single-Cell Data). Entropy 2022, 24, 311. https://doi.org/10.3390/e24030311

[7] Kauffman, S.A. At Home in the Universe (1995) The Search for Laws of Self-Organization and Complexity; Oxford University Press: Oxford, UK

[8] Roli, A.; Villani, M.; Filisetti, A.; Serra, R. Dynamical Criticality: Overview and Open Questions. J. Syst. Sci. Complex. 2018, 31, 647–663. https://doi.org/10.1007/s11424-017-6117-5

[9] Han, X.; Zhou, Z.; Fei, L.; Sun, H.; Wang, R.; Chen, Y.; Chen, H.; Wang, J.; Tang, H.; Ge, W.; et al. (2020) Construction of a Human Cell Landscape at Single-Cell Level. Nature, 581:303–309

[10] Müssel C.; Schmid F.; Blätte T.J.; Hopfensitz M.; Lausser L.; Kestler H.A.; BiTrinA—multiscale binarization and trinarization with quality analysis, Bioinformatics, 2016, V.32, Issue 3 465–468, https://doi.org/10.1093/bioinformatics/btv591

Shape Change and Control of Pressure-based Soft Agents

Federico Pigozzi

Evolutionary Robotics and Artificial Life lab, Department of Engineering and Architecture, University of Trieste
pigozzife@gmail.com

Abstract

Biological agents possess bodies that are mostly of soft tissues. Researchers have resorted to soft bodies to investigate Artificial Life (ALife)-related questions; similarly, a new era of soft-bodied robots has just begun. Nevertheless, because of their infinite degrees of freedom, soft bodies pose unique challenges in terms of simulation, control, and optimization. Here we propose a novel soft-bodied agents formalism, namely Pressure-based Soft Agents (PSAs): they are bodies of gas enveloped by a chain of springs and masses, with pressure pushing on the masses from inside the body. Pressure endows the agents with structure, while springs and masses simulate softness and allow the agents to assume a large gamut of shapes. Actuation takes place by changing the length of springs or modulating global pressure. We optimize the controller of PSAs for a locomotion task on hilly terrain and an escape task from a cage; the latter is particularly suitable for soft-bodied agents, as it requires the agent to contort itself to squeeze through a small aperture. Our results suggest that PSAs are indeed effective at those tasks and that controlling pressure is fundamental for shape-changing. Looking forward, we envision PSAs to play a role in the modeling of soft-bodied agents, including soft robots and biological cells.[1]

Introduction and related works

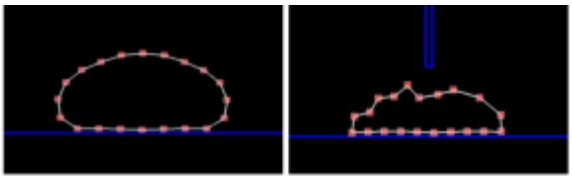
Softness is arguably one of the greatest gifts of mother nature. Every living creature on Earth possesses a body that is mostly made of soft tissues. Soft bodies can continuously bend, stretch, and twist, achieving adaptation to the environment; evolution keeps illuminating new ways to exploit softness, from the amazing manipulation feats of cephalopods (Hochner, 2012), to the protozoans of the genus *Lacrymaria* (Mast, 1911), that can contort their soft flagellum to grasp hard-to-reach preys, allowing for complex hunting dynamics to emerge. It is not surprising that researchers have adopted soft materials to fabricate a new generation of soft robots (Rus and Tolley, 2015), that promises to leverage shape change to recover from damages (Kriegman et al., 2019) and adapt to novel environments (Shah et al., 2021b). In simulation, soft bodies are suitable to investigate virtual creatures for Artificial Life (ALife)-related questions (Joachimczak et al., 2016; Kriegman et al., 2018), including evolutionary robotics (Cheney et al., 2014).

At the same time, the simulation and optimization of soft agents pose unique challenges. No analytical methods exist, as soft bodies have infinite degrees of freedom and entail, in general, hard-to-simulate dynamics (Laschi et al., 2016). Moreover, softness of bodies reinforces the paradigm known as *embodied cognition* (Pfeifer and Bongard, 2006), which posits a deep entanglement between the "brain" of an agent and the "body" that carries it (Pigozzi, 2022). While promising in terms of morphological computation (Nakajima et al., 2015), i.e., the brain offloading part of the computation to the body, such entanglement makes any co-optimization of brain and soft body arduous (Lipson et al., 2016). Finally, how to effectively achieve shape change remains an open issue in the literature (Shah et al., 2021a).

We propose a novel formalism to study soft-bodied agents, namely Pressure-based Soft Agents (PSAs). They are bodies of gas enveloped by a chain of springs and masses, with pressure pushing on the masses from inside the body. Pressure endows the agent with structure, while springs and masses simulate softness and allow the agent morphology to assume a large gamut of shapes, modelling the many degrees of freedom of soft bodies. Actuation takes place by changing the resting length of springs and modulating global pressure. We thoroughly describe the mechanical model and how to simulate it. We also equip the agent with sensing abilities and a closed-loop controller that performs actuation by contracting or expanding the springs and changing global pressure. See Figure 1a for a snapshot of simulation.

Other soft agents formalisms do exist for virtual creatures, in particular, Voxel-based Soft Agents (VSAs) (Hiller and Lipson, 2012; Medvet et al., 2020; Bhatia et al., 2021), which achieve softness by means of a spring-and-masses system, and Tensegrity-based Soft Agents (TSAs) (Rieffel et al., 2009; Zappetti et al., 2017), which achieve softness by connecting cables that are constantly in tension with rods that are constantly under compression. Albeit far-reaching

[1] Videos of evolved agents are available at `https://pressuresoftagents.github.io`.



(a) a PSA (b) escaping from a cage

Figure 1: (Left) Pressure-based Soft Agents (PSAs) are bodies of gas enveloped by a chain of springs and masses, with internal pressure endowing them with structure. Red squares are masses, white strings are springs, and blue shapes are environment bodies. (Right) PSAs can effectively achieve shape change to escape from a cage.

they might be, they still rely on an internal structure of rigid elements for the sake of modelling softness, severely limiting their ability to change shape. Computer graphics, on the other side, employs also pressure-based soft bodies (Matyka and Ollila, 2003), that rely on internal pressure to maintain structure and can thus stretch and bend in any possible configuration. As a result, we ask ourselves whether it is possible to (a) attain PSAs by endowing pressure-based soft bodies with a robotic controller, and (b) effectively exploit shape change for PSAs.

We experiment with a two-dimensional simulation of PSAs and carry out an extensive experimental campaign aimed at validating PSAs on two different tasks: a classic locomotion task on hilly terrain to answer (a), and an escape task from within a cage to answer (b). The latter is particularly suitable to this work as it forces the agent to radically shape-shift in order to escape through an aperture in the cage. We experiment with PSAs of three different sizes and optimize their controller with an established numerical optimizer (Hansen and Ostermeier, 2001).

Our results suggest that PSAs are indeed proficient at solving both traditional tasks—locomotion—and tasks that require changing shape—escape. Moreover, we also show that preventing the controller from modulating pressure (i.e., pressure is the result of only physical interactions) makes it impossible for PSAs to solve the tasks.

Looking forward, we believe PSAs can play a role in the simulation of soft-bodied agents. Indeed, many existing soft robots rely on pressure to shape change, by means of pumps (Kriegman et al., 2021; Shah et al., 2021b) or inflatable tubes (Usevitch et al., 2020; Drotman et al., 2021). Finally, we envision many exciting ALife applications, including the modelling of biological cells that, similarly to PSAs, consist in a fluid, the cytoplasm, enveloped by a flexible membrane.

Proposed agent model

We propose a simple, yet expressive model of soft agents, namely Pressure-based Soft Agents (PSAs). They are bodies of gas contained within an envelope (a chain) of springs and masses, with pressure pushing on the masses from inside the body: actuation takes place by (a) contracting or expanding the springs, and (b) changing pressure. The harmonious execution of these two allows the PSA to assume a large gamut of shapes. By virtue of their many degrees of freedom, PSAs are both 1. expressive, and 2. challenging to control.

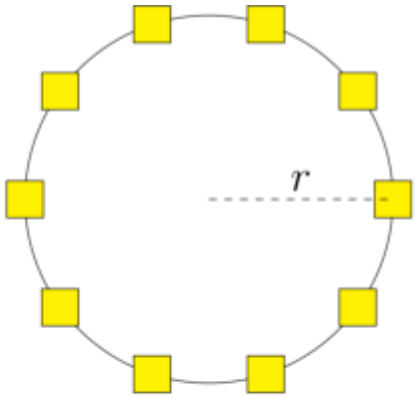


Figure 2: The building blocks of a PSA morphology of radius r: yellow squares are masses, black strings are springs.

We take inspiration from the work of Matyka and Ollila (2003) on pressure-based soft bodies for computer graphics. Such model is particularly suitable for bodies that can bend and twist in arbitrary shapes, as balloons and cloth; as a result, we introduce it to soft agents. To ease modelling, we work with a two-dimensional simulation in discrete time and continuous space. However, we remark that the representations and algorithms of this work are easily portable to the three-dimensional setting.

We define a PSA as the combination of an embodiment, which obeys a *mechanical model* and possesses *sensing* capacities, and a brain, which we implement with a *controller*.

Mechanical model

Morphology A PSA morphology is a *body* of gas contained within an *envelope*. We define an envelope from a circle of radius r; for simplicity, let us assume its center is the origin. We place n_{mass} masses of rigid material equispaced along the circumference, i.e., at points $r + \cos \frac{2\pi i}{n_{\text{mass}}}, r + \sin \frac{2\pi i}{n_{\text{mass}}}$, and fix their rotation. We join each mass with the previous and the next masses along the circumference with distance joints of frequency f, damping ratio d, maximum length l_{max}, minimum length l_{min}. Moreover, there exists an internal pressure p (in Pa) that acts on the masses; with no pressure, the envelope would collapse because of gravity. Pressure thus endows the body with structure. We remark p is global, in the sense that it is the same for all masses. We summarize the building blocks of a PSA morphology in Figure 2.

The masses define the boundaries of the morphology and

collide with external bodies. The joints, by choosing appropriate values for f and d, act as springs: they contract and expand in response to forces acting on the masses they join. As a result, the envelope is not rigid but soft, and the morphology deforms under forces acting on the masses, either exogenous, e.g., contact with other bodies, or endogenous, i.e., changes in p.

We remark that, indeed, spring-and-damper systems are at the heart of other soft agents simulators, including VSAs (Hiller and Lipson, 2012; Medvet et al., 2020) and TSAs (Zappetti et al., 2017). Moreover, springs allow masses to change their relative position, endowing the mechanical model with many degrees of freedom; in fact, the envelope can stretch and bend, and the body can contract or expand in limitless configurations. By virtue of such freedom, our model is suitable for modelling the infinite degrees of freedom of soft bodies, including soft robots. At the same time, such freedom entails that computing the area of the morphology is not tractable: with PSAs, we solve this problem by indirectly updating the area with p, in a way that we detail in the next paragraph.

As an aside, PSAs can be seen not only as robotic agents, but also as a minimal model of a cell: the envelope constitutes the cellular membrane (Singleton et al., 2004), with masses playing the role of membrane proteins and springs the role of lipids. Being fluid, the gas effectively models the cytoplasm (Shepherd, 2006). Finally, p closely resembles turgor pressure acting on the membrane (Pritchard, 2001).

Simulation Area a (in m^2) alters according to pressure p; p, in turn, can be the output of a controller or change according to physical laws. Since pressure is what endows PSAs with structure, we treat the latter as an ablation study in the Results section, and focus on PSAs that control Δp (thus affecting p).

At every time step of simulation, we compute the total pressure acting on the side of a joint, and distribute it over the masses. In detail, we:

(1) query the controller for Δp, and sum it to p.

(2) for every i-th joint, compute the total pressure acting on its side as $p_i = l_i p$, where l_i is the length of the joint, and the normalized normal vector $\hat{\boldsymbol{n}}_i \in [-1, 1]^2$ pointing to the interior of the morphology.

(3) for every j-th mass, let i^- and i^+ be the joints joining it to the previous and next masses in the envelope, respectively. We transform scalar pressure into directed pressure forces $\boldsymbol{p}_{j,i^-} = \frac{p_{i^-}}{2}\hat{\boldsymbol{n}}_{i^-}$ and $\boldsymbol{p}_{j,i^+} = \frac{p_{i^+}}{2}\hat{\boldsymbol{n}}_{i^+}$ acting on the two joints. We divide by 2 to equally distribute pressure on the the masses that anchor a joint.

(4) for every j-th mass, we compute $\boldsymbol{p}_j = \boldsymbol{p}_{j,i^-} + \boldsymbol{p}_{j,i^+}$ and apply it as a force to the mass center. We remark that $\boldsymbol{p}_j$ is indeed in N, as $\hat{\boldsymbol{n}}$ is dimensionless, l_i is in m, and p is in Pa, with $1\,\mathrm{Pa} = 1\,\mathrm{Nm}^{-1}$.

(5) step the physics engine.

Thus, a is not a free parameter (as p), but we affect it through pressure, as higher pressure on the masses implies larger area, and vice versa. Finally, the overall shape of the PSA morphology, i.e., the arrangement and relative positions of the masses, depends on contacts with other bodies, and changes in the resting length of springs dictated by the controller.

Parameters Masses are squares of side $1\,\mathrm{m}$ and density $2500\,\mathrm{kg\,m}^{-2}$; we found results to be consistent also with other sizes and densities. After preliminary experiments and relying on our previous knowledge, we set $f = 8\,\mathrm{Hz}$, $d = 0.3$, $l_{\text{max}} = 1.25l$, and $l_{\text{min}} = 0.75l$, where l is the resting length of a spring. As far as r and n_{mass} are concerned, they vary according to the morphology to simulate, as we shall see in the next section.

Sensing

In order to implement a closed-loop controller, we equip PSAs with sensors. Indeed, sensing is an important property for agents that interact with an environment (Talamini et al., 2019). In this work, we employ touch, pressure, position, and velocity sensors. Touch sensors perceive whether masses are touching other bodies (e.g., the ground) or not, and, for each mass, return 1 if yes, 0 otherwise. The pressure sensor perceives the current internal pressure p, and is thus a proprioceptor. Position sensors perceive the relative x- and y-position of each mass from the center of mass of the morphology. Finally, velocity sensors perceive the x- and y-velocity of the center of mass of the body.

We normalize every sensor reading into $[0, 1]$, and, to introduce sensory memory, compute its average over the last t time steps (using the normalized values). We then concatenate all the sensor readings into an observation vector $\boldsymbol{o} \in [0, 1]^{3n_{\text{mass}}+3}$. After preliminary experiments, we set $t = 25$.

Controller

At every time step of simulation, we feed the current observation vector $\boldsymbol{o}$ to a controller that decides two sets of actions: (a) the resting length of the springs, and (b) the change in pressure Δp.

As far as the former is concerned, given a control value $s \in [-1, 1]$, we instantaneously modify the resting length l of a spring as:

$$l = \begin{cases} l - s(l - l_{\text{min}}) & \text{if } s > 0 \\ l & \text{if } s = 0 \\ l - s(l_{\text{max}} - l) & \text{if } s < 0 \end{cases} \quad (1)$$

Thus, $s = -1$ corresponds to the maximum expansion, and $s = 1$ corresponds to the maximum contraction, all other values lying in between. This is the same model of actuation of other soft robotics simulators, e.g., (Medvet et al., 2020; Bhatia et al., 2021).

Change in pressure, on the other side, requires a different domain, as $[-1, 1]$ is not morphology-agnostic, given that different morphologies require different pressure ranges. As a result, we split the controller into a *pressure controller* π_p and a *springs controller* π_s. The former takes as input $\boldsymbol{o}$ and outputs Δp, that we clip to $[p_{\text{min}}, p_{\text{max}}]$ in order to remain within meaningful boundaries; the latter takes as input $\boldsymbol{o}$ and outputs $\boldsymbol{s} \in [-1, 1]^{n_{\text{mass}}+1}$ (i.e., one control value for every spring).

After preliminary experiments, we implemented π_p as a linear model of the form:

$$\Delta p = \boldsymbol{W}_p \boldsymbol{o} + b_p \tag{2}$$

with weights $\boldsymbol{W}_p \in \mathbb{R}^{1 \times |\boldsymbol{o}|}$ and bias $b_p \in \mathbb{R}$. As a result, $\Delta p \in \mathbb{R}$ and the model can choose the output most appropriate to its morphology. Similarly, we implemented π_s with a non-linearity to ensure the output lies in $[-1, 1]$:

$$\boldsymbol{s} = \tanh(\boldsymbol{W}_s \boldsymbol{o} + \boldsymbol{b}_s) \tag{3}$$

with weight matrix $\boldsymbol{W}_s \in \mathbb{R}^{(n_{\text{mass}}+1) \times |\boldsymbol{o}|}$ and bias vector $\boldsymbol{b}_s \in \mathbb{R}^{n_{\text{mass}}+1}$.

We focus on optimizing the controller of a PSA for a task. Thus, the parameters we optimize are the controller parameters $\boldsymbol{\theta} = [\boldsymbol{\theta}_p \ \boldsymbol{\theta}_s]$, where $\boldsymbol{\theta}_p = [\boldsymbol{W}_p \ b_p]$ are the pressure controller parameters, and $\boldsymbol{\theta}_s = [\boldsymbol{W}_s \ \boldsymbol{b}_s]$ are the springs controller parameters.

Experimental procedure

We performed an experimental campaign aimed at answering the following research questions:

- RQ1 Is the mechanical model valid to simulate pressure-based soft bodies?
- RQ2 Can we control PSAs? In other words, are PSAs capable of solving a classic locomotion task on hilly terrain?
- RQ3 Can we effectively exploit shape change for PSAs?

We design a specific task for each question; we detail the tasks in the next sub-section.

For all tasks, we evaluate three different PSA morphologies, in order to get a sense of the effectiveness of PSAs across a wide array of morphological conditions. For the *large* morphology, we set $n_{\text{mass}} = 20$ and $r = 10\,\text{m}$; for the *medium* morphology, we set $n_{\text{mass}} = 15$ and $r = 7.5\,\text{m}$; finally, for the *small* morphology, we set $n_{\text{mass}} = 10$ and $r = 5\,\text{m}$. For the three morphologies, the input size is 33, 48, and 63, respectively. As a result, the size of the parameter space $|\boldsymbol{\theta}|$ is 408, 833, and 1408, respectively.

Since pressure is what endows PSAs with structure, we investigate whether it is really necessary or not to accomplish the tasks and conduct the following ablation study in RQ2 and RQ3.

Ablation

As an ablation study, we experiment with a configuration *without pressure control*, in contrast to the configuration *with pressure control* considered so far. To this end, we dispense with the pressure controller π_p and let p be the result of physical laws. The pressure of an ideal gas changes according to the ideal gas law of Clapeyron (1834):

$$pa = nRT \tag{4}$$

where p (in Pa) is the pressure value, n is the amount of substance (in mol), R is the ideal gas constant (in $\text{m}^2\text{Pa mol}^{-1}\text{K}^{-1}$), and T is temperature (in K). We remark that many real gases do behave as ideal under various temperature and pressure conditions (Cengel et al., 2011). By fixing T, the right-hand side of Equation (4) is constant: then, p must change to accommodate changes in a and balance the equation. At every time step, we compute a by triangulation and plug it into Equation (4) to compute p.

We set $T = 288.15\,\text{K} = 15\,^\circ\text{C}$ to simulate room temperature, and the gas (the PSA is filled with) to be N_2 (nitrogen), a cheap and common gas. n is the ratio between the gas mass m (in kg) and the molar mass (in kg mol^{-1}), that is $0.028\,031\,4\,\text{kg mol}^{-1}$ for N_2. We set $m = 0.1\,\text{kg}$, $m = 0.075\,\text{kg}$, and $m = 0.05\,\text{kg}$ for the three morphologies respectively. As usual, $R = 8.314\,562\,6\,\text{m}^2\text{Pa mol}^{-1}\text{K}^{-1}$ is the ideal gas constant.

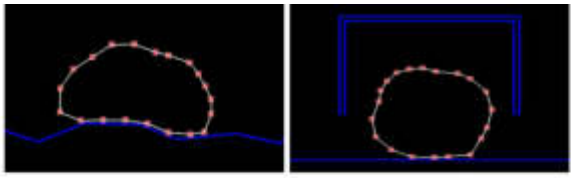
For this configuration, the size of the parameter space $|\boldsymbol{\theta}|$ is 374, 784, and 1334, respectively for the three morphologies; thus, disabling pressure control results not only in simpler actuation, but also in a smaller search space that might benefit optimization.

Finally, for the configuration with pressure control, we set $p_{\text{max}} = \frac{nRT}{\pi r^2}$, where πr^2 is the area of a perfect circle of radius r and $p_{\text{min}} = 0.2 p_{\text{max}}$ to prevent the PSA from collapsing.

Tasks

We evaluate our method on two tasks: locomotion and escape. See Figure 3 for sample frames from these tasks.

Locomotion Locomotion is a classic task in evolutionary robotics (Sims, 1994; Nolfi and Floreano, 2000), and provides a benchmark of basic control skills. It consists in walking as fast as possible over a terrain along the x direction, over an amount of simulated time t_{final}. The fitness function is the velocity $\overline{v}_x$ of the center of mass of the PSA over the



(a) locomotion (b) escape

Figure 3: The tasks considered in our experiments.

simulation. We set $t_{\text{final}} = 30\,\text{s}$. While the terrain is usually a flat surface, we here consider a more challenging hilly terrain, with bumps of different heights and distances. For a given seed, we randomly procedurally generate the bumps with an average height of $1\,\text{m}$, $2\,\text{m}$, and $3\,\text{m}$ for the three morphology sizes, and an average distance of $10\,\text{m}$.

Escape Escape is particularly suitable for soft agents (Cheney et al., 2015), as it forces the agent to radically change its shape to pass through an aperture. At the onset of each simulation, we place the PSA within a cage. The cage amounts to a roof and two walls, with one small aperture per side. The task consists in escaping as fast as possible in any direction over a maximum amount of simulated time t_{final}. The fitness function is the average velocity $\overline{v}$ of the center of mass of the PSA over the simulation, regardless of the direction. We set $t_{\text{final}} = 30\,\text{s}$. The cage is rigid, immobile, and indestructible, forcing the PSA to contort itself and squeeze through one of the apertures. After preliminary experiments, for a PSA of radius r, we set the roof height to $2r + 1$, the walls $3r$ apart from each other, and the apertures one third of the roof height. Escape differs from locomotion in that there is a clear-cut condition for "solving" it, namely when all of the PSA masses are outside of the cage: if this is the case, we terminate the simulation.

Optimization

We optimize the controller parameters $\boldsymbol{\theta}$ with Covariance Matrix Adaptation Evolution Strategy (CMA-ES) (Hansen and Ostermeier, 2001; Hansen, 2016), an established numerical optimizer. While it is possible to use any optimization algorithm, we found CMA-ES to be stable across the different tasks, also thanks to the small size of the search space (Müller and Glasmachers, 2018). CMA-ES iteratively optimizes the solution in the form of a multivariate normal distribution, against a given fitness function. At each iteration, it samples the distribution obtaining a population of solutions and then updates the parameters of the distribution based on the best half of the population. CMA-ES employs non-trivial heuristics while updating the distribution—we refer the reader to Hansen and Ostermeier (2001) for more details.

We use the default parameters suggested in (Hansen, 2006), namely the initial step size $\sigma = 0.5$ and the population size $\lambda = 3\lfloor \log |\boldsymbol{\theta}| \rfloor$. We set the initial vector of means by sampling uniformly the interval $[-1, 1]$ for each vector element. We let CMA-ES iterate until $10\,000$ fitness evaluations have been done.

Settings

For each experiment, we performed 5 evolutionary runs by varying the random seed for CMA-ES and the terrain generation in locomotion. We carried out all statistical tests with the Mann-Whitney U rank test for independent samples. We employ as physics engine the Python wrapper[2] to Box2D (Catto, 2011), a popular 2D physics library written in C++. We set the simulation frequency to $60\,\text{Hz}$ and left all other parameters unchanged. We remark that, for a given seed and controller, all simulations are deterministic. For CMA-ES, we used the implementation of Ha (2017), that is a wrapper around the pycma library (Hansen et al., 2019), and, at a given iteration, parallelize fitness evaluations using multiprocessing. Each run took approximately $1\,\text{h}$ on an Apple M1 MacBook Pro at $3.2\,\text{GHz}$ with $8\,\text{GB}$ RAM and 8 cores. We made the code publicly available at `https://github.com/pigozzif/PressureSoftAgents`.

Results

RQ1: validation of the mechanical model

We validate whether the proposed mechanical model is suitable for simulating pressure-based soft bodies. In particular, for a PSA of radius r, we verify if there exists a p such that a is that of a perfect circle of radius r; in this way, we assess whether our model can correctly simulate a balloon—an ideal pressure-based soft body.

To this end, we conduct the following experiment:

(a) We define a controller that, for a morphology of n_{mass} masses and radius r, outputs $\boldsymbol{s} \in \mathbf{0}^{n_{\text{mass}}+1}$ and $\Delta p = \frac{p_{\text{max}}}{100}$ at every time step. In other words, it does not alter the resting length of springs, while constantly increasing the pressure.

(b) For each of the morphologies, we run a simulation on flat terrain using the aforementioned controller, setting $p = p_{\text{min}} = 0$ at the beginning.

(c) For each time step of simulation, we record pressure p and $\rho = \frac{a}{\pi r^2}$ as performance indexes, πr^2 being the area of a perfect circle of radius r.

If our proposed mechanical model correctly simulates pressure-based soft bodies, there must exist a value p such that $\rho = 1$.

We report the results in Figure 4. According to the figure, the results are qualitatively similar for the three morphologies. Area starts off just above 0, since $p = 0$ and there

[2] `https://github.com/pybox2d/pybox2d`

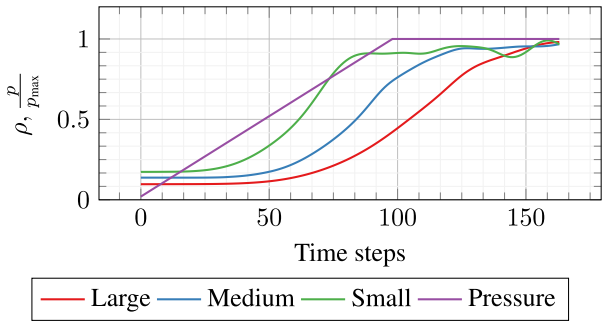


Figure 4: Ratio ρ between the PSA area a and the area of a perfect circle of the same radius, together with relative pressure $\frac{p}{p_{\max}}$, obtained with three sizes. Our proposed mechanical model effectively simulates pressure-based soft bodies, as ρ approaches 1 by constantly increasing pressure.

is no pressure supporting the envelope; it then smoothly increases throughout the simulation, before plateauing at 1 after $p = p_{\max}$.

Through that evidence, we can answer positively to RQ1: our proposed mechanical model is suitable for simulating pressure-based soft bodies.

RQ2: can we control PSAs?

In order to validate the effectiveness of our proposed agent model, we measure the performance of PSAs in a classic locomotion task, in two different settings: with and without pressure control. In both cases, we use $\overline{v}_x$ as performance index.

We summarize the results in Figure 5, which plots $\overline{v}_x$ in terms of median $\pm$ standard deviation for the best individuals over the course of evolution. Moreover, Figure 6 reports boxplots for the distribution of $\overline{v}_x$ of the best individuals. For every morphology, we also show the p-value for the statistical test against the null hypothesis of equality between the medians with and without pressure control.

From the figures, we see that our proposed agent model is effective at the task of locomotion and succeeds in mastering it, regardless of the morphology. We visually inspected the behaviors and found them to be highly adapted for a locomotion task on hilly terrain. PSAs evolve to "roll" over the ground, sliding the masses one after the other, and modulating pressure in order to have the right shape to overcome bumps: in fact, we found that decreasing pressure right before a bump allows the PSA to lower its center of gravity and generate enough momentum to walk over it. On the other side, increasing pressure on flat portions of terrain allows the PSA to bounce over it and generate enough momentum to walk faster. Interestingly, we found some individuals to show life-like behaviors: as a matter of example, when approaching bumps, some stretched out their front (masses and joints) to reach over the tip of the bump, grasp it, and finally

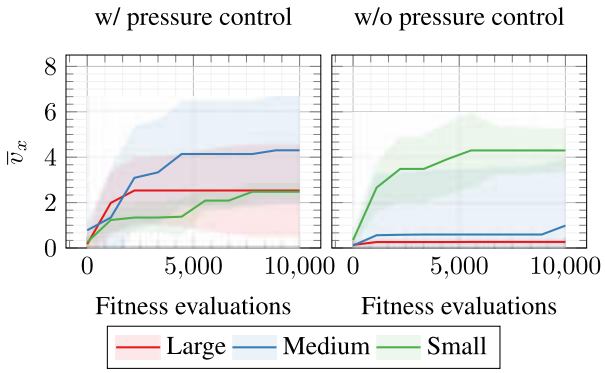


Figure 5: Median $\pm$ standard deviation (solid line and shaded area) of the average velocity of locomotion for the best individuals found during each evolutionary run, obtained with three sizes and with or without pressure control. Our agent model is effective at locomotion on hilly terrain.

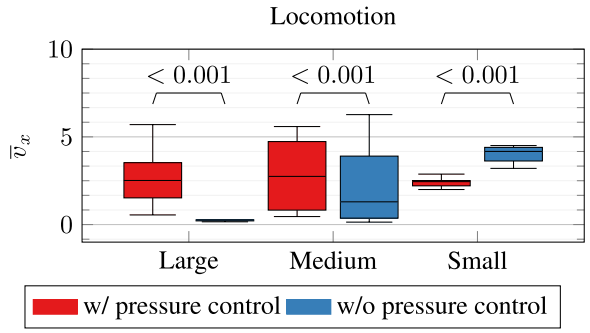


Figure 6: Distribution of the average velocity of locomotion for the best individuals found for each evolutionary run, obtained with three sizes and with or without pressure control. Dispensing with pressure control generally hampers performance in locomotion on hilly terrain. Numbers are p-values.

walk over it. Others appeared to adopt the same strategy to "probe" the terrain in front of them, and plan future actions accordingly. We made videos with pressure control available at `https://pressuresoftagents.github.io`.

According to the figures, PSAs evolved without pressure control were not as effective. In two morphologies out of three, $\overline{v}_x$ does not even depart from its initial value, meaning that no adaptation takes place; surprisingly, the same is not true for the Small morphology, which even succeeds in outperforming its counterpart with pressure control. We remark that, as shown in Figure 6, p-values are significant for all three comparisons. To gain further insights into this phenomenon, we visually inspected the evolved behaviors without pressure control. We found them to be not adapted to a locomotion task on hilly terrain. In particular, Medium and Large PSAs often get stuck in hollows of the terrain; other times, they unsuccessfully struggle to walk over a bump. We believe the reason to be the lack of pressure control: as mentioned before, modulating pressure allows the PSA to deform according to the terrain at hand. This fact also hints at why Small PSAs are effective even without pressure control: thanks to their small size, contacts with the terrain body are relatively enough to allow for sufficient deformation.

Through that evidence, we can answer positively to RQ2: we can conclude that, after optimization, it is possible to control PSAs for a task requiring a decent level of cognition, considering the challenging nature of the hilly terrain. Moreover, ablating the pressure control component of the controller results in much worse performance, especially for bigger (and, we believe, more realistic) morphologies, suggesting that pressure control is an inextricable part of our proposed agent model.

RQ3: can we effectively exploit shape change?

In order to assess the shape-changing abilities of our proposed agent model, we measure the performance of PSAs in an escape task, in two different settings: with and without pressure control. In both cases, we use $\overline{v}$ as performance index.

We summarize the results in Figure 7, which plots $\overline{v}$ in terms of median $\pm$ standard deviation for the best individuals over the course of evolution. Moreover, Figure 8 reports boxplots for the distribution of $\overline{v}$ of the best individuals. For every morphology, we also show the p-value for the statistical test against the null hypothesis of equality between the medians with and without pressure control.

We remark that we say the task "solved" once all of the masses of a PSA are out of the cage; that happens when $\overline{v} \approx 1.0$ for Large, $\overline{v} \approx 0.75$ for Medium, and $\overline{v} \approx 0.5$ for Small. From this consideration and the figures, we see that our proposed agent model effectively solves the task of escape from a cage. We visually inspected the behaviors and found them to be highly adapted for an escape task. Effective individuals decreased their internal pressure to reduce

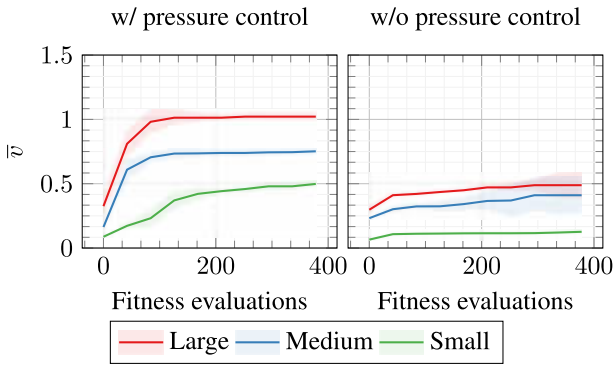


Figure 7: Median $\pm$ standard deviation (solid line and shaded area) of the average velocity of escape for the best individuals found during each evolutionary run, obtained with three sizes and with or without pressure control. Our agent model is effective at shape-changing to escape from a cage.

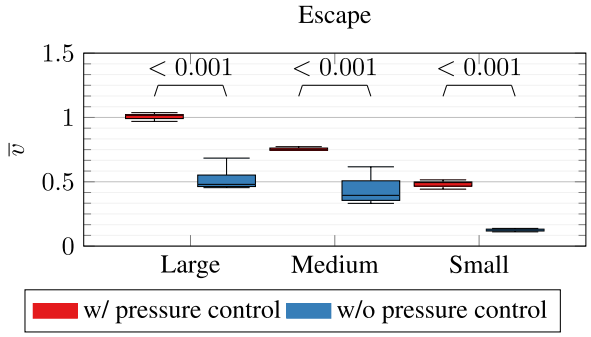


Figure 8: Distribution of the average velocity of escape for the best individuals found for each evolutionary run, obtained with three sizes and with or without pressure control. Dispensing with pressure control makes it impossible to escape from a cage. Numbers are p-values.

their area, almost flattening on the ground (see Figure 1b for a snapshot); then, they slithered through one of the apertures to successfully exit the cage (we remark that agents cannot evolve their initial pressure). Albeit this turned out to be a recurring pattern, we observed some variations. Some individuals, for example, evolved a repulsion for the walls: as soon as any of their touch sensors perceived a wall, they would contract themselves in the opposite direction. The evolution of this trait might be due to the fact that, early in the optimization, we found many individuals to become tangled up as one of the walls wedged between two of their masses (joints, having no mass, cannot oppose to penetration). Other individuals, when flattened, would literally crawl as big cats do when approaching preys, cautiously stretching out one mass after the other.

At the same time, evolution without pressure control did not find effective individuals. From the right plot of Figure 7, we see that $\overline{v}$ barely departs from its initial value. We visually inspected the evolved individuals, and found them to be not adapted at all for an escape task: all of them approached the walls to gain a little $\overline{v}$, but made no attempt at squeezing through the apertures. Intuitively, the reason is their inability to control pressure, as they cannot shape change to effectively solve the task. Figure 8 corroborates these findings by showing that p-values are significant for all three comparisons.

Through that evidence, we can answer positively to RQ3: PSAs can effectively leverage shape change to solve a task that requires squeezing through a small aperture, after optimization. Moreover, ablating the pressure control component of the controller results in no adaptation. To the best of our knowledge, other works on soft robots solve this task by joint optimization of morphology and control (Zardini et al., 2021; Bhatia et al., 2021), which is complex, or morphology alone (Cheney et al., 2015), that might be less feasible than control alone in a real-world setting. In the future, we envision such escape task to be the starting point of more interesting scenarios, like crawling inside caves with challenging terrain, as well as navigating "claustrophobic" mazes as cephalopods can do (Moriyama and Gunji, 1997).

Conclusion

Because of their infinite degrees of freedom, soft bodies pose unique challenges in terms of simulation, control, and optimization. Here we propose a novel soft-bodied agents formalism, namely Pressure-based Soft Agents (PSAs): they are bodies of gas enveloped by a chain of springs and masses that simulates softness, with pressure pushing from inside the body and endowing the agent with structure. Actuation takes place by changing the length of springs or modulating global pressure. By virtue of such a mechanical model, PSAs can assume a large gamut of shapes.

We experimentally investigate whether it is possible to control PSAs and exploit their shape change potential. That is what the paper demonstrates:

(a) we can control PSAs, as optimization finds effective controllers for a locomotion task on hilly terrain, a task that requires a decent degree of cognition to be solved;

(b) we can effectively exploit shape change for PSAs, as optimization finds effective controllers for the task of escape from a cage, a task that requires the agent to contort itself and squeeze through a small aperture.

Among the limitations of this work, it is worth mentioning that, having a low density, PSAs might not be suitable for object manipulation tasks. At the same time, while we believe the model to be promising, as many real soft robots do rely on pressure to achieve shape change (Usevitch et al., 2020; Kriegman et al., 2021), the manufacturability of PSAs is yet to be proven. Future work will address these issues; for the moment, we agree with Kriegman (2019) that virtual creatures can be "as beautiful and complex as life itself". Indeed, we believe PSAs advance reality by providing a unified framework for soft-bodied agents that rely on shape change, so that several aspects can be tested prior to experimental implementation. Other future directions include three-dimensional simulation, distributed controllers, the joint optimization of morphology and control (as already done for other soft agents (Medvet et al., 2021; Zardini et al., 2021)), as well as the simulation of phenomena related to biological cells, such as phagocytosis and mitosis. Opportunities are indeed many, and we have open-sourced our code with a gym (Brockman et al., 2016) interface to encourage usage by other researchers.

Acknowledgements

The authors wish to thank Eric Medvet for providing feedback on an early version of the manuscript.

References

Bhatia, J., Jackson, H., Tian, Y., Xu, J., and Matusik, W. (2021). Evolution gym: A large-scale benchmark for evolving soft robots. *Advances in Neural Information Processing Systems*, 34.

Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., and Zaremba, W. (2016). Openai gym. *arXiv preprint arXiv:1606.01540*.

Catto, E. (2011). Box2d: A 2d physics engine for games. *URL: http://www. box2d. org*.

Cengel, Y. A., Boles, M. A., and Kanoğlu, M. (2011). *Thermodynamics: an engineering approach*, volume 5. McGraw-hill New York.

Cheney, N., Bongard, J., and Lipson, H. (2015). Evolving soft robots in tight spaces. In *Proceedings of the 2015 annual conference on Genetic and Evolutionary Computation*, pages 935–942. ACM.

Cheney, N., MacCurdy, R., Clune, J., and Lipson, H. (2014). Unshackling evolution: evolving soft robots with multiple materials and a powerful generative encoding. *ACM SIGEVOlution*, 7(1):11–23.

Clapeyron, É. (1834). Mémoire sur la puissance motrice de la chaleur. *Journal de l'École polytechnique*, 14:153–190.

Drotman, D., Jadhav, S., Sharp, D., Chan, C., and Tolley, M. T. (2021). Electronics-free pneumatic circuits for controlling soft-legged robots. *Science Robotics*, 6(51):eaay2627.

Ha, D. (2017). Evolving stable strategies. *blog.otoro.net*.

Hansen, N. (2006). The CMA evolution strategy: a comparing review. In *Towards a new evolutionary computation*, pages 75–102. Springer.

Hansen, N. (2016). The cma evolution strategy: A tutorial.

Hansen, N., Akimoto, Y., and Baudis, P. (2019). Cma-es/pycma on github. *Zenodo, doi*, 10.

Hansen, N. and Ostermeier, A. (2001). Completely derandomized self-adaptation in evolution strategies. *Evolutionary Computation*, 9(2):159–195.

Hiller, J. and Lipson, H. (2012). Automatic design and manufacture of soft robots. *IEEE Transactions on Robotics*, 28(2):457–466.

Hochner, B. (2012). An embodied view of octopus neurobiology. *Current biology*, 22(20):R887–R892.

Joachimczak, M., Suzuki, R., and Arita, T. (2016). Artificial metamorphosis: Evolutionary design of transforming, soft-bodied robots. *Artificial Life*, 22(3):271–298. PMID: 27139940.

Kriegman, S. (2019). Why virtual creatures matter. *Nature Machine Intelligence*, 1(10):492–492.

Kriegman, S., Cheney, N., and Bongard, J. (2018). How morphological development can guide evolution. *Scientific reports*, 8(1):13934.

Kriegman, S., Nasab, A. M., Blackiston, D., Steele, H., Levin, M., Kramer-Bottiglio, R., and Bongard, J. (2021). Scale invariant robot behavior with fractals. *arXiv preprint arXiv:2103.04876*.

Kriegman, S., Walker, S., Shah, D., Levin, M., Kramer-Bottiglio, R., and Bongard, J. (2019). Automated shapeshifting for function recovery in damaged robots. *arXiv preprint arXiv:1905.09264*.

Laschi, C., Mazzolai, B., and Cianchetti, M. (2016). Soft robotics: Technologies and systems pushing the boundaries of robot abilities. *Science Robotics*, 1(1):eaah3690.

Lipson, H., Sunspiral, V., Bongard, J., and Cheney, N. (2016). On the difficulty of co-optimizing morphology and control in evolved virtual creatures. In *Artificial Life Conference Proceedings 13*, pages 226–233. MIT Press.

Mast, S. O. (1911). Habits and reactions of the ciliate, lacrymaria. *Journal of Animal Behavior*, 1(4):229.

Matyka, M. and Ollila, M. (2003). Pressure model of soft body simulation. In *The Annual SIGRAD Conference. Special Theme-Real-Time Simulations. Conference Proceedings from SIGRAD2003*, number 010, pages 29–33. Citeseer.

Medvet, E., Bartoli, A., De Lorenzo, A., and Seriani, S. (2020). 2D-VSR-Sim: A simulation tool for the optimization of 2-D voxel-based soft robots. *SoftwareX*, 12.

Medvet, E., Bartoli, A., Pigozzi, F., and Rochelli, M. (2021). Biodiversity in evolved voxel-based soft robots. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 129–137.

Moriyama, T. and Gunji, Y.-P. (1997). Autonomous learning in maze solution by octopus. *Ethology*, 103(6):499–513.

Müller, N. and Glasmachers, T. (2018). Challenges in high-dimensional reinforcement learning with evolution strategies. In *International Conference on Parallel Problem Solving from Nature*, pages 411–423. Springer.

Nakajima, K., Hauser, H., Li, T., and Pfeifer, R. (2015). Information processing via physical soft body. *Scientific reports*, 5(1):1–11.

Nolfi, S. and Floreano, D. (2000). *Evolutionary robotics: The biology, intelligence, and technology of self-organizing machines*. MIT press.

Pfeifer, R. and Bongard, J. (2006). *How the body shapes the way we think: a new view of intelligence*. MIT press.

Pigozzi, F. (2022). Robots: the century past and the century ahead.

Pritchard, J. (2001). Turgor pressure.

Rieffel, J., Valero-Cuevas, F., and Lipson, H. (2009). Automated discovery and optimization of large irregular tensegrity structures. *Computers & Structures*, 87(5-6):368–379.

Rus, D. and Tolley, M. T. (2015). Design, fabrication and control of soft robots. *Nature*, 521(7553):467.

Shah, D., Yang, B., Kriegman, S., Levin, M., Bongard, J., and Kramer-Bottiglio, R. (2021a). Shape changing robots: bioinspiration, simulation, and physical realization. *Advanced Materials*, 33(19):2002882.

Shah, D. S., Powers, J. P., Tilton, L. G., Kriegman, S., Bongard, J., and Kramer-Bottiglio, R. (2021b). A soft robot that adapts to environments through shape change. *Nature Machine Intelligence*, 3(1):51–59.

Shepherd, V. A. (2006). The cytomatrix as a cooperative system of macromolecular and water networks. *Current topics in developmental biology*, 75:171–223.

Sims, K. (1994). Evolving virtual creatures. In *Proceedings of the 21st annual conference on Computer graphics and interactive techniques*, pages 15–22. ACM.

Singleton, P. et al. (2004). *Bacteria in biology, biotechnology and medicine.* Number Ed. 6. John Wiley & Sons.

Talamini, J., Medvet, E., Bartoli, A., and De Lorenzo, A. (2019). Evolutionary synthesis of sensing controllers for voxel-based soft robots. In *Artificial Life Conference Proceedings*, pages 574–581. MIT Press.

Usevitch, N. S., Hammond, Z. M., Schwager, M., Okamura, A. M., Hawkes, E. W., and Follmer, S. (2020). An untethered isoperimetric soft robot. *Science Robotics*, 5(40):eaaz0492.

Zappetti, D., Mintchev, S., Shintake, J., and Floreano, D. (2017). Bio-inspired tensegrity soft modular robots. In *Conference on Biomimetic and Biohybrid Systems*, pages 497–508. Springer.

Zardini, E., Zappetti, D., Zambrano, D., Iacca, G., and Floreano, D. (2021). Seeking quality diversity in evolutionary co-design of morphology and control of soft tensegrity modular robots. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 189–197.

Adversarial Takeover of Neural Cellular Automata

Lorenzo Cavuoti[1], Francesco Sacco[2], Ettore Randazzo[3] and Michael Levin[4]

[1]University of Trieste, [2]University of Pisa, [3]Google Research, [4]Allen Discovery Center
lorenzocav97@gmail.com, francesco215@live.it

Abstract

The biggest open problems in the life sciences concern the algorithms by which competent subunits (cells) could cooperate to form large-scale structures with new, system-level properties. In synthetic bioengineering, multiple cells of diverse origin can be included in chimeric constructs. To facilitate progress in this field, we sought an understanding of multi-scale decision-making by diverse subunits beyond those observed in frozen accidents of biological phylogeny: abstract models of life-as-it-can-be. Neural Cellular Automata (NCA) are a very good inspiration for understanding current and possible living organisms: researchers managed to create NCA that are able to converge to any morphology. In order to simulate a more dynamic situation, we took the NCA model and generalized it to consider multiple NCA rules. We then used this generalized model to change the behavior of a NCA by injecting other types of cells (adversaries) and letting them take over the entire organism to solve a different task. Next we demonstrate that it is possible to stop aging in an existing NCA by injecting adversaries that follow a different rule. Finally, we quantify a distance between NCAs and develop a procedure that allows us to find adversaries close to the original cells.

Introduction

Biology operates in a multiscale competency architecture: cells follow local rules in ways that result in interesting and robust large-scale patterns. Major knowledge gaps, despite progress in molecular genetics, include the policies guiding individual cell behaviors toward body-level anatomical structures. This especially concerns the algorithms needed to reliably reach a consistent form under a range of changing conditions. Cellular behavior is guided in part by gene-regulatory networks inside cells, and coordinated by biochemical and bioelectrical networks at the tissue level. Both of these can be represented as neural networks that guide the mechanisms determining the cell's activity Moore et al. (2018); Biswas S (2016).

Neural Cellular Automata (NCA) represent a recent development of Cellular Automata, where the underlying rule is represented as a neural network and is learned using gradient-based optimization Mordvintsev et al. (2020); the NCA starts from a seed state and is trained to reach a target state (figure 1).

Figure 1: Example of seed and target state, left: the seed state, right: the target state.

In recent advances in this field, scientists have managed to change the global properties of a NCA by adding some cells that follow a different rule Randazzo et al. (2021); this corresponds to biological situations when cells of diverse genetics are assembled into chimeras Nanos V (2021). However these cells remain fixed and can't expand in space.

This is a problem, because the number of new cells (aka. adversaries) required must be relatively high in order to steer the behavior of the whole organism, however, substituting a high number of cells in a biological organism could be difficult. In general, one task in biomedical interventions and in guided self-assembly (bioengineering) contexts is to find the minimal intervention that achieves a given outcome.

One way to minimize the intervention is to generalize NCA to multiple rules, allowing us to simulate what happens when one type of cell overtakes the other. This way we can inject very few adversaries and let them take over other types of cells (figure 2). Furthermore, this model is a generalization of the NCA model[1] that is more biologically plausible, since it can simulate the growth of a group of cells at the expense of another.

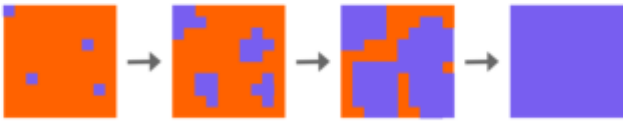


Figure 2: In orange we indicate the original cells while in blue the adversarial cells. The adversaries take over the original cells and change the behavior of the whole organism.

[1]Because, if all the rules are the same, the model behaves like evolving with a single rule.

We then apply the model to two different scenarios:

- **Changing static properties** of a NCA: Examples of such properties are changing the color of an organism (figure 3), adding or removing a limb, making the tail longer, and so on. All of these properties can be observed with a before/after photo of the organism.
- **Changing dynamical properties** of a NCA: Examples of these properties are altering the lifespan of an organism, or the ability to regenerate damage. These properties are much more interesting from an aging and regenerative medicine point of view, however, as we will see, they are much harder to train than the static ones.

Figure 3: Example of changing a static property, the lizard color turns from green to red.

Lastly, the parameters of the adversaries can become drastically different from the original cells they replace, which presents biologists with the challenge of identifying DNA or pharmacological reagents that change some of the cells' behaviors in the necessary fashion. This gives rise to an important inverse problem Lobo D (2014): what can be tweaked at the lowest level (e.g., DNA mutations) to give rise to desired changes at the system level (anatomy)? The difficulty of solving this problem is what prevents true Lamarckian inheritance, and also limits regenerative medicine applications of modern technologies such as CRISPR.

Therefore, we explore ways to make the parameters of the adversarial cells as similar as possible to the original cells, while still being able to accomplish the given task, demonstrating that only a small change in the parameters is sufficient to turn an original cell into an adversarial one.

Figures in video form and the code is available here[2].

The model

NCA model

Before diving into the generalization of NCA, we summarize how the NCA model works; a better explanation can be found in the original paper Mordvintsev et al. (2020).

A Cellular Automata (CA) consists of a grid of cells that is iteratively updated using the same update rule at each step Neumann and Burks (1966), the only requirement is that the next state of each cell depends only on its previous state, x_t and the state of its neighbors, $N(x_t)$.

$$x_{t+1} = f(x_t, N(x_t))$$

Neural Cellular Automata (NCA) use a neural network to model the function f and consider the states x_t to be continuous, this allows training f using gradient-based optimization. The cell state x_t is represented by a vector where the first 4 components represent the RGBA channels of the pixel and the remaining are hidden channels that allow the NCA to pass information between its cells (figure 5 left).

The α channel (transparency) has an important role: if a cell has $\alpha > 0.1$ it means that the cell is mature, otherwise it's dead. This distinction is essential because a cell can change its state if at least one of its immediate neighbors[3], or itself, is mature (figure 4), if this is not the case its state is set to 0. The evolution starts with only one mature cell in the center of the canvas, then the cells are evolved and reach the target image (figure 1).

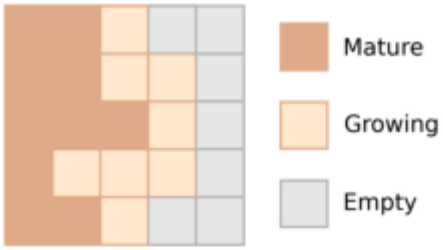


Figure 4: Illustration of mature, growing and empty cells, growing cells are in the immediate neighborhood of mature ones. Image adapted from Mordvintsev et al. (2020), licensed under CC BY 4.0.

Multiple NCA model

We call our new model Multiple NCA because it generalizes a single NCA to multiple update rules. For ease of explanation, we are going to consider the case of only 2 rules, f_1 and f_2.

Masking Since the α channel tells whether a cell is alive or dead, if we have two different types of cells we need two alpha channels, α_1 and α_2. In this new model we decided to put the alpha channels at the end of the state vector (figure 5).

[2] https://letteraunica.github.io/neural_cellular_automata/extra

[3] We consider a Moore neighborhood.

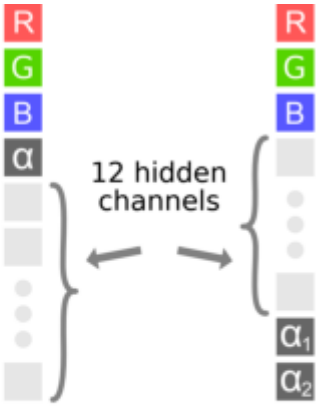


Figure 5: left: location of the channels in the NCA model, right: channels location in the Multiple NCA model.

To make the model more realistic we added some constraints:

1. A cell can be mature in only one channel, this means that no cell can have both alphas > 0.1. We do this because we consider the two cells as having different DNA, so they must have different rules and there is no in-between.

2. We impose that new cells can only grow near mature ones of the same type, example: cells of type 2 can only grow near mature cells of type 2.

3. When both alphas are in $0 \leq \alpha < 0.1$, and the cell is near a mature one of both f_1 and f_2, the cell evolves following the average of both rules (figure 6). Biologically this means that two kinds of cells are fighting for the control of one spot. Mathematically, we did this because otherwise we would have a privileged rule. This also implies that, if the two rules are the same, the system acts identically to what it did if it was evolved with only one rule.

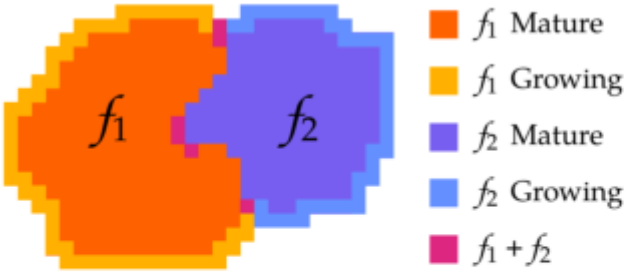


Figure 6: Representation of two different rules evolving toghether in the Multiple NCA model. Magenta cells are growing cells of both f_1 and f_2, so they evolve following the average of both rules.

Furthermore, we change the perception stage and the output of the NCA as follows.

Perception The Multiple NCA model perceives only the sum of all alpha channels[4], and not the individual channels, we do it for two main reasons:

- Computational: we have 2 alpha channels, however, the NCA model uses only one alpha channel, which means that we need to find a function that reduces the number of alpha channels before passing the state to the NCA model to be evolved; summing up the two alpha is one of the simplest ways to do this without having a privileged rule.
- Biological: Summing up the two alpha has a nice biological interpretation, it assumes that a cell is aware of its surroundings but can't distinguish the type of the neighboring cells trivially, since it doesn't have access to α_1 and α_2; this encourages a NCA to take advantage of the hidden channels to be able to distinguish itself from the other NCAs.

From an implementation standpoint we only need a function that takes a state x, sums the alpha, and places the sum right after the RGB components (figure 7); this new state can now be passed to the NCA to be evolved.

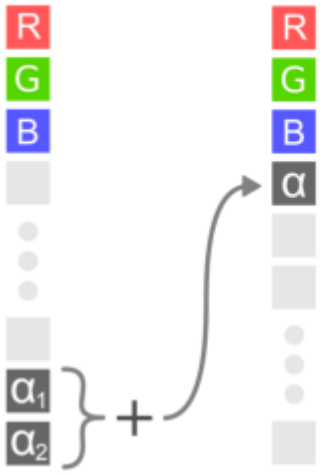


Figure 7: Before passing the state to f_1 or f_2 to be evolved, we sum up α_1 and α_2 then place the sum in the right location in the state vector.

Output We impose that each NCA can only update its alpha channel and not the other ones (figure 8). This makes sense because we don't want a cell of type 1 to edit the alpha channel of a cell of type 2 and therefore kill it trivially.

[4]Other possible functions could have been:

1. Weighted sum of the alphas. However, this implies that there is a privileged rule.

2. Randomly choose an alpha channel to be perceived. However, this doesn't have a nice biological interpretation like summing up the alpha channels.

Instead, if the adversaries want to take over, they must rely on changing their internal state in such a way that makes the original cells undergo the process of apoptosis.

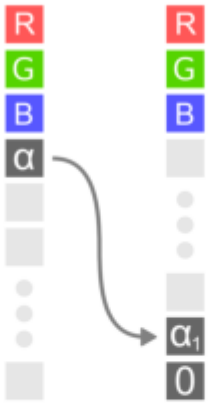


Figure 8: After f_1 computes the update we take α_1 and put it at the end of the state vector, leaving α_2 unchanged.

The whole update step can be seen in figure 9.

Training technique

One of the first issues we encountered, was that the adversarial cells never tried to overtake the original cells, so they never took over the organism. We solved this problem by penalizing the percentage of old cells still present, like so:

$$L = L_{target} + \lambda N_{old}$$

Where L_{target} is the distance to the target image, N_{old} is the number of old cells[5], and λ is a hyperparameter. Considering a loss function like this, nevertheless, leads to another problem: when we first introduce the new cells N_{old} is very high, which in turn makes the loss very high. This means that the adversaries will trade some of the image quality in favor of a faster cell replacement. A solution could be to give the NCA plenty of time before evaluating the loss, however, the NCA might learn to destroy the image at the start, just to rebuild it before the loss evaluation. To address both these problems we made a custom loss function that is dependent on the number of steps n.

$$L = \sum_{n=n_{start}}^{n_{end}} \lambda_1(n) L_{target}(n) + \lambda_2(n) N_{old}(n)$$

Now the hyperparameters λ_1 and λ_2 are functions that depend on the number of steps n, while L_{target} and N_{old} represent respectively the distance from the target image and the number of old cells at the n-th step.

[5]This is slightly wrong since N_{old} is not differentiable, so in practice we used the sum of the α_1 channel over the entire image. However, it is more intuitive to think about penalizing N_{old} rather than α_1.

Experiments

Changing static properties

As we stated in the introduction, static properties include changing the color of an organism, adding or removing a limb, making the tail longer, and so on. We decided to apply our model to 3 cases of increasing difficulty (figure 10):

1. Turning the lizard from green to red: the shape remains fixed but the color of the organism changes.
2. Removing the tail of the lizard: the shape of the organism changes but the color remains the same[6].
3. Turning a bug into a butterfly: both the shape and the color pattern change dramatically.

In every case we start with a pre-trained Persistent NCA (which means it reaches the final image and keeps it for an infinite amount of time) and inject a small percentage of adversaries in it, then we only train the adversaries in order to change the appearance of the entire organism.

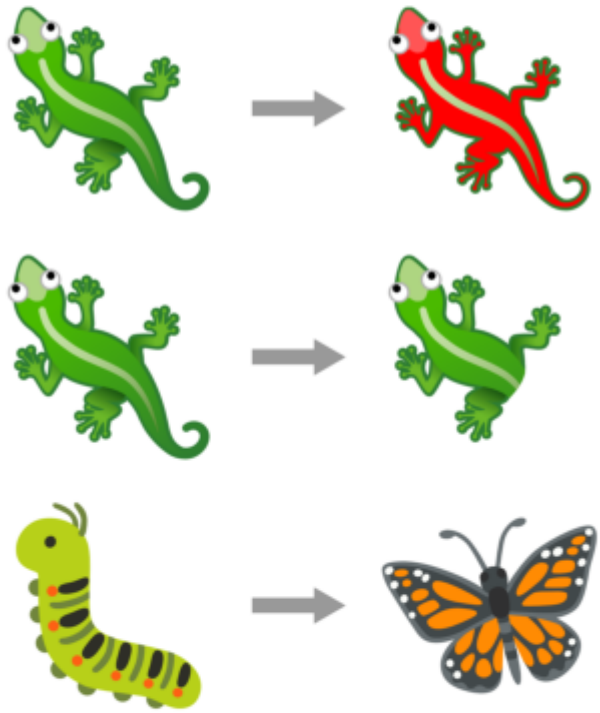


Figure 10: Illustration of the three static experiments.

[6]We found that altering the shape is harder because the adversaries have to learn to overtake the organism first, and, once the original cells are gone, they have to remove the tail.

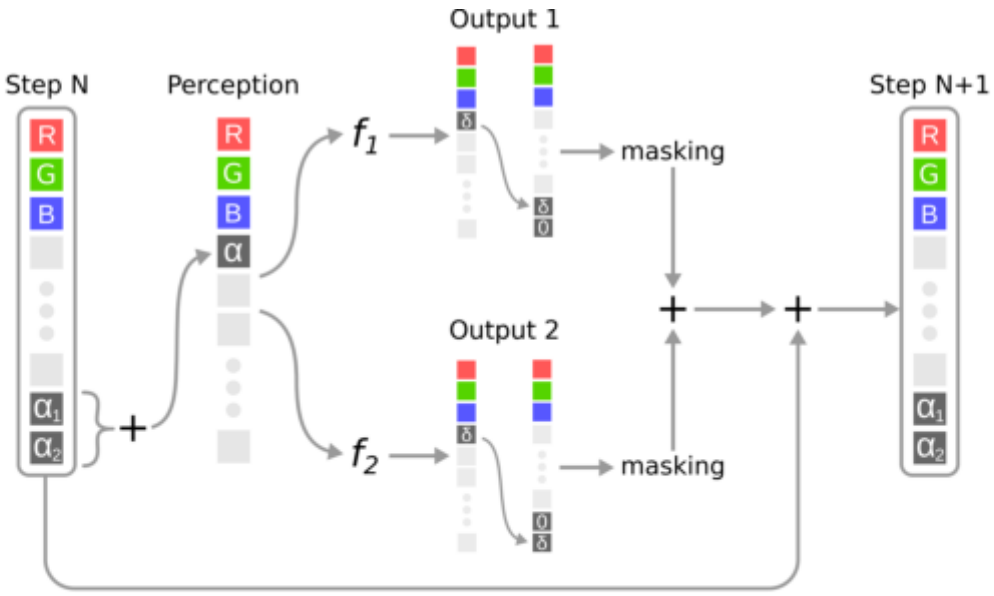


Figure 9: Illustration of the entire forward pass of the Multiple NCA model. f_1 and f_2 represent the two NCA rules, while masking refers to the 3 operations described in the Masking subsection.

Results In figure 12 we plotted the evolution of the NCAs, the time indicates the number of steps since the adversaries are first injected, we always inject the adversaries in a 2x2 square randomly located inside the organism. As you can see, the adversaries learn to influence the original cells to undergo the process of apoptosis, thus taking over the whole organism. Furthermore, as they are expanding, they try to match the color and target shape.

From a biological point of view, we think that adding the adversaries in a small square is more interesting than using, for example, a spray pattern (figure 11), because it shows that we can edit the cells in a single location then the change propagates throughout the body.

Figure 11: Examples of spray patterns with different percentages of adversaries, in blue, of respectively, $5\%, 25\%$ and 50%.

From a practical perspective, we noticed that training a model with adversaries that start from a small square is much harder than training starting from a spray pattern (figure 11), however, models that are able to take over starting from a small square generalize well to spray patterns, while the opposite isn't true, these results can be found here.

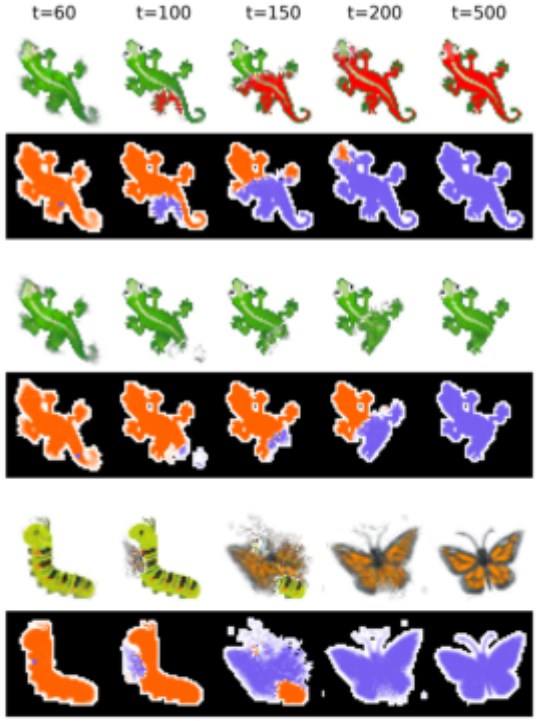


Figure 12: Results of the static experiments, first we plot the evolution of the organism, and right below it the cell mask, which tells where the original (orange) and the adversarial (blue) cells are located. t refers to the number of steps since the beginning of the evolution, we inject the adversaries always at step 60. We encourage the reader to take a look at the videos of this evolution at `https://letteraunica.github.io/neural_cellular_automata/extra#static-properties`

Changing dynamic properties

Changing a dynamic property means to change the whole NCA evolution, for example avoiding the decayment of a Growing NCA (figure 13). In this part we focus on turning 3 Growing NCA into Persistent ones, this is much harder than the previous experiment because the organism remains in the final state only for a limited amount of time, so the adversaries have to take over the whole organism before it decays.

Results In figure 13 we plot the evolution of 3 Growing NCA before we turn them into Persisting ones, in all cases we perform the adversarial injection at step t=60, which corresponds to the time the Growing NCA reaches the target state.

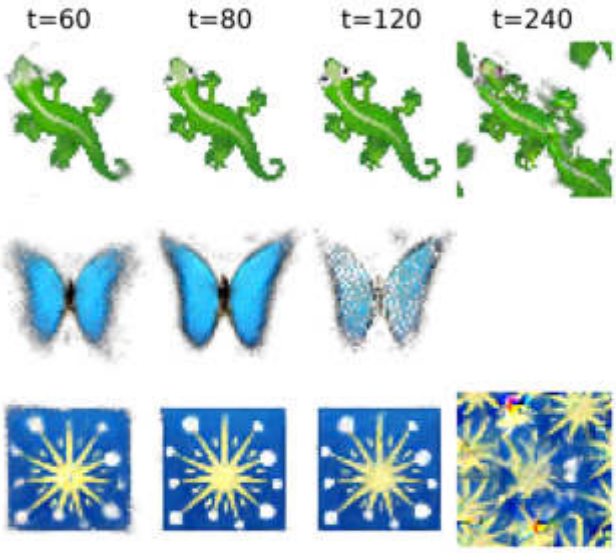


Figure 13: Growing NCA rules that we used in the dynamic experiment. After some steps they degenerate.

The Growing lizard was the easiest to turn into Persistent, because it decays at about step 200. This leaves about 140 steps for the adversaries to take over the organism, which are sufficient even when injecting the adversaries in a small 3x3 square seed (figure 14).

The butterfly was a little harder. In this case the organism decays by vanishing at about step 120. We see that the adversaries exploit this feature and take over the organism exactly when it vanishes. We weren't able to train this NCA with a square seed, which indicates the difficulty of the task. Still, we managed to reduce the initial number of adversaries to a very low number, only 3% of the total cells.

Finally, to turn a Growing NCA of the firework into a Persistent one, we needed about 50% of cells substituted, much higher than the lizard and the butterfly. We think this is due to 2 factors:

1. Unlike the butterfly, the firework decays by exploding rather than vanishing (figure 13).

2. Unlike the lizard, the firework decays at about step 140 (figure 13), since we perform the adversarial injection at step 60, this leaves only 80 steps for the adversaries to take over.

These two factors combined mean that the adversaries must take over before the growing cells explode. However, this time is very limited, which leads to a high initial number of adversaries.

From these experiments we can hypothesize that the more time it takes for the organism to decay the lower the initial percentage of adversaries is needed.

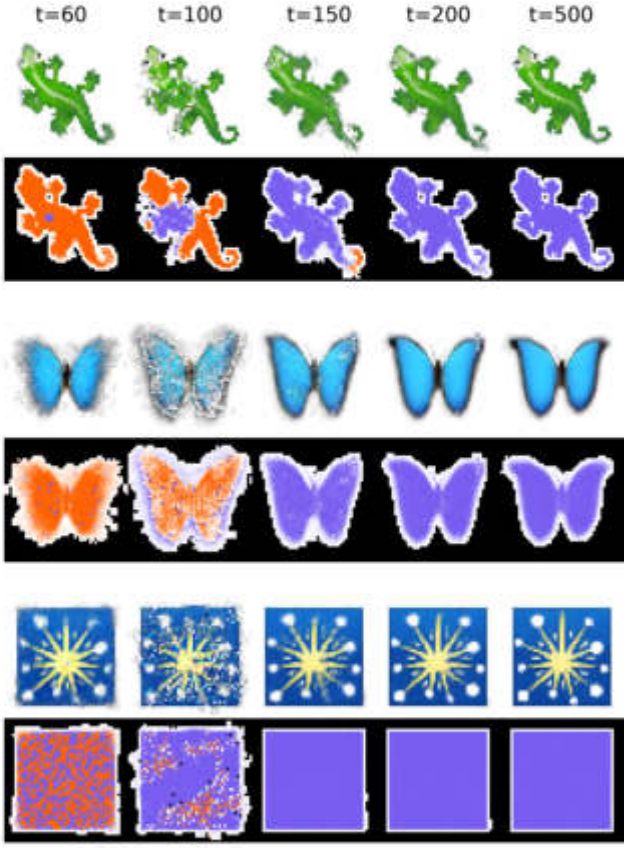


Figure 14: Results of the dynamic experiments, as before, first we plot the evolution of the organism, and right below it the cell mask, which tells where the original (orange) and the adversarial (blue) cells are located. t refers to the number of steps since the beginning of the evolution, we inject the adversaries always at step 60. We encourage the reader to take a look at the videos of this evolution at `https://letteraunica.github.io/neural_cellular_automata/extra#dynamic-properties`

Adding a perturbation

Iterated maps, like cellular automata and differential equations, oftentimes lead to chaotic systems. This implies that small changes to the initial conditions or to the function parameters, will lead to completely different results after some time Berto and Tagliabue (2022).

This is a double-edged sword:

- On one hand, chaotic systems, by definition, are very hard to predict and understand.
- On the other hand, lying at the edge of chaos gives us the power of influencing the system by a lot, with very little changes to its parameters Berto and Tagliabue (2022). Mother nature knows this very well, for example, humans have 99% of the DNA in common with chimpanzees, yet we are very different from them.

In the previous paragraphs, when training the adversaries, oftentimes the parameters become widely different from the ones of the original cells. This is a problem, because in a real organism we would like to edit the cells as little as possible. In this section we try to fix this problem by finding adversaries with parameters close to the original NCA parameters.

The model

As we said, we'd like to have adversaries with weights that are only a little perturbation off the original ones:

$$w_{new} = w_{old} + \Delta w$$

To be sure that the perturbation Δw remains as small as possible, we added an additional term in the loss, which penalizes the L^2 norm of the perturbation $|\Delta w|_2$, so the total loss will be:

$$L = L_{target} + \lambda_1 N_{old} + \lambda_2 |\Delta w|_2$$

Where L_{target} is the distance to the target image and λ_1, λ_2 are hyperparameters. We used two different metrics to evaluate the results, the norm of the perturbation $|\Delta w|_2$ and the the cosine similarity between w_{old} and w_{new}.

$$cos(w_{old}, w_{new}) = \frac{\langle w_{old}, w_{new} \rangle}{|w_{old}||w_{new}|}$$

Results

In figure 15 you can see the metrics for turning a green Persistent lizard into a red Persistent lizard for lower and lower initial percentage of adversaries and for the following training regimes.

1. When training the adversaries starting from a random initialization of the weights and set $\lambda_2 = 0.01$
2. When training the adversaries starting from the same weights of the original cells, $w_{new} = w_{old}$, and set $\lambda_2 = 0.01$
3. When training the adversaries starting from the same weights of the original cells, $w_{new} = w_{old}$, and set $\lambda_2 = 0$
4. When training the adversaries starting from a random initialization of the weights and set $\lambda_2 = 0$

We train each model until it has a loss < 0.01, which we found was an appropriate value to have visually indistinguishable images.

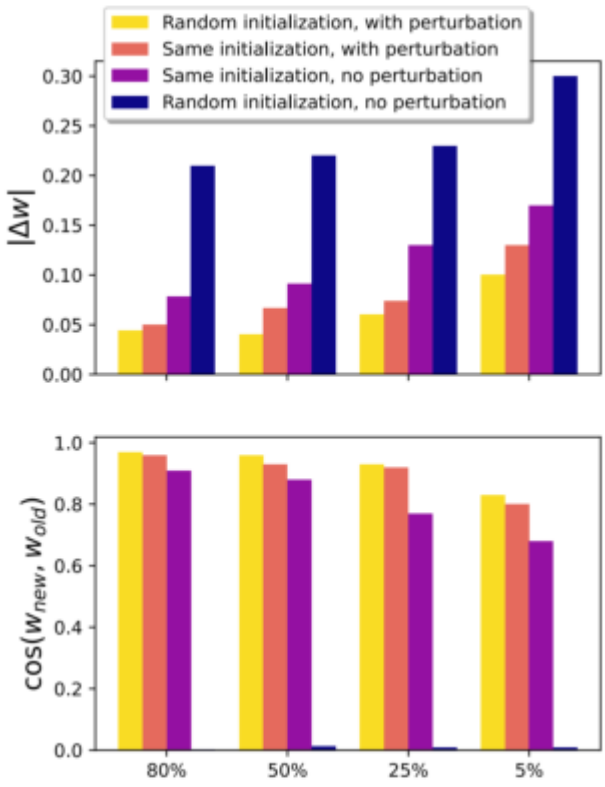


Figure 15: $|\Delta w|$ and cosine similarity measures in different training regimes. By penalizing $|\Delta w|$ we manage to find adversaries close to the original cells even when starting from a random initialization. Furthermore, as we decrease the number of initial cells, $|\Delta w|$ increases and the cosine similarity decreases.

Related Work

The paper *Adversarial Reprogramming of Neural Cellular Automata* Randazzo et al. (2021) laid the foundations for this work, while the talk given by Michael Levin at NeurIPS 2018 Levin (2018) provided biological insight and ideas for multiple experiments.

The field of Neural Cellular Automata is already vast, NCAs have been used in texture generation Niklasson et al. (2021), image classification Randazzo et al. (2020), and image segmentation Sandler et al. (2020). Furthermore, researchers managed to find the NCA that converges to a given image without training the NCA Chen and Wang (2020), in other words, the authors use a neural net to encode an image in the weights of a NCA. Other relevant works include the application of NCA to reaction-diffusion systems Mordvintsev et al. (2021), in this case the learned rule is more general because it doesn't depend on the structure of the grid, this allows a NCA trained on a 2D grid to be used on different geometries.

The kinds of adversarial attacks shown in this paper stem from the Generative Advesarial Networks Goodfellow et al. (2014) area of research and in particular *Adversarial Reprogramming of Neural Netowrks* Elsayed et al. (2018), where a target model is kept frozen while ad-hoc inputs are used to change the functional behaviour of the original model.

Additionally, computer-to-in-vivo experiments were conducted in which organisms were developed from scratch to perform specific tasks Kriegman et al. (2020). Other examples of significantly modifying anatomical outcomes without altering the genome include lines of flatworms that regenerate with two heads following alteration of bioelectric signaling Durant F (2016).

Conclusion

We demonstrated that it is possible to change global properties of a Neural Cellular Automata, by injecting very few adversaries that gradually take over the entire organism. For some tasks, where the time is a major factor, a larger injection of adversaries will be needed; for example, if we want to make a mortal organism immortal, the adversaries must be able to take over before the organism dies. Finally, we showed that the parameters of the adversaries can be chosen to be close to the original cells that they replace.

The many similarities between Neural Cellular Automata and real biological organisms, may indicate that a more refined technique could be used to bioengineer already living organisms.

Acknowledgments

We thank Alessio Ansuini for his feedback and support.

References

Berto, F. and Tagliabue, J. (2022). Cellular Automata. In Zalta, E. N., editor, *The Stanford Encyclopedia of Philosophy*. Metaphysics Research Lab, Stanford University, Spring 2022 edition.

Biswas S, A. S. (2016). Neural model of gene regulatory network: a survey on supportive meta-heuristics. *Theory Biosci*. https://pubmed.ncbi.nlm.nih.gov/27048512/.

Chen, M. and Wang, Z. (2020). Image generation with neural cellular automatas.

Durant F, Lobo D, H. J. L. M. (2016). Physiological controls of large-scale patterning in planarian regeneration: a molecular and computational perspective on growth and form.

Elsayed, G. F., Goodfellow, I., and Sohl-Dickstein, J. (2018). Adversarial reprogramming of neural networks.

Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014). Generative adversarial networks.

Kriegman, S., Blackiston, D., Levin, M., and Bongard, J. (2020). A scalable pipeline for designing reconfigurable organisms. *Proceedings of the National Academy of Sciences*, 117(4):1853–1859.

Levin, M. (2018). What bodies think about: Bioelectric computation outside the nervous system. NeurIPS 2018.

Lobo D, Solano M, B. G. L. M. (2014). A linear-encoding model explains the variability of the target morphology in regeneration.

Moore, D. G., Walker, S. I., and Levin, M. (2018). Pattern regeneration in coupled networks. ALIFE 2018: The 2018 Conference on Artificial Life:204–205.

Mordvintsev, A., Randazzo, E., and Niklasson, E. (2021). Differentiable programming of reaction-diffusion patterns.

Mordvintsev, A., Randazzo, E., Niklasson, E., and Levin, M. (2020). Growing neural cellular automata. *Distill*. https://distill.pub/2020/growing-ca.

Nanos V, L. M. (2021). Multi-scale chimerism: An experimental window on the algorithms of anatomical control. *Cells Dev.* Epub ahead of print, https://pubmed.ncbi.nlm.nih.gov/34974205/.

Neumann, J. V. and Burks, A. W. (1966). *Theory of Self-Reproducing Automata*. University of Illinois Press, USA.

Niklasson, E., Mordvintsev, A., Randazzo, E., and Levin, M. (2021). Self-organising textures. *Distill*.

Randazzo, E., Mordvintsev, A., Niklasson, E., and Levin, M. (2021). Adversarial reprogramming of neural cellular automata. *Distill*. https://distill.pub/selforg/2021/adversarial.

Randazzo, E., Mordvintsev, A., Niklasson, E., Levin, M., and Greydanus, S. (2020). Self-classifying mnist digits. *Distill*. https://distill.pub/2020/selforg/mnist.

Sandler, M., Zhmoginov, A., Luo, L., Mordvintsev, A., Randazzo, E., and y Arcas, B. A. (2020). Image segmentation via cellular automata.

Exploiting Intrinsic Multi-Agent Heterogeneity for Spatial Interference Reduction in an Idealised Foraging Task

Chris Bennett[1] , Jonathan Lawry[1] and Seth Bullock [2]

[1] Department of Engineering Maths, University of Bristol, Bristol, UK
[2] Department of Computer Science, University of Bristol, Bristol, UK
christopher.bennett@bristol.ac.uk

Abstract

Typically, collective behaviour research has tended to focus on behaviour arising in populations of *homogeneous* agents. However, humans, animals, robots and software agents typically exhibit various forms of *heterogeneity*. In natural systems, this heterogeneity has often been associated with improved performance. In this work, we ask whether spatial interference within a population of co-operating mobile agents can be managed effectively via conflict resolution mechanisms that exploit the population's intrinsic heterogeneity. An idealised model of foraging is presented in which a population of simulated ant-like agents is tasked with making as many journeys as possible back and forth along a route that includes tunnels that are wide enough for only one agent. Four conflict resolution schemes are used for determining which agent has priority when two or more meet within a tunnel. These schemes are tested in the context of heterogeneous populations of varying size. The findings demonstrate that a conflict resolution mechanism that exploits agent heterogeneity can achieve a significant reduction in the impact of spatial interference. However, whether or not a particular scheme is successful depends on how the heterogeneity that it exploits is implicated in the population-wide dynamics that underpin system-level performance.

Introduction

Alife has explored collective behaviour in populations of simple agents. Often these agents are homogeneous in order to keep models tractable. However, in nature heterogeneity is a key property of many systems that exhibit interesting or sophisticated collective behaviour. For example, biodiversity has been linked to increased stability and productivity of an ecosystem by improving its resilience to environmental effects (Tilman et al., 2006; Hooper et al., 2005). Individual differences in a group can improve collective problem solving in humans (Page, 2008) and behavioural differences, together with social connectivity, can change the attack speed of social spiders (Hunt et al., 2019).

This paper explores one particular aspect of this wide ranging topic by taking the concepts of functional diversity and fitness and asking if they can be exploited to create a better engineered system. Specifically we consider whether linking the "fitness" of an agent to its priority (status) in the system produces better overall system performance. How do different schemes for assigning priority affect the results? And, how does increased functional diversity (heterogeneity) in the system change the behaviour?

Asking these questions in this specific case contributes to the wider aim of finding design rules for exploiting the right kind of heterogeneity in multi-agent systems, thereby moving away from the view that regards all heterogeneity as a potential problem in engineered systems.

We begin with a brief discussion of interference in multi-agent systems, followed by an ecological definition of fitness as arising from a combination of extrinsic and intrinsic traits, and consider the hypothesis that giving priority to the fittest agent during local interference interactions will improve system level performance. To test these concepts, four different mechanisms for assigning priority are described. These are then evaluated within a simple simulation environment in which a population of heterogeneous ant-like agents are tasked with repeatedly navigating to a source of food and carrying the food home along a route with narrow tunnels. The findings show that the performance achieved using each mechanism is sensitive to both the environment and the heterogeneous traits of the population.

Interference

Interactions between agents co-operating on a task can be broadly classified as being either beneficial or detrimental to the overall function of the group. Here, we focus on "same place, same time" interference (Goldberg and Matarić, 1997) where two or more agents are not able to access the same limited resource at the same time. This form of interference is present in a wide range of scenarios including organisms foraging for food (Vahl et al., 2005) and robots navigating cluttered environments (Trautman and Krause, 2010). The specific question addressed in this paper is how best to resolve same place, same time interference between two or more agents that may differ in some of their intrinsic characteristics.

One approach is to limit or avoid spatial interference completely by suitably partitioning the environment (and task),

for instance via the use of a hand over area to avoid foraging agents crowding the "nest" (Pini et al., 2009) or dividing the space and assigning a single agent to operate in each spatial zone (Schneider-Fontán and Matarić, 1998). This is comparable to the approach taken by territorial animals (Bullock, 2016). However, avoiding other agents is not always possible, and in these scenarios, a strategy is needed for deciding which agent has priority.

Agents that are obstructing one another could engage in agonistic interactions in order to determine which should back down. However, it is likely to be less costly to resolve the impasse by employing a conflict resolution mechanism that avoids direct combat. One approach to deciding priority is to base it on a trait of the individual which can be attributed to the likely future success of the population. In nature, this trait may be based on physical characteristics such as body size and success described quantitatively as the fitness of the individual (Blanckenhorn, 2000).

When we create artificial systems, for example a team of robots, the concepts of fitness and traits are no less relevant but can be harder to define. When the goal is to maximise team performance then an analogy to functional traits may be appropriate (Petchey and Gaston, 2006). These are a subset of the agent's traits which affect the performance of the ecosystem. While fitness may not be directly applicable without a mechanism for reproduction, researchers have developed proxies for it in robotic systems by relating a robot's current state to its contribution to a task. In Mayya et al. (2019), a swarm distribution task algorithm is proposed which is designed to reduce spatial interference by enabling robots in densely packed regions to judge when it is more beneficial to move away and make space than it is to stay and participate in the task. Alternatively, in Brown et al. (2005) the amount an agent has invested in the task is used to assign higher priority to an agent if it has more to lose by giving way to another agent. In both these approaches, the proxy for the fitness of each agent is a function of its current state. All agents are intrinsically identical - they only differ in terms of some transient properties related to their current circumstances or history of interactions with the world.

Contrasting these perspectives from ecology and engineering highlights that a fitness proxy may either be related to intrinsic properties of the individual (weight, height, speed...) or extrinsic properties (task investment, personal space, level of aggression..). Previous work in the multi-agent literature on agent interference has often used physically homogeneous agents. In Vaughan et al. (2000), two homogeneous robots would attempt to pass though the same narrow opening. It was found that a proxy for "agression" was effective at resolving interference and increased the collective performance of the population. Surprisingly, there was no significant difference between assigning a robot's "aggression" at random vs a fixed hierarchy. The authors speculated that if a functional difference between the robots had existed (i.e., the population had been heterogeneous), then some mechanisms for assigning priority, based on a proxy for fitness, would be more beneficial than others.

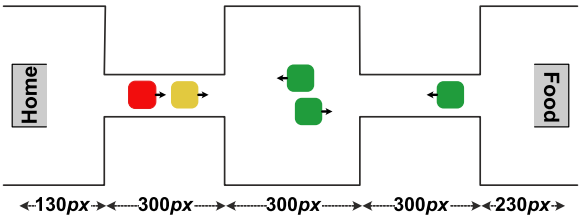


Figure 1: The experimental environment showing two tunnels and five ant-like agents foraging for food. Green agents are currently in the Navigating state, the yellow agent currently is Afraid and the Red agent currently is Brave.

Here, we build on these prior works via two experiments using heterogeneous populations which test the utility of different intrinsic properties for resolving interference. The hypothesis is that for the purposes of resolving "same place, same time" interference, the overall performance of the group is improved when the conflict resolution is informed by a proxy for fitness that determines an agent's priority. In the first experiment, speed is taken as the proxy since it directly affects how much food an agent can transport in a given time. In experiment two, the proxy is based on the maximum sensor range within which an ant can detect other ants that it is in conflict with. Through these two experiments, we explore the way in which the nature of some heterogeneity in an agent's intrinsic properties impacts the efficacy of conflict resolution mechanisms intended to reduce the impact of inter-agent interference.

Experimental Approach

Foraging is a popular scenario in the multi-agent and swarm literature because it can be implemented using simple behaviours for each agent while still creating interesting results relevant to the real-world (Brambilla et al., 2013; Liu and Winfield, 2010; Dugatkin, 2002; Pitonakova et al., 2018). Here, we simulate a small population of ant-like organisms tasked with transporting food from one location to another along a route that includes a narrow tunnel (fig. 1). The virtual ants can pass each other without penalty outside the tunnel, but inside the tunnel interference occurs, preventing one agent from overtaking a slower agent or moving past an agent that is approaching from the opposite direction. The tunnel is only a single lane wide and therefore the ants must decide who will back up and give way to avoid a deadlock. Note that when a faster ant approaches a slower ant and both are travelling in the same direction, both ants continue along their current path at the speed of the slower ant.

If two ants are identical, then it can be presumed they would take the same action when they meet travelling in op-

posite directions. That is, either both back up or both go forward. To avoid a deadlock, there needs to be some symmetry breaking (or diversity) between the agents. In the following we consider two situations in which agents differ in terms of their priority and their functional traits.

Assigning Priority

Four simple mechanisms for assigning priority are as follows:

- **Arbitrary (Transient):** Each agent is assigned a unique random value when it enters a conflict. This value is lost when the conflict is resolved. During a conflict, the agent with the highest value assumes priority.
- **Arbitrary (Fixed):** Each agent is assigned a unique random value at the start of the simulation. In a conflict, the agent with the highest value assumes priority.
- **Fittest First:** Each agent is assigned a value based on a proxy for fitness. In a conflict, the agent with the highest value assumes priority.
- **Fittest Last:** Each agent is assigned a value based on a proxy for fitness. In a conflict, the agent with the *lowest* value assumes priority.

Assigning Functional Traits

From an ecological perspective, a functional trait is a measurable characteristic of an organism that can be linked to the overall function of the ecosystem (Petchey and Gaston, 2006). Here we take a simplified approach and assign each ant a unique set of traits. In experiment one, the trait is speed and there is an implicit assumption that a faster ant will be "fitter" since it has the potential to transport more food over a given time period. In each simulation, speeds were assigned to agents such that the mean population speed remains constant across all scenarios. This avoids skewing the results to favour populations containing agents with a higher average speed. In the idealised case where no spatial interference occurs, this causes the total food collected to increase (linearly) with the number of ants, but the average food collected per ant to remain constant (see fig. 2).

Ecosystem Concepts of Diversity

The experiment uses heterogeneous populations and therefore a workable definition of diversity is needed. The diversity of a population can be defined based on species richness (the number of types present), evenness (how many of each type are present) and divergence (the difference between types) (Mason et al., 2005). An agent type is defined via its traits. In each simulation, each agent has a unique set of traits and therefore richness is varied by changing the number of ants present with evenness remaining constant.

Divergence is a measure of how different two agents are. Metrics that address this for two individuals are often based on some form of difference norm. Compound metrics extend this measure of divergence by combining it with richness and evenness, to give a single number representing how diverse (heterogeneous) a population is. A discussion of the different classifications, norms and metrics together with their relative merits is given by Daly et al. (2018) with particular examples given in Balch (2000); Twu et al. (2014). Here, divergence is varied primarily by changing the separation of speed values within a population. For example in a population of 3 agents with velocities $V = \{35, 40, 45\}$, the mean velocity is $\bar{v} = 40$ and the separation is $\Delta v = 5$. If the velocities were $V = \{30, 40, 50\}$,the mean velocity remains $\bar{v} = 40$ and the separation is now $\Delta v = 10$. In the experiments reported here, we keep the number of different speeds equal to the number of agents and manipulate only the range of speeds, keeping the mean speed constant. (fig. 2).

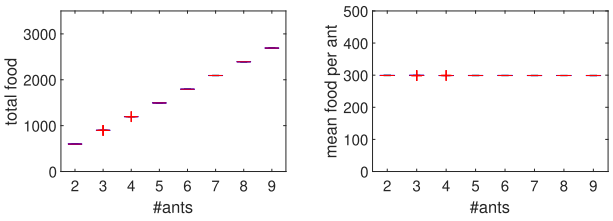


Figure 2: Boxplots of the total food transported in experiment 1 with no interference. This is a maximum under idealised conditions. Mean Speed is 50. Speed range is 10. *Left:* total food. *Right:* total food normalised by the number of agents. Note, speeds are chosen such that the average remains constant for all population sizes.

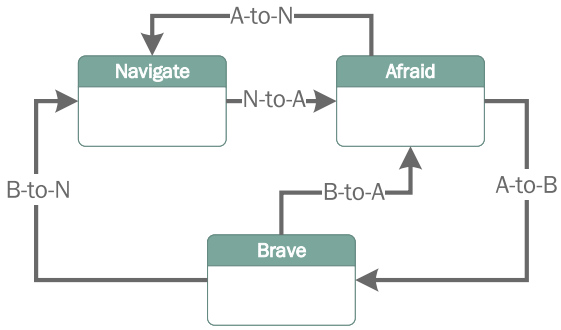
Simulation Environment

The simulation environment was written in python and is available for download from the University of Bristol's data repository (Bennett et al., 2022)[1].

Finite State Machine A FSM is used to control the behaviour of an ant transitioning between transporting food (Navigating), backing up (Afraid) and advancing with priority over any afraid ants ahead (Brave). The transitions between the states are described by the state machine and table shown in figure 3.

When entering the Afraid state, the ant calculates its priority according to the conflict resolution mechanism being implemented. An ant A maintains a list of other ants that it is currently in conflict with and will transition to Brave if it has the highest priority. New ants are added to A's list if they move within conflict range of A, are in the tunnel, and are on a collision course with A. An ant is removed from A's conflict list if either 1) if is outside the passive range of A, 2) it is outside the tunnel and A is Brave, or 3) A is Afraid and

[1]https://doi.org/10.5523/bris.3ota45j8tbx332daufr3vp35ad



Transition	Conditions
N-to-A	**not** isEmpty(conflict_list)
A-to-N	**not** in tunnel **and** isEmpty(conflict_list)
A-to-B	my_priority > max(conflict_list)
B-to-A*	my_priority < max(conflict_list)
B-to-N	isEmpty(conflict_list) **or not** in tunnel

Figure 3: State machine used by an ant to transport food, back away from another agent when afraid and move forward after winning a contest and becoming brave. *An agent does not re-calculate its priority when it transitions from state Brave to Afraid

more than 20px outside the tunnel. The list is cleared when the agent transitions to the Navigate state.

The condition for transitioning from Afraid to Navigate is designed to ensure an Afraid ant reverses far enough to enable other ants to clear the tunnel and allow a Brave ant to exit. Similarly, the passive range is always larger than the conflict range to prevent an agent being rapidly added to and removed from the conflict list.

Collision Avoidance A simple mechanism is used to prevent two or more ants occupying the same space while in a tunnel. An ant will move at $u(t) \leftarrow max\_speed * dt$ provided reaching the position at $p(t + dt) = p(t) + u(t)$ does not collide with, or pass through, another ant. If it does then $u(t) \leftarrow 0$ and the ant doesn't move.

Simulation Parameters

The fixed parameters are given in table 1. The independent variables are the number of ants ($[2, 9]$), the functional diversity (set to either a speed separation of $\Delta v = 5$ or $\Delta v = 10$) and the scenario for assigning priority.

Conflict Range The visible range of an ant is divided into three segments determining: when ants are visible (Detection Range), when they are close enough to cause a conflict (Conflict Range), and when they are far enough away to be removed from a conflict (Passive Range). If the conflict

Parameter	Value	Units
Time Limit	75000	sec
Step Size (dt	0.1	sec/tick
Number of repeats	50	–
Detection Range [min,max]	[50,350]	px
Conflict Range [min,max]	[20,350]	px
Passive Range [min,max]	[40,350]	px
Range Growth Rate	+5	px/tick
Range Decay Rate	-2	px/tick
Average Population Speed	50	px/tick
Speed Difference (Δv)	{5,10}	px/tick
Number of Repeats	50	–
Arena Size	1280	pxwide
Agent Size	10	wide px

Table 1: Simulation parameters. Agents move along a fixed line parallel with the x-axis. Measurements in pixels (px). The range of speeds in a population of N is $\Delta v N$.

range is too small then unrecoverable deadlocks can occur (see figure 4). Such deadlocks are resolved by gradually increasing an ant's Conflict, Passive and Detection ranges by +5 for each tick during which the ant is stationary. The ranges then decrease by -2 for each tick during which the ant is moving. A minimum and maximum for these ranges was set and is given in table 1. Agents are initialised at their minimum range values. In the first experiment these limits are the same for the all agents, but in the second experiment each agent has different maximum ranges.

Update Method At $t = 0$, the ants are distributed along the route and assigned a random direction of travel. Updates are performed asynchronously (all agents take a turn to sense and move) and the turn order is randomised at the start of each time step to avoid the possibility of introducing a systematic "first mover advantage".

Step Size The simulation step size was found to have a significant effect on the results with large step sizes (greater than 0.5) creating the possibility of artefacts caused by agents moving too far in a single step. These occurred more frequently when faster agents (and also larger numbers of agents) were present, and they could not be resolved by the combination of priority, state machine and heterogeneity in the population. This is a consequence of the rudimentary collision mechanics which allows a slower agent to move closer to other agents than a faster one. A step size of 0.1 was found to be a good compromise between preventing these artefacts and enabling the simulation to complete in a reasonable time. It is worth stressing that the simulation is designed to investigate heterogeneity and fitness rather than provide an accurate model of robot locomotion.

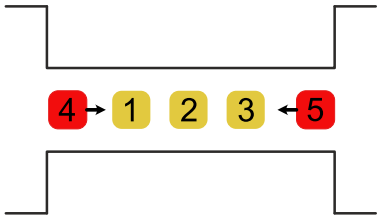


Figure 4: The conflict range is an important parameter for preventing deadlocks. In this example, the conflict range of 4 and 5 is too small so neither considers the other a source of interference and both agents assume priority.

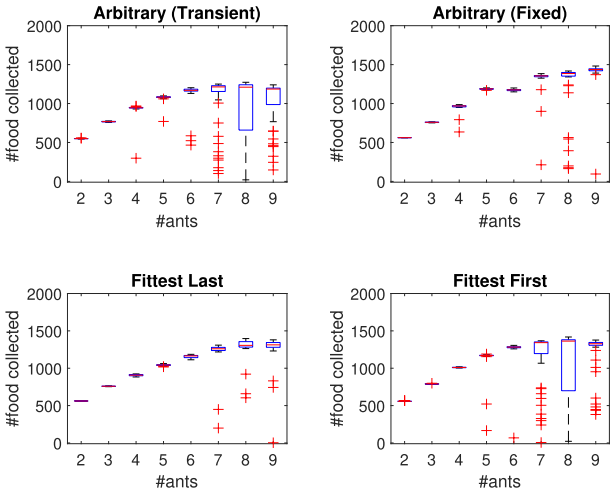


Figure 5: The total food collected for different population sizes with $\Delta v = 5$. The plots show the results from 50 simulation replicates per population size. The heterogeneous trait was speed. Outliers are denoted with a red cross, the box extends from the 25th to 75th percentiles and the red line represents the median. Whiskers extended to the most extreme data point that is not an outlier.

Experimental Results

Two experiments were undertaken with the aim of establishing whether prioritising agents based on a fitness-proxy could be an effective mechanism for resolving spatial interference. The first experiment uses a population with different speeds, the second experiment adds maximum sensor range as an additional heterogeneous trait which affects how many other agents an ant may consider during a conflict.

Experiment One: Heterogeneous Speeds

In the first experiment, the speed of each agent was unique to create a population that was heterogeneous in one intrinsic trait (speed). The average speed was maintained at 50 for each population size.

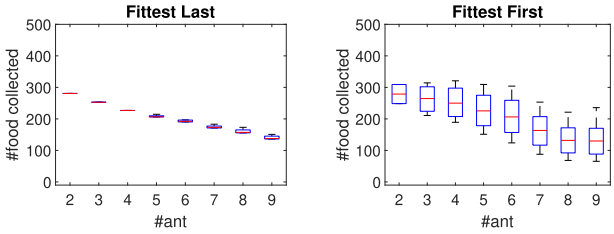


Figure 6: $\Delta v = 5$. The food collected by each agent in the population averaged over 50 independent simulations per population size. The heterogeneous trait was speed. *Left:* each ant in the population collected a similar amount of food when slower ants had priority. *Right:* prioritising faster ants produced a greater spread in the amounts collected across a population.

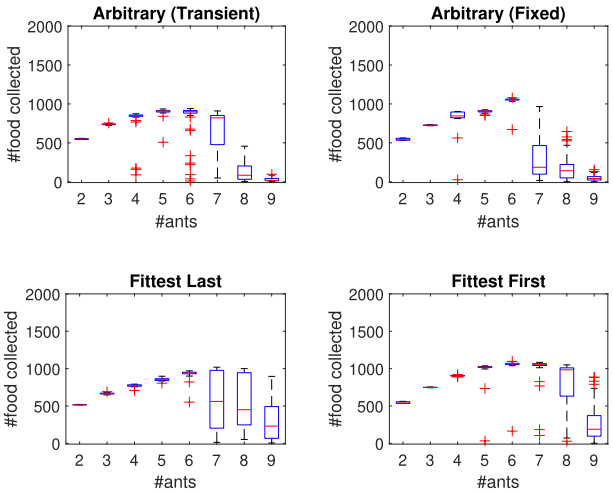


Figure 7: $\Delta v = 10$, The total food collected for different population sizes. The plots show the results from 50 independent simulations per population size. The heterogeneous trait was speed.

Results As the number of agents in the population is increased from 2 to 9, the total amount of food collected initially increases but at a slowing rate, with performance eventually deteriorating (and becoming more variable) beyond a certain population size (fig. 5). It might be expected that prioritising the fastest agent (Fittest First) would produce a population that collects substantially more food but this was not the case. Instead, the performance curves of the four scenarios tested were very similar, The performance, P_i, across the range of population sizes can be summarised as $P_i = \sum_{n=2}^{9} F_n^i$ where F_n^i is the total food collected for population size n during simulation replicate $i \in [1, 50]$. Taking the median of P_i for each priority-mechanism shows that Arbitrary Fixed (8807) collected the most food on average over the range of population sizes tested, followed by Fittest Last (8314), Fittest First (8185) and Arbitrary Tran-

sient (7635). A Wilcoxon rank test on the values of P_i revealed some subtleties. Firstly, the Arbitrary Transient results were significant ($p < 0.05$) in pair-wise tests with the other 3 scenarios indicating that a transient hierarchy collects less food on average than a fixed one. Of the fixed hierarchies, Arbitrary Fixed performed better than either Fittest First or Fittest Last ($p < 0.05$), and there was no difference between Fittest First and Fittest Last ($p > 0.05$). Collectively these results suggest a subtle benefit to not always prioritising either the faster or slower agent and to an agent maintaining the same priority over time.

Despite a faster agent having the potential to transport more food in a given time, prioritising speed as a proxy for fitness did not produce the expected gain in performance when resolving interference. This can be seen in the results obtained with 5 ants for Fittest First and Fittest Last where the fastest ant had the potential to move $1.5\times$ faster than the slowest ant, but giving faster ants priority only resulted in 12% more food being collected by the population.

One reason for this is that using speed as a proxy for fitness results in a near zero-sum scenario. This can be seen by plotting the average food collected by each ant in the population (fig. 6). Prioritising slower agents (Fittest Last) resulted in each ant collecting a similar amount of food. Slower agents were enabled to maximise their individual performance levels, but faster agents were penalised by being forced to move at the speed of slower agents. A greater range of individual performance levels was seen when priority was given to faster agents (Fittest First), but the overall population performance remained roughly similar because improvements for faster agents were almost entirely compensated for by reduced performance levels for slower agents.

Increasing the diversity of the population by doubling the separation between agent speeds within the population decreased performance in all scenarios (fig. 7) which suggests the conflict resolution mechanisms do not scale well to increasing heterogeneity of speed. Calculating the median of the P_i values for each priority mechanism showed Fittest First (6432) now collected more food than the other mechanisms (Wilcoxon rank test on P_i, $p < 0.05$). There was no difference ($p > 0.05$) between Arbitrary Fixed (4470) and Arbitrary Transient (4796), and both these performed worse than Fittest Last (5208). This suggests that there *is* a benefit to prioritising an agent based on its speed when the range of speeds in the population increases above a threshold, and that benefit is greater when priority is given to the faster agent.

It is also interesting to compare the shape of figures 5 and 7 to the hypothetical curve proposed by Vaughan et al. (2000) and reproduced in fig. 8. Doubling the diversity reduced the height of the peak performance and it occurred at smaller population sizes. Also, the performance did not roll off smoothly as the population size increased. Instead

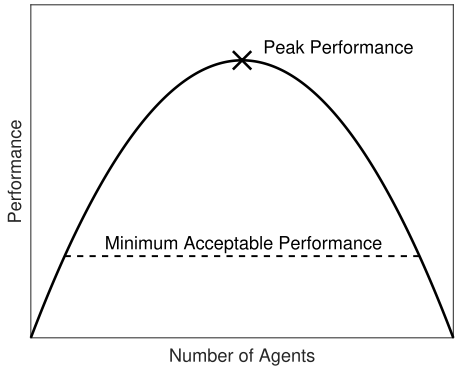


Figure 8: Hypothetical relationship between the size of a population and its performance in the presence of inter-agent interference.

there is a large drop in performance at a population size in the range $[7, 9]$ depending on the priority-mechanism. The curves with lower diversity (fig. 5) suggest a similar shape to the hypothetical curve in figure 8 but the drop in performance did not occur below nine agents.

Experiment Two: Heterogeneous Ranges

Similar to the findings for homogeneous robots reported in Vaughan et al. (2000), the mechanisms for assigning priority gave similar results when speed was used as a proxy for fitness and diversity in the population was low. One possible reason for this is that although agent speed would appear to have a positive effect on task performance, the nature of the task and environment ensured that interference damped the benefit of one agent being faster than another. The significance of a functional trait being dependent on the environment has been seen in the natural world where researchers have compiled lists of which functional traits are significant for particular ecosystems (see Stubbs and Wilson (2004)).

In experiment two, maximum sensor range was chosen as a new trait and proxy for fitness because it is closely coupled to the conflict resolution mechanic. Its effect is to change the maximum number of other agents that an ant can consider during conflict resolution. Ants with a higher maximum have the potential to appreciate the full range of antagonists within larger conflicts. The assumption was that overall system performance would be more sensitive to conflict resolution mechanisms that prioritised this property.

When an ant joins a contest, its sensing range is 10px ensuring that the ant will only be in conflict with its immediate neighbours. This range increases by 5px for every tick the agent remains stationary and decreases by 2px per tick that the agent is moving (down to a minimum of 10px). Therefore, the longer an ant remains stationary, the more ants it can include in its conflict list up to a maximum determined by its assigned maximum sensor range. Figure 4 shows that

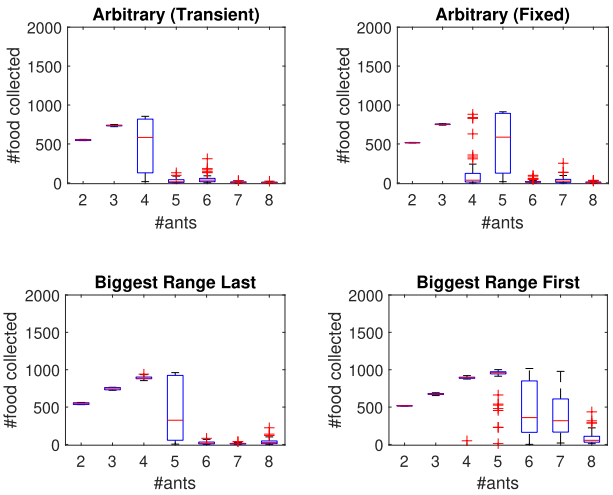


Figure 9: $\Delta v = 10$, The total food collected for different population sizes where priority is assigned based only on maximum sensing range. The plots show the results averaged over 50 repeats of the simulation. Speed is randomised.

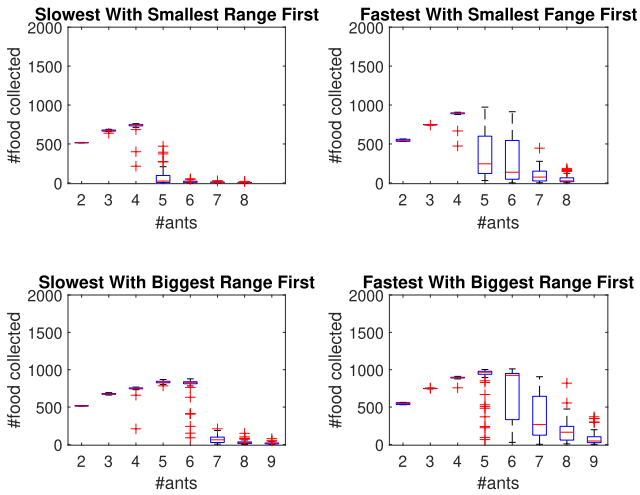


Figure 10: $\Delta v = 10$, The total food collected for different population sizes where priority is assigned based on maximum sensing range and speed. The plots show the results averaged over 50 repeats of the simulation.

if this maximum sensor range is too small then agents may be unable to "see" other antagonists involved in a conflict, resulting in multiple agents becoming Brave and causing a deadlock. In experiment one, the maximum range was the same for all agents (350px) whereas in experiment two each agent was assigned a unique integer value from the range $[20, 10(n+1)]$ where n is the population size. This created a total of two functional traits in experiment two: speed and maximum sensor range.

Results Figure 9 shows the results of assigning priority based on sensor range as a proxy for fitness. The allocation of a unique speed to each agent is randomised in each simulation. Each of the four scenarios gave a very different set of results, both in terms of the population size for which the most food was collected and the number of ants at which the performance deteriorated. This indicates that performance is more sensitive to sensor range than speed as a fitness-proxy.

Calculating the median of the 50 P_i values for each priority-mechanism and performing a significance test (Wilcoxon rank test on P_i, $p < 0.05$) showed that Biggest First (3910) was the highest performing followed by Biggest Range Last (2617), Arbitrary Transient (2049) and Arbitrary Fixed (1995). The difference between Arbitrary Transient and Arbitrary Fixed was not significant ($p > 0.05$). This indicates that there was a clear advantage to resolving interference using an ordered hierarchy based on maximum sensor range, and there was a further advantage if the hierarchy prioritised agents that could consider more (rather than fewer) agents during a conflict. Interestingly, the significance and ordering of the priority-mechanisms was the same as seen in the $\Delta v = 10$ case for experiment one. This may indicate that a low sensitivity trait (e.g. speed) can display the same priority-behaviour as a high sensitivity trait when the diversity in the population passes a threshold.

Figure 10 considers scenarios in which speed and maximum sensor range are related together in the population, e.g., agents with longer sensor range are also faster or agents with longer sensor range are also slower.

Again, ordering the scenarios based on the median of the P_i values showed that Fastest With Biggest Range First (4300) collected the most food on average across the range of $[2, 9]$ agents. This was followed by Slowest With Biggest Range First (3686), Fastest With Smallest Range First (3061), and Slowest With Smallest Range First (1988). Each pairing of these results was significant at $p < 0.05$. These results show a positive benefit to prioritising maximum sensor range *and* speed. The results for Slowest with Biggest Range First are the second best performing which adds further evidence to the observation that the mechanism for assigning priority is more sensitive to maximum sensor range than speed.

Summary

In this work, we aimed to test an approach to resolving "same place, same time" spatial interference based on the fitness of an agent. The hypothesis was that assigning priority to an agent that was "fitter" (i.e., higher performing) at the task would produce the best system-level performance. The results demonstrated that this was possible but required an understanding of the environment and inter-agent dynamics. In particular, performance was found to be more sensitive to certain traits, which mirrors how functional traits in natural systems are related to the specific ecosystem that a species inhabits. For the foraging problem presented, the results showed that sensing range (a trait that determined the

maximum number of agents that could be considered within a contest), was more significant as a conflict resolution mediator than the speed of an agent. Assigning priority based on a positive correlation between sensing range and speed was found to amplify the effect, demonstrating that a less significant intrinsic trait can amplify the effect of a significant one. Finally, prioritising the "fitter" individual was found to have a greater relative benefit when the diversity (heterogeneity) in the population increased.

Further Work

This work focused on two traits (speed and sensing range) that are intrinsic to the agent. Previous work has considered either similarly intrinsic properties (Hunt et al., 2019) or more extrinsic properties (Jacyno et al., 2009). Future work should consider how the combination of heterogeneity in both extrinsic and intrinsic agent properties might be exploited within the same multi-agent population.

Increasing the diversity of the population decreased the total amount of food collected for larger population sizes. This raises the issue of relating the way in which a system scales with population *size* to the way that it scales with population *diversity*. What design and analysis tools and techniques could be used to predict and potentially exploit this kind of dynamic variability in a population's heterogeneity?

Acknowledgements

This work was funded and delivered in partnership between the Thales Group and the University of Bristol, and with the support of the UK Engineering and Physical Sciences Research Council Grant Award EP/R004757/1 entitled 'Thales-Bristol Partnership in Hybrid Autonomous Systems Engineering (T-B PHASE)'.

This work was carried out using the computational facilities of the Advanced Computing Research Centre, University of Bristol (http://www.bristol.ac.uk/acrc/).

References

Balch, T. (2000). Hierarchic social entropy: An information theoretic measure of robot group diversity. *Autonomous Robots*, 8(3):209–237.

Bennett, C., Bullock, S., and Lawry, J. (2022). A model to investigate exploiting intrinsic multi-agent heterogeneity for spatial interference reduction in an idealised foraging task. https://doi.org/10.5523/bris.3ota45j8tbx332daufr3vp35ad.

Blanckenhorn, W. U. (2000). The evolution of body size: What keeps organisms small? *The Quarterly Review of Biology*, 75(4):385–407.

Brambilla, M., Ferrante, E., Birattari, M., and Dorigo, M. (2013). Swarm robotics: A review from the swarm engineering perspective. *Swarm Intelligence*, 7(1):1–41.

Brown, S., Zuluaga, M., Zhang, Y., and Vaughan, R. (2005). Rational aggressive behaviour reduces interference in a mobile robot team. In *12th International Conference on Advanced Robotics (ICAR '05)*, pages 741–748. IEEE Press.

Bullock, S. (2016). "Shit happens": The spontaneous self-organisation of communal boundary latrines via stigmergy in a null model of the european badger, *Meles meles*. In Froese, T., editor, *Fifteenth International Conference on Artificial Life*, pages 518–525. MIT Press.

Daly, A. J., Baetens, J. M., and De Baets, B. (2018). Ecological diversity: Measuring the unmeasurable. *Mathematics*, 6(7):119.

Dugatkin, L. A. (2002). Cooperation in animals: An evolutionary overview. *Biology and Philosophy*, 17(4):459–476.

Goldberg, D. and Matarić, M. J. (1997). Interference as a tool for designing and evaluating multi-robot controllers. In Kuipers, B. and Webber, B., editors, *14th National Conference on AI (AAAI'97)*, page 637–642. AAAI Press.

Hooper, D. U., Chapin, F. S., Ewel, J. J., Hector, A., Inchausti, P., Lavorel, S., Lawton, J. H., Lodge, D. M., Loreau, M., Naeem, S., Schmid, B., Setälä, H., Symstad, A. J., Vandermeer, J., and Wardle, D. A. (2005). Effects of biodiversity on ecosystem functioning: A consensus of current knowledge. *Ecological Monographs*, 75(1):3–35.

Hunt, E. R., Mi, B., Geremew, R., Fernandez, C., Wong, B. M., Pruitt, J. N., and Pinter-Wollman, N. (2019). Resting networks and personality predict attack speed in social spiders. *Behavioral Ecology and Sociobiology*, 73(7):1–12.

Jacyno, M., Bullock, S., Luck, M., and Payne, T. R. (2009). Emergent service provisioning and demand estimation through self-organizing agent communities. In Sierra, C., Castelfranchi, C., Decker, K. S., and Sichman, J. S., editors, *Eighth International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2009)*, pages 481–488. ACM.

Liu, W. and Winfield, A. F. T. (2010). Modeling and optimization of adaptive foraging in swarm robotic systems. *The International Journal of Robotics Research*, 29(14):1743–1760.

Mason, N. W., Mouillot, D., Lee, W. G., and Wilson, J. B. (2005). Functional richness, functional evenness and functional divergence: The primary components of functional diversity. *Oikos*, 111(1):112–118.

Mayya, S., Pierpaoli, P., and Egerstedt, M. (2019). Voluntary retreat for decentralized interference reduction in robot swarms. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 9667–9673. IEEE Press.

Page, S. (2008). *The Difference*. Princeton University Press.

Petchey, O. L. and Gaston, K. J. (2006). Functional diversity: Back to basics and looking forward. *Ecology Letters*, 9:741–758.

Pini, G., Brutschy, A., Birattari, M., and Dorigo, M. (2009). Interference reduction through task partitioning in a robotic swarm. In Filipe, J., Andrade-Cetto, J., and Ferrier, J.-L., editors, *6th International Conference on Informatics in Control, Automation and Robotics*, pages 52–59. INSTICC Press.

Pitonakova, L., Crowder, R., and Bullock, S. (2018). The information-cost-reward framework for understanding robot swarm foraging. *Swarm Intelligence*, 12(1):71–96.

Schneider-Fontán, M. and Matarić, M. J. (1998). Territorial multi-robot task division. *IEEE Transactions on Robotics and Automation*, 14(5):815–822.

Stubbs, W. J. and Wilson, J. B. (2004). Evidence for limiting similarity in a sand dune community. *Journal of Ecology*, 92(4):557–567.

Tilman, D., Reich, P. B., and Knops, J. M. (2006). Biodiversity and ecosystem stability in a decade-long grassland experiment. *Nature*, 441(7093):629–632.

Trautman, P. and Krause, A. (2010). Unfreezing the robot: Navigation in dense, interacting crowds. In *Int. Conf. on Intelligent Robots and Systems, IROS 2010*, pages 797–803. IEEE Press.

Twu, P., Mostofi, Y., and Egerstedt, M. (2014). A measure of heterogeneity in multi-agent systems. In *Proc. 2014 American Control Conference*, pages 3972–3977. IEEE Press.

Vahl, W. K., Van Der Meer, J., Weissing, F. J., Van Dullemen, D., and Piersma, T. (2005). The mechanisms of interference competition: Two experiments on foraging waders. *Behavioral Ecology*, 16(5):845–855.

Vaughan, R. T., Støy, K., Sukhatme, G. S., and Matarić, M. J. (2000). Go ahead, make my day: Robot conflict resolution by aggressive competition. In Meyer, J.-A., Berthoz, A., Floreano, D., Roitblat, H., and Wilson, S. W., editors, *Intl. Conf. on Simulation of Adaptive Behavior*, pages 491–500. MIT Press.

The Last One Standing? - Recent Findings on the Feasibility of Indirect Reciprocity under Private Assessment

Marcus Krellner[1,2] and The Anh Han[1]
[1]Teesside University, UK, [2]TU Dresden, Germany
E-Mail: Krellner.Marcus@gmx.de, T.Han@tees.ac.uk

Abstract

Indirect reciprocity (IR) is an important mechanism for promoting cooperation among self-interested agents. Simplified, it means: "you help me, therefore somebody else will help you" (in contrast to direct reciprocity: "you help me; therefore I will help you"). IR can be achieved via reputation and norms. However, it was often argued that IR only works if reputations are public and does not do so under private assessment (PriA). Yet, recent papers suggest that IR under PriA is feasible, and that it has more variety and ways to improve, than have been considered before.

Indirect reciprocity is usually modeled using self-interested agents playing the donation game (Sigmund, 2016): a random agent (donor) is selected to pay a personal cost c to grant benefit b ($b > c$) to another random agent (recipient). Hence, for the population as a whole, it is best if every agent decides to pay the cost (i.e. cooperate). However, the individually preferred choice for each agent is to avoid the cost (i.e. defect), making the game a social dilemma. To maintain cooperation and prevent defection, agents may use strategies that are based on reputations and norms (Nowak and Sigmund, 1998a; Ohtsuki and Iwasa, 2006).

The reputation of the recipient determines how the potential donor should act (pay the cost or not). Norms determine what reputations the acting agent will earn. A simple version of such a strategy may state: If the reputation of the recipient is good, then donate, otherwise defect; and: if an agent donates, he earns a good reputation, otherwise a bad one. This strategy was named "scoring" (Nowak and Sigmund, 1998b). It was shown that such simple strategies, whose norms only consider actions, cannot maintain stable cooperation. With these norms, agents will earn a bad reputation if they do not donate to those with a bad reputation. This is known as the problem of justified punishment (Panchanathan and Boyd, 2003).

This problem can be solved using norms that also consider the current reputation of the recipient (so-called 2nd-order norms) or even the reputation of the donor itself (3rd-order norms). For any strategy to work, defection against bad individuals should not worsen a player's reputation. This holds for the so-called "leading-eight" strategies (Ohtsuki and Iwasa, 2006), which were discovered by a pioneering exhaustive analytical search (Ohtsuki and Iwasa, 2004). They include L1 "standing" (Leimar and Hammerstein, 2001) and L7 "staying" (Sasaki et al., 2017).

Yet, most previous studies on the subject assumed a simplified condition: public assessment (for an excellent review see Okada (2020a)). It means, that every agent has a single reputation value (good or bad), which is agreed upon by all. Imagine a scenario where a single agent observes the interaction and then shares its perception with all others. It may commit an error, e.g. perceiving a cooperation as a defection, causing an undeserved bad reputation. But, all still agree on a single reputation value. If however, at least two agents observe the interaction and may commit perception errors independently, they may disagree on the donor's reputation afterwards. Thus, instead of a single public reputation, there are private *opinions*. This causes severe problems to the reputation based system since the current opinions influence the assessment of future interactions (problem of PriA). Two agents with different opinions may assess the same situation differently, even if no further errors occur (Figure 1).

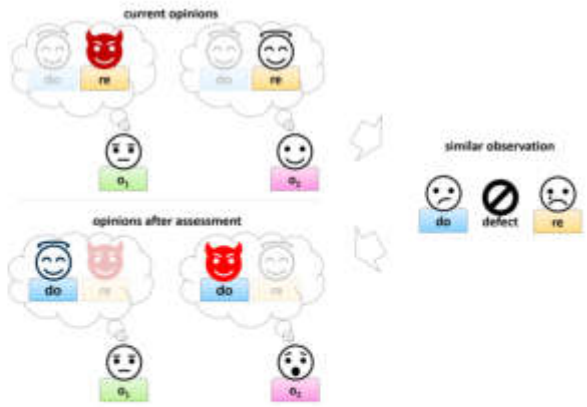


Figure 1: Disagreement about the recipient (**re**) between observer 1 ($\mathbf{o}_1$) and observer 2 ($\mathbf{o}_2$) causes new disagreement about the donor (**do**).

This way, bad opinions can wrongly cascade through the population, even to the point where no one will cooperate any longer. Strategies which assess cooperation against bad individuals as bad seem especially struck (Okada et al., 2017; Brandt and Sigmund, 2004), but this is a problem for all leading-eight strategies (Hilbe et al., 2018).

We will discuss recent developments in three directions. First, two new alterations of indirect reciprocity strategies have been introduced. The first of which was "pleasing" (Krellner and Han, 2020, 2021). A pleasing agent acts in accordance with others' expectations of its behavior (i.e. it pleases them), instead of being guided by its own assessment. As such, a pleasing agent can achieve a better reputation than previously considered strategies when there is disagreement in the population. It was shown that most leading-eight strategies can overcome the problem of PriA by applying pleasing. Especially L1, L6 & L7 have excellent results. Pleasing is effective even if the opinions of only a few other individuals are considered and when it bears additional costs. Pleasing is a selfish strategy, but since it increases the player's pay-off and also leads to more cooperation in the population, it causes no further dilemmas.

Another newly introduced alteration were zero-determinant strategies of IR (Schmid et al., 2021a). Such strategies can assess actions probabilistically and can therefore show some lenience. For example, instead of assessing every defection as bad, they assess 10% of them as good. This could help dealing with unintentional errors, and indeed the principle works in the strategy of "generous scoring". It is based on "scoring" (Nowak and Sigmund, 1998b), a strategy simpler than the leading-eight but that was shown to be unstable in public assessment (Leimar and Hammerstein, 2001). The zero-determinant approach revives this idea. "Generous scoring" can prevail in noisy IR under PriA, in striking analogy how "generous tit-for-tat" can prevail in noisy direct reciprocity. Sadly, this kind of probabilistic generousity seems not to help leading-eight strategies under PriA, neither applied to assessment nor action (Schmid et al., 2021b).

The second direction of recent developments were reevaluations whether the leading-eight are not successful under PriA. Depending on the evolutionary parameters of mutation rate and selection strength, there can be substantial amount of cooperation (Krellner and Han, 2021), because leading-eight tend to coexist with unconditional cooperators (All-C) (Okada et al., 2017). The more All-C are in the population, the easier unconditional defectors (All-D) can spread. However, the more All-D there are, the more some leading-eight can outperform All-C (and spread since they outperform All-D anyway). This feedback causes a stable state of cooperation. Although this is not true for all leading-eight, considerable cooperation occurs for L1 "standing", L2 & L7 "staying", and somewhat for L3 & L4.

In the last direction, two investigations were able to exhaustively search for successful strategies, as was done for strategies under public assessment almost two decades ago (Ohtsuki and Iwasa, 2004). Conveniently, they differ in their approach. The first (Okada, 2020b) generalized previous work on replicator dynamics in populations with three strategies: the IR strategy of interest together with All-C and All-D. They found that only two groups of strategies were successful. First, a group of four, the leading-eight strategies: L1, L7, L3 & L4. Second, a group of eight, novel strategies, which all ignore defections against cooperators. This was a surprising result, and the authors dubbed it a second way to overcome the problem of justified punishment (by not letting it be an issue in the first place). However, L1 & L7 seem strictly better than any of these new strategies.

The second paper conducted an exhaustive search for evolutionarily stable strategies (ESS) (Perret et al., 2021) This approach pitches all strategies together, considering the states of single mutants in populations of infinite residents. There was no cooperative ESS in the presence of errors. This may not surprise, because being ESS, i.e. having at least the same fitness as 257 possible invading strategies, is much harder than competing against just All-C and All-D. Especially, since the state of stable coexistence with All-C cannot be considered. If errors were excluded however, three groups of cooperative strategies were found to be ESS. The most successful group contained only L1, L2, L3 & L4. The other strategies were novel, with no overlap to the strategies found for replicator dynamics (Okada, 2020b).

The implications of both exhaustive searches are limited. They rely on deterministic evolutionary dynamics and the assumption of solitary observations (Okada et al., 2018). Yet, it is striking how some leading-eight emerge again in both of them. And, there is also overlap with results of probabilistic dynamics (Hilbe et al., 2018; Krellner and Han, 2021). Some strategies reappear in several studies: L2, L3, L4 & L7. Only L1 "standing" did so in all four (although, since no strategy is ESS in the presence of errors, L7 "staying" not being ESS should be taken with a grain of salt). L1 was actually the first strategy ever proposed to solve the problem of justified punishment (Leimar and Hammerstein, 2001). And, it should be considered a 2^{nd}-order strategy (and so should L4, L7 & L8), because it does not use 3^{rd}-order information (the donor's reputation). Instead, it simply ignores defections against bad recipients by all donors (the same as "keep" in Okada (2020b)). This does not require an opinion about the donor, it does not even require knowing the donor's *identity*. So, is L1 the last one ... standing?

Given the current findings, L1 but also L2, L3, L4 and especially L7 should be considered as potentially capable of indirect reciprocity under private assessment. And there might be more strategies, which were formerly not discovered under public assessment. In addition, new approaches such as *pleasing* and *generous scoring* give even more credentials to the validity of IR under PriA.

References

Brandt, H. and Sigmund, K. (2004). The logic of reprobation: Assessment and action rules for indirect reciprocation. *Journal of Theoretical Biology*, 231(4):475–486.

Hilbe, C., Schmid, L., Tkadlec, J., Chatterjee, K., and Nowak, M. A. (2018). Indirect reciprocity with private, noisy, and incomplete information. *Proceedings of the National Academy of Sciences*, page 201810565.

Krellner, M. and Han, T. A. (2020). Putting oneself in everybody's shoes - Pleasing enables indirect reciprocity under private assessments. In *The 2020 Conference on Artificial Life*, pages 402–410, Cambridge, MA. MIT Press.

Krellner, M. and Han, T. A. (2021). Pleasing Enhances Indirect Reciprocity-Based Cooperation Under Private Assessment. *Artificial Life*, pages 1–31.

Leimar, O. and Hammerstein, P. (2001). Evolution of cooperation through indirect reciprocity. *Proceedings of the Royal Society of London. Series B: Biological Sciences*, 268(1468):745–753.

Nowak, M. A. and Sigmund, K. (1998a). Evolution of indirect reciprocity by image scoring. *Nature 1998 393:6685*, 393(6685):573–577.

Nowak, M. A. and Sigmund, K. (1998b). The Dynamics of Indirect Reciprocity. *Journal of Theoretical Biology*, 194:561–574.

Ohtsuki, H. and Iwasa, Y. (2004). How should we define goodness? - Reputation dynamics in indirect reciprocity. *Journal of Theoretical Biology*, 231(1):107–120.

Ohtsuki, H. and Iwasa, Y. (2006). The leading eight: Social norms that can maintain cooperation by indirect reciprocity. *Journal of Theoretical Biology*, 239(4):435–444.

Okada, I. (2020a). A Review of Theoretical Studies on Indirect Reciprocity. *Games*, 11(3):27.

Okada, I. (2020b). Two ways to overcome the three social dilemmas of indirect reciprocity. *Scientific Reports*, 10(1).

Okada, I., Sasaki, T., and Nakai, Y. (2017). Tolerant indirect reciprocity can boost social welfare through solidarity with unconditional cooperators in private monitoring. *Scientific Reports*, 7(1):9737.

Okada, I., Sasaki, T., and Nakai, Y. (2018). A solution for private assessment in indirect reciprocity using solitary observation. *Journal of Theoretical Biology*, 455:7–15.

Panchanathan, K. and Boyd, R. (2003). A tale of two defectors: The importance of standing for evolution of indirect reciprocity. *Journal of Theoretical Biology*, 224(1):115–126.

Perret, C., Krellner, M., and Han, T. A. (2021). The evolution of moral rules in a model of indirect reciprocity with private assessment. *Scientific Reports*, 11(1):1–10.

Sasaki, T., Okada, I., and Nakai, Y. (2017). The evolution of conditional moral assessment in indirect reciprocity. *Scientific Reports*, 7(1):41870.

Schmid, L., Chatterjee, K., Hilbe, C., and Nowak, M. A. (2021a). A unified framework of direct and indirect reciprocity. *Nature Human Behaviour*.

Schmid, L., Shati, P., Hilbe, C., and Chatterjee, K. (2021b). The evolution of indirect reciprocity under action and assessment generosity. *Scientific Reports*, pages 1–14.

Sigmund, K. (2016). *The calculus of selfflessness*. Princeton University Press.

Shake on It: The role of Commitments and the Evolution of Coordination in Networks of Technology Firms

Ndidi Bianca Ogbo[1], Theodor Cimpeanu[1], Alessandro Di Stefano[1,2], The Anh Han[1,2,⋆]

[1] School of Computing, Engineering and Digital Technologies, Teesside University
[2] Center for Digital Innovation, Teesside University
[⋆] Corresponding author: The Anh Han (han@tees.ac.uk)

Abstract

Before embarking on a new collective venture, it is important to understand partners' preferences and intentions and how strongly they commit to a common goal. Arranging prior commitments of future actions has been shown to be an evolutionary viable strategy in the context of social dilemmas. Previous works have focused on simple well-mixed population settings, for ease of analysis. Here, starting from a baseline model of a coordination game with asymmetric benefits for technology adoption in the well-mixed setting, we examine the impact of different population structures, including square lattice and scale-free (SF) networks, capturing typical homogeneous and heterogeneous network structures, on the dynamics of decision-making in the context of coordinating technology adoption. We show that, similarly to previous well-mixed analyses, prior commitments enhance coordination and the overall population payoff in structured populations, especially when the cost of commitment is justified against the benefit of coordination, and when the technology market is highly competitive. When commitments are absent, slightly higher levels of coordination and population welfare are obtained in SF than lattice. In the presence of commitments and when the market is very competitive, the overall population welfare is similar in both lattice and heterogeneous networks; though it is slightly lower in SF when the market competition is low, while social welfare suffers in a monopolistic setting. Overall, we observe that commitments can improve coordination and population welfare in structured populations, but in its presence, the outcome of evolutionary dynamics is, interestingly, not sensitive to changes in the network structure.

Introduction

In a variety of societies and contexts, achieving a collective effort among individuals with their own personal interests is a significant social and economic problem. Bringing agents with different individual interests together to achieve a common goal is a social challenge that is often encountered in socio-economic, biological as well as artificial life system settings (Ostrom, 1990; Pitt et al., 2012; Sigmund, 2010; Han et al., 2012; Sayama, 2011; Bedau, 2003). Social dilemmas are among the problems encountered in biological and social sciences, and these are typically used as general metaphors for studying the evolution of cooperation or coordination by describing the conflicts between the self-interests of individuals and the common goals or interests of the community. Researchers from different fields of study, such as Economics, Evolutionary Biology, Mathematics and Computer Science have made efforts towards developing mechanisms to tackle social dilemma problems (Nowak, 2006b; Sigmund, 2010; West et al., 2007; Han, 2013; Andras et al., 2018; Pereira et al., 2021; Han et al., 2020).

In the study of evolution of collective behaviours such as cooperation, coordination and fairness, several mechanisms that can enhance collective behaviours among agents have been explored. They include kin selection, direct reciprocity, indirect reciprocity, group selection, reward and punishment, and spatial networks (Nowak, 2006a; Sigmund, 2010; Uchida et al., 2019; Perc et al., 2017; Perret et al., 2021).

Another mechanism that has been explored recently in tackling these kinds of social dilemma problems is pre-commitments (or prior commitments), whereby players can choose to make a prior agreement before commencing an interaction (Nesse, 2001b; Frank, 1988; Han et al., 2017; Han, 2022; Akdeniz and van Veelen, 2021). *Commitment* has been proposed as an evolutionarily viable strategy inducing cooperative behaviour in the context of cooperation dilemmas; namely, the Prisoner's Dilemma (PD) (Han et al., 2013) and the Public Goods Game (PGG) (Han et al., 2017). Commitment serves as a different means of incentivizing against inappropriate behaviours, while simultaneously stimulating cooperative acts in the population (Martinez-Vaquero et al., 2015; Sasaki et al., 2015; Powers et al., 2012). Previous works have mostly focused on utilizing commitment for the purpose of promoting mutual cooperation among self-interested individuals. Comparatively, few studies have investigated ways in which commitments could be applied to promote other desirable collective behaviours, such as coordination, fairness and the safe adoption of technologies (Ogbo et al., 2022; Nesse, 2001b; Ostrom, 1990; Barrett, 2016; Akdeniz and van Veelen, 2021; Han et al., 2022).

Considering today's technology-driven economy, man-

agers often face challenging decisions regarding the adoption of innovative technology. Often, companies might choose to invest in new technologies in order to gain a competitive advantage over their peers (Parsons, 1983; Clemons, 1991; Zhu and Weyant, 2003). These selfish executive priorities may lead to a coordination problem. Ensuring higher social welfare by providing several technological solutions is ignored if newer technology promises larger profits (Zhu and Weyant, 2003; Sachs, 2000). In this setting, commitments have already been shown to promote coordination in the face of technology adoption in homogeneous (namely, well-mixed) populations (Ogbo et al., 2022). However, real-world firms and their interactions are far from homogeneous. Some developers are more influential than others, or can play a more impactful role in the adoption of innovative technologies. These economies are shaped by complex networks of exchange, influence and competition where diversity is ubiquitous. Particular networks of contacts have been shown to promote the evolution of positive behaviours in settings such as cooperation (Ohtsuki et al., 2006; Santos et al., 2008; Perc et al., 2017; Chen et al., 2015; Perc and Szolnoki, 2010), fairness (Page et al., 2000; Szolnoki et al., 2012; Wu et al., 2013; Santos et al., 2017; Cimpeanu et al., 2021) and trust (Kumar et al., 2020). In the present work, we pay heed to the divide between the previous research and the diversity observed in real-world interactions, and ask whether network topology plays an important role in the coordination of beneficial *technology adoption*.

Technology innovation and collaboration networks (e.g. among firms and stakeholders) are highly heterogeneous (Schilling and Phelps, 2007; Newman, 2004). Firms interact more frequently within their spheres than outside them, forming alliances and networks of followers and collaborators (Barabasi, 2014; Ahuja, 2000). Developers may compete in numerous markets, while others might choose only to invest in a few, and their positions in inter-organisational networks strongly influence not only their behaviours (such as information and resource sharing), but also innovation outcomes (Ahuja, 2000; Shipilov and Gawer, 2020). Therefore, it is paramount to comprehend the roles that spatiality and diversity in the network of contacts play in the formation and stability of commitment-based strategies and the resulting coordination outcome. Therefore, here we depart from well-mixed settings (Ogbo et al., 2022) and examine how network structures shape the dynamics of adoption decisions and commitments.

Related Work

A large body of literature has explored the mechanisms driving the emergence and stability of collective, pro-social behaviours (Nowak, 2006a; Santos et al., 2006; Perc et al., 2017; West et al., 2007). Closely related to our work is the study of cooperative behaviours and pre-commitment in cooperation dilemmas, for both two-player and multiplayer games (Han et al., 2013, 2017). Works on collective behaviour have proven, through theoretical analysis and behavioural experiments, that to enhance cooperation among agents, arranging commitments requires enforcement. Furthermore, it has been shown that the benefits achieved from engaging in a commitment deal justify the cost of arranging a commitment (Ostrom, 1990; Cherry and McEvoy, 2013). Pre-commitment has also been investigated experimentally in Public Good Games (Chen and Komorita, 1994), showing that arranging an agreement between members of the group before the interaction enhances cooperation because their intentions are clarified. These results indicate that a pledge enhances cooperation among members of a group, although the degree of commitment required in the pledge deferentially affected the cooperation rate (Chen and Komorita, 1994). These results are in accordance to previous works exploring how coordination and cooperation can be improved by using prior commitment deals in the case of two-player and multi-player games in well-mixed populations, most especially when there exists a particular market demand (Bianca and Han, 2019; Ogbo et al., 2022). However, there exist no prior works studying pre-commitment decision-making in the context of spatial settings (i.e. beyond well-mixed, complete graph populations).

Highly relevant to our work, Santos et al. (2006) investigated the effect of population structures on different types of two player social dilemmas including the Prisoner's Dilemma, Stag Hunt and Snowdrift games. Cooperation is shown to be more prevalent in heterogeneous networks than homogeneous ones (such as square lattice and well-mixed populations). Similarly, Cimpeanu et al. (2022) have shown that safety adoption in the development of AI technology can be enhanced by heterogeneous network structures. This is also in line with what is explored in (Di Stefano et al., 2015, 2020) where it has been shown how highly heterogeneous networks, such as SF, constitute the most suitable network topology for the emergence and sustainability of cooperation in a multilayer network. This is even more marked in the presence of homophily (i.e., the tendency to associate and interact more frequently with similar people), increasing the speed and size of the formation of cooperative groups, and allowing the network to quickly converge to cooperation. Our coordination setting is different from the aforementioned models, as they focus on symmetric games, whereas we address the context in which players need to coordinate to choose different actions (i.e. an asymmetric game) (Ogbo et al., 2022). Our analysis below shows that, surprisingly, the outcomes of our model differ from (Santos et al., 2006; Cimpeanu et al., 2022) insomuch that heterogeneous networks have little positive impact on, or are even detrimental to coordination and social welfare, compared to homogeneous settings.

Moreover, this observation is partially coherent with other previous works (Gracia-Lázaro et al., 2012b,a) where it has

been shown how the population structure has little relevance as a cooperation promoter, and where the presence of a structured population does not promote or inhibit cooperation with respect to well mixed populations. Our results are also in line with (Pena et al., 2009), where authors, by introducing conformity, namely the tendency of humans to imitate locally common behaviours, have shown that when fair quantities of conformity are added in the imitation rules of the players, SF networks are no longer powerful amplifiers of cooperation. Such weakening of the cooperation-promoting abilities of SF networks is the result of a less biased flow of information in SF topologies, making hubs more susceptible to being influenced by less-connected neighbours.

Model and Methods

We first recall the models describing technology adoption decision-making as a coordination game (with asymmetric benefits), in the absence or presence of pre-commitments, as described by Ogbo et al. (2022). We introduce a two-player technology adoption game and extend it to accommodate the formation of pre-commitment arrangements before the game. We then describe our models and methods for studying the impact of population structures.

Technology Adoption (TD) game

The technology adoption (TD) game consists of two firms competing in a common market, making a decision on investing in a technology (Ogbo et al., 2022). These firms either invest in innovative technology which promises the highest profits (i.e. high technology, denoted by **H**), or to invest in less sophisticated products, resulting in a lower potential benefit (i.e. low technology, **L**). The interaction between these firms is described in terms of costs and benefits of the investments by the following payoff matrix (for row player):

$$\begin{array}{c} \\ H \\ L \end{array} \begin{array}{c} \begin{array}{cc} H & \quad L \end{array} \\ \begin{pmatrix} \alpha b_H - c_H & b_H - c_H \\ b_L - c_L & \alpha b_L - c_L \end{pmatrix} \end{array} = \begin{array}{c} \\ H \\ L \end{array} \begin{array}{c} \begin{array}{cc} H & \quad L \end{array} \\ \begin{pmatrix} a & b \\ c & d \end{pmatrix} \end{array}, \quad (1)$$

where c_L, c_H and b_L, b_H ($b_L \leq b_H$) represent the costs and benefits of choosing L and H, respectively. The parameter $\alpha \in (0, 1)$ describes the fraction of the total market benefit when two firms choose to invest in the same technology. It indicates the (reversed) competitive level of the product market. That is, the smaller α is (i.e. the higher the market competitiveness), the more important that the firms coordinate to choose different technologies. The entries of the payoff matrix are simplified and denoted by a, b, c, d, as above, for easy of referencing. Without loss of generality, we assume that H would generate a greater net benefit than L, i.e. $c = b_L - c_L < b_H - c_H = b$ (Ogbo et al., 2022).

It is noteworthy that the payoff matrix for this technology adoption model can be generally applied to several other coordination problems, see details in (Ogbo et al., 2022).

Technology adoption game in presence of pre-commitments

In the technology adoption model with prior commitments (Ogbo et al., 2022), players are able to arrange a commitment deal before an interaction. In this game, a player who is a commitment proposer asks the other player to adopt a different technology. This means a commitment proposer (HP or LP) intending to invest in high technology (H), asks the co-player to invest in low technology (L). We have described our model such that H will always lead to a high benefit technology and L to a low benefit technology. Similarly to previous models of commitments (for PD and PGG) (Han et al., 2013, 2015; Ogbo et al., 2022), to make the commitment deal reliable, a proposer pays an arrangement cost ϵ. If the co-player is satisfied and agrees to the proposal, then the proposer assumes that their competitor will adopt the agreed choice. Yet, there exists no guarantee that this will actually be the case. Thus, whenever a co-player refuses to commit, HP and LP would both play H in the game. Failure to comply with an agreed deal will result in a penalty, i.e. the player dishonouring the agreement has to pay a compensation cost δ to the co-player. Differently from previous models on PD and PGG, where an agreed outcome leads to the same payoff for all parties in the agreement (mutual cooperation benefit) (Han et al., 2013), in the current model such an outcome would lead to different payoffs for those involved. Therefore, as part of the agreement, HP would compensate after the game an amount θ_1 to the player which honours the agreement; while LP would request a compensation θ_2 from their co-player. Besides HP and LP, we consider a minimal model with the following (basic) strategies in this commitment version:

- *Non-proposing acceptors*, HC and LC, who always commit when being proposed a commitment deal, wherein they are willing to adopt any technology proposed (even when it is different from their intended choice), honour the adopted agreement, but do not propose a commitment themselves. They play their intended choice, i.e. H and L, respectively, when there is no agreement in place;

- *Non-acceptors*, HN and LN, who do not accept commitment, play their intended choice during the game, and do not propose commitments;

- *Fake committers*, HF and LF, who accept a commitment proposal, yet play the choice opposite to what has been agreed whenever the game takes place. These players assume that they can exploit the commitment proposing players without suffering any consequences;

Altogether, the model consists of eight strategies that define the following payoff matrix, capturing the average payoffs that each strategy will receive upon interacting with one of the other strategies (where we denote $\lambda = \theta_1 + \theta_2$, $\lambda_1 = b - \epsilon - \theta_1$, $\lambda_2 = c - \epsilon + \theta_2$, $\lambda_3 = a - \epsilon + \delta$ and $\lambda_4 = d - \epsilon + \delta$, just for the sake of clear representation)

$$\begin{array}{c} \\ \text{HP} \\ \text{LP} \\ \text{HN} \\ \text{LN} \\ \text{HC} \\ \text{LC} \\ \text{HF} \\ \text{LF} \end{array} \begin{array}{c} \begin{array}{cccccccc} \text{HP} & \text{LP} & \text{HN} & \text{LN} & \text{HC} & \text{LC} & \text{HF} & \text{LF} \end{array} \\ \begin{pmatrix} \frac{b+c-\epsilon}{2} & \frac{2b-\epsilon-\lambda}{2} & a & b & \lambda_1 & \lambda_1 & \lambda_3 & \lambda_3 \\ \frac{2c-\epsilon+\lambda}{2} & \frac{b+c-\epsilon}{2} & a & b & \lambda_2 & \lambda_2 & \lambda_4 & \lambda_4 \\ a & a & a & b & a & b & a & b \\ c & c & c & d & c & d & c & d \\ c+\theta_1 & b-\theta_2 & a & b & a & b & a & b \\ c+\theta_1 & b-\theta_2 & c & d & c & d & c & d \\ a-\delta & d-\delta & a & b & a & b & a & b \\ a-\delta & d-\delta & c & d & c & d & c & d \end{pmatrix} \end{array}. \tag{2}$$

Note that when two commitment proposers interact only one of them will need to pay the cost of setting up the commitment. Yet, as either one of them can take this action, they pay this cost only half of the time (on average). In addition, the average payoff of HP when interacting with LP is given by $\frac{1}{2}(b-\epsilon-\theta_1+b-\theta_2) = \frac{1}{2}(2b-\epsilon-\theta_1-\theta_2)$. When two HP players interact, each receives $\frac{1}{2}(b-\epsilon-\theta_1+c+\theta_1) = \frac{1}{2}(b+c-\epsilon)$.

Network Topology

Real-world networks are not static and inherently heterogeneous (Barabási and Albert, 1999; Dorogovtsev, 2010; Newman, 2003). Networks evolve with new nodes entering and creating connections to already existing nodes (Dall'Asta et al., 2006). Several works have unveiled how structural *heterogeneity* plays a key role in both the evolution of cooperation (Poncela et al., 2007; Dercole et al., 2019; Cimpeanu et al., 2019; Di Stefano et al., 2020, 2015) or emergence of fairness (Sinatra et al., 2009; Cimpeanu et al., 2021). To study the effects of heterogeneity on the evolution of coordination and prior commitments, we will study spatially structured populations, as well as a growing network model characterized by *preferential attachment*.

Links in the networks describe both proximity for the purposes of interactions (i.e. whom the agents can interact with), but also for the purposes of social learning (whom the agents can imitate). Thus, the network of interactions coincides with the network of imitation (Ohtsuki et al., 2007). Structured populations converge more readily than heterogeneous populations, and our choice of population sizes and maximum number of generations reflects on these differences.

We model spatially structured populations using a square lattice graph (SL) of size $Z = L \times L$ with periodic boundary conditions— a widely adopted population structure in population dynamics and evolutionary games (for a survey, see (Szabó and Fáth, 2007)). This network adds spatial structure, but each agent can only interact with its four immediate edge neighbours; thus, we can say it is homogeneous in the number of neighbours, unlike real-world networks of firms (Schilling and Phelps, 2007; Newman, 2004). For all of our experiments with this network type, we choose $L = 30$ (i.e. $Z = 900$).

In an effort to introduce a more realistic interaction setting, we turn to SF networks, specifically the popular Barabási and Albert (BA) model (Barabási and Albert, 1999). SF networks constructed using the BA model follow a preferential attachment rule, leading to a typical *power-law degree distribution*. To construct such a network, we start from a small complete graph of size m_0 and gradually introduce new nodes. Each new node selects m other nodes according to a probability proportional to their degree (how connected they are in the network), creating a new link with each of these nodes. This procedure repeats until a network of size Z is obtained. The resultant network follows a power-law degree distribution, which is skewed with a long tail. There are few hubs in the network, which gradually increase in size (number of connections) as the size of the network increases, in a classic *rich-get-richer* setting. The obtained network has average degree $h = 2m$, small clustering (of order $1/N$) and a power-law degree distribution $P(k) \sim k^{-\gamma}$, with $\gamma = 3$. For all of our experiments with this network type, we seed 10 different networks of size $Z = 1000$, with an average connectivity of $h = 4$. The latter is chosen for ease of comparison against the square lattice graph.

Population Dynamics

We consider a population of agents distributed on a network (see below for different topologies), randomly assigning one of the eight strategies. At each time step or generation, each agent plays the game with its immediate neighbours. The success of each agent (i.e., its fitness) is the sum of the payoffs in all these encounters. Each individual fitness, as detailed below, defines the time-evolution of strategies, as successful choices will tend to be imitated by their peers.

At the end of each generation, a randomly selected agent A with a fitness f_A chooses to copy the strategy of a randomly selected neighbour, agent B, with fitness f_B with a probability p that increases with their fitness difference. Here we adopt the well-studied Fermi update or pairwise comparison rule, where (Traulsen et al., 2006):

$$p = (1 + e^{\beta(f_A - f_B)})^{-1}. \tag{3}$$

In this case, β conveniently describes the selection intensity — i.e., the importance of individual success in the imitations process: $\beta = 0$ represents neutral drift while $\beta \to \infty$ represents increasingly deterministic imitation (Traulsen et al., 2006). Varying β allows capturing a wide range of update rules and levels of stochasticity, including those used by humans, as measured in lab experiments (Zisis et al., 2015; Rand et al., 2013; Grujić and Lenaerts, 2020). In line with

previous works and lab experiments, we set $\beta = 1$ in our simulations, ensuring a high intensity of selection (Pinheiro et al., 2012). This update rule implicitly assumes an asynchronous update rule, where at most one imitation occurs at each time-step. With a given probability μ, this process is replaced instead by a randomly occurring mutation. A mutation is equivalent to behavioural exploration, whereby the individual makes a stochastic decision switch to one of the eight available strategies.

We simulate this evolutionary process until a stationary state or a cyclic pattern is reached. As network topology and the number of available strategies affects rates of convergence, we select different maximum numbers of runs for each network type, for robustness. The simulations have been conducted for 15000 generations in the case of lattice and the results are averaged over the final 1000 steps. For BA networks, we run the simulations for 250000 generations (for 8 strategies, with commitment) and 50000 generations (for 2 strategies, no commitment), averaging over the final 25000 steps. Furthermore, to improve accuracy, for each set of parameter values and pre-seeded network (in the case of BA networks), the final results are obtained from averaging 15 independent realizations (replicates). When shown in figures, the error bars represent the standard deviation of the mean between replicates.

Results

In absence of pre-commitments

To begin with, we study evolutionary outcomes in the technology adoption (TD) game when arranging commitments is not available as an option. Figure 1 shows that for all values of α, i.e. regardless of market competition, players dominantly choose to adopt high technology (i.e. playing H in the TD game). This observation is applied to both lattice and SF networks, with L having a slightly higher frequency in the SF network than lattice when α is sufficiently small (i.e., highly competitive markets). To support the understanding of this observation, we show in Figure 2 the evolution of the two strategies H and L over time, for different values of α. We observe that only in SF and when the market competitiveness is high ($\alpha = 0.1$), L players have a chance to survive the dominance of H players, even though they are still in minority. This observation differs from previous findings in homogeneous networks, but is in line with previous works studying cooperation dilemmas in structured populations, where SF networks can lead to higher levels of prosocial behaviours than in lattice counterparts (Santos et al., 2008; Perc et al., 2017; Santos et al., 2006).

In the following, unless stated otherwise, we set $\alpha = 0.5$. As such, there is a clear need for arranging prior commitments among players to enhance coordination. Otherwise, this would lead to the extinction of firms that would choose to adopt low-benefit technologies (i.e., L players) in the population.

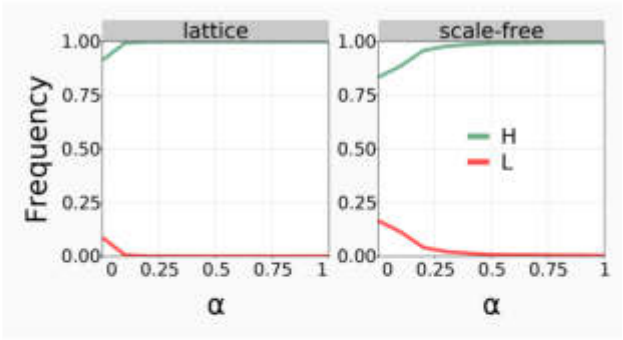


Figure 1: **In the absence of commitments: Depicted are frequencies of H and L as a function of α for lattice (left panel) and scale-free (right panel) networks.** Parameters: in all panels $c_H = 1, c_L = 1, b_L = 2, b_H = 6$ and $\beta = 1$.

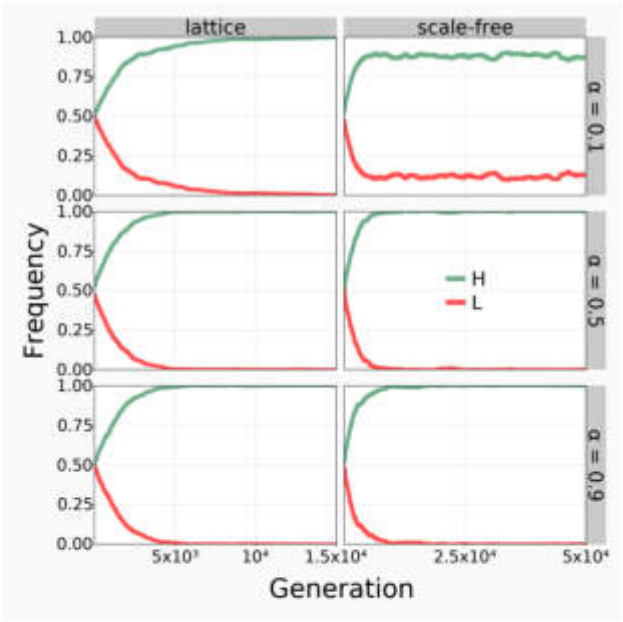


Figure 2: **Time-evolution of frequencies of two strategies H and L in the absence of commitments**, for different levels of competitiveness: high ($\alpha = 0.1$), medium ($\alpha = 0.5$) and low ($\alpha = 0.9$). Only when the market competitiveness is high ($\alpha = 0.1$) and in case of SF (first row), L players have a non-negligible (but still rather small) frequency. Other parameters' values are the same as in Figure 1.

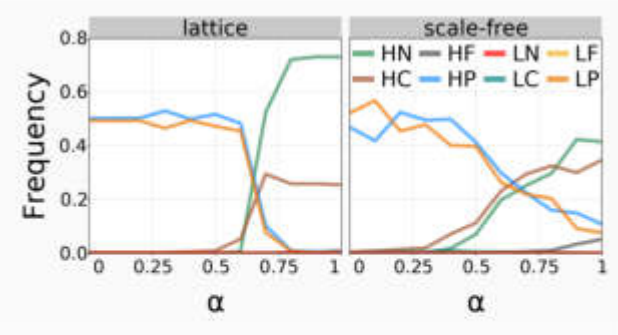


Figure 3: **Frequencies for the eight strategies (HP, LP HN, LN, HC, LC, LF, HF) as a function of α for lattice (left panel) and scale-free (right panel) networks.** In both types of network, we observe that when α is small, indicating a highly competitive market, the commitment proposing strategies (HP and LP) dominate the population. When α is large, the non-commitment strategies HN and HC dominate the population. Comparing the two types of network, commitment proposers have higher frequencies in SF than in lattice when α is sufficiently large. Parameters: in all panels $c_H = 1$, $c_L = 1$, $b_L = 2$, and $b_H = 6$, $\beta = 1$; $\mu = 0.001$; $\epsilon = 0.1$; $\delta = 6$. Fair agreements are applied, where θ_1 and θ_2 are given by $\theta_1 = (b - c - \epsilon)/2$ and $\theta_2 = (b - c + \epsilon)/2$.

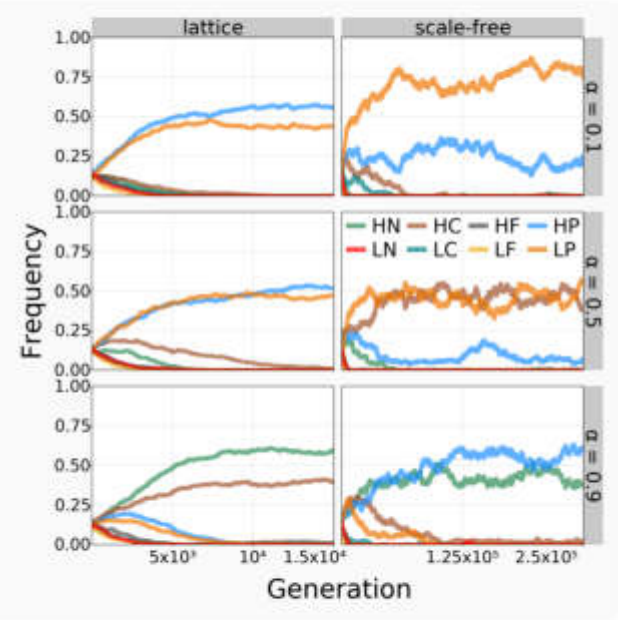


Figure 4: **Time-evolution of frequencies of eight strategies (HP, LP HN, LN, HC, LC, LF, HF) in the presence of commitments**, for different levels of competitiveness: High ($\alpha = 0.1$), medium ($\alpha = 0.5$) and low ($\alpha = 0.9$). Other parameter values are the same as in Figure 3

In presence of pre-commitments

We now examine whether arranging pre-commitments can improve coordination outcomes, and whether these outcomes change for different population structures (homogeneous versus heterogeneous networks). We analyse evolutionary outcomes in a population with eight strategies (see Model and Methods), for varying different key factors. Firstly, Figure 3 shows the frequency of these strategies as a function of α in both lattice and SF networks. In general, the commitment proposing strategies HP and LP are more frequent in the population when the market competition is high (i.e. when α is small). When there is less competition in the market, HN and HC take over the population, leading to a monopolized market. This result is similar in lattice and SF networks. However, we note a small discrepancy between them, where commitment proposers have higher frequencies in SF network than in lattice, if α is sufficiently large. We also observe that the HN strategy has a higher frequency in lattice networks than in heterogeneous networks. For instance, when $\alpha = 1$, the fraction of HN strategists is 0.78 of the total in lattice networks, and 0.4 in SF networks.

Figure 5 studies the frequency of the strategies for varying the cost of arranging commitments, i.e. ϵ. For commitment proposers to engage in a commitment arrangement, the cost of arranging such a commitment needs to be sufficiently small. This observation is in line with previous works on pre-commitments in cooperation (Han et al., 2013, 2015, 2017; Sasaki et al., 2015; Akdeniz and van Veelen, 2021) and coordination (Ogbo et al., 2022) games, in the well-mixed population settings. That is, our analysis has again confirmed that prior commitments can provide a pathway for the evolution of coordination even in the presence of more complex networks of interaction.

In order to determine when commitments can lead to significant improvement (in terms of overall population payoff or welfare), in Figure 6 we compare the average population payoff when commitment is present and when it is absent, as a function of α (which is the competitiveness of the market, left panel) and ϵ (which is the cost of arranging a commitment, right panel). We observe here that commitments can lead to significant improvements when α and ϵ are sufficiently small. That is, arranging pre-commitments is highly beneficial whenever the market competitiveness level is high, and when it is not too costly to do so. We also observe that SF and lattice networks lead to similar population payoffs for all values of ϵ and α. One small difference is that SF leads to a slightly higher payoff when commitment is absent in highly competitive markets ($\alpha \lessapprox 0.2$), while this effect is reversed when commitments can be initiated. That is, the heterogeneity of SF networks might be slightly detrimental for population welfare when commitment is present. The main reason for this effect is that when the competitiveness level is low (i.e. large α), coordination is not as important as when the competitiveness level is high (a pop-

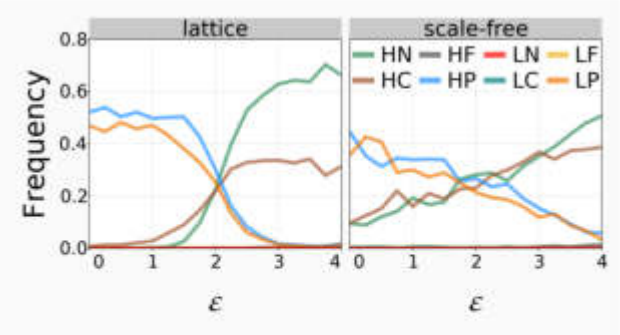


Figure 5: **Frequencies for the eight strategies (HP, LP HN, LN, HC, LC, LF, HF) as a function of the cost of commitment ϵ for lattice (left panel) and scale-free (right panel) networks.** In general, we observe that for both lattice and SF networks, when the cost of arranging a commitment (ϵ) is sufficiently small (for lattice network when $\epsilon = 2$ and for SF network when $\epsilon = 1.8$, approximately), commitment strategies HP and LP are more abundant in the population. When ϵ exceeds these points, the non-proposing strategies HN and HC who do not pay for the cost of commitment deal dominate the population. Moreover, when ϵ is sufficiently large, commitment proposing strategies are more frequent in SF than in lattice. Parameters: in all panels $c_H = 1, c_L = 1, b_L = 2$, and $b_H = 6, \beta = 1; \mu = 0.001; \alpha = 0.5; \delta = 6$. Fair agreements are applied, where θ_1 and θ_2 are given by $\theta_1 = (b - c - \epsilon)/2$ and $\theta_2 = (b - c + \epsilon)/2$.

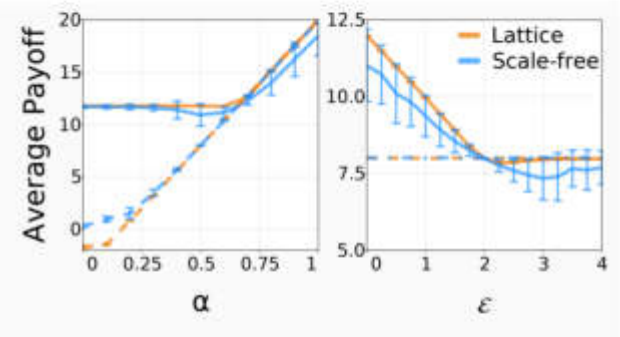


Figure 6: **Average population payoff (welfare) as a function of α (left panel) and ϵ (right panel), in lattice and scale-free networks. We compare these payoffs when commitment is present (solid lines) vs when it is absent (dashed lines).** We observe that arranging commitments leads to larger population payoffs than when it is not an option whenever α and ϵ are sufficiently small (up to around 0.7 and 2.0, respectively). That is, arranging pre-commitments is highly beneficial whenever the market competitiveness level is high, and when it is not too costly to do so. We also observe that, SF and lattice lead to similar population payoffs for all values of ϵ and α. SF leads to slightly higher payoff when commitment is absence when α is quite small (up to 0.2), while it is reversed when commitment is present. Other parameters in all: $\beta = 1; \mu = 0.001; \delta = 6, \theta_1 = (b-c-\epsilon)/2, \theta_2 = (b-c+\epsilon)/2$. Parameters in panel a; $\epsilon = 0.1$, and panel b; $\alpha = 0.5$.

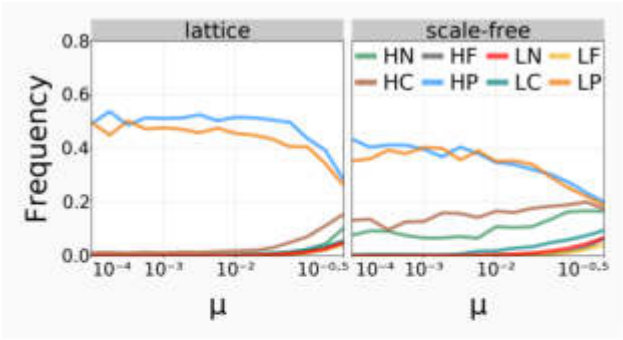


Figure 7: **Frequencies for the eight strategies (HP, LP HN, LN, HC, LC, LF, HF) as a function of the mutation rate μ (ranging from a small $10^{-4} = 0.0001$ to a high $10^{-0.5} = 0.316$ values) for lattice (left panel) and scale-free (right panel) networks.** Larger mutation leads to greater levels of randomness in agents' behaviours, leading to strategies' frequencies being closer. Also, for the whole range of μ, SF leads to higher diversity in terms of strategic behaviours than in lattice. Parameters: in all panels $c_H = 1, c_L = 1, b_L = 2$, and $b_H = 6$. Other parameters: $\beta = 1$; $\epsilon = 0.1$; $\alpha = 0.5$; $\delta = 6$. Fair agreements are applied, where θ_1 and θ_2 are given by $\theta_1 = (b - c - \epsilon)/2$ and $\theta_2 = (b - c + \epsilon)/2$.

ulation of H players has a higher welfare when α increases, see Figure 6), and it is costly to arrange commitments. The evolution of commitment proposing strategies for large α (see Figures 3 and 4) leads to higher spending for arranging commitments in the population.

We also measure the effects of varying mutation rates in our work. In particular, we focus on how increasing mutation rates affect the coordination outcome in the population, differently from previous works which assume a small mutation limit (for convenience of theoretical and numerical analyses) (Ogbo et al., 2022; Han et al., 2013). Indeed, Figure 7 shows the outcomes for varying mutation rates in both lattice and SF networks. We observe that a higher mutation leads to a more balanced state of the strategy frequencies in the population. This effect is more notable in SF networks than in lattice populations. Behavioural exploration aids in eliminating market monopolies.

Conclusions and Future Work

In this work, we have explored and quantified the effects of social diversity on the outcomes of technology adoption and coordination dynamics amongst firms, using two popular spatial structures. Specifically, we have studied the use of prior commitments to enhance coordination among agents making decisions to adopt competing technological solutions. We have discovered that prior commitments enhance coordination and social welfare in both lattice and scale-free networks, especially when the market is highly competitive or the cost of arranging commitments is sufficiently small. We have also observed that coordination is more achievable in the absence of commitments in the case of SF networks, compared to lattice networks. Correspondingly, the opposite

is observed if the market is competitive and commitments are possible. Our results are in line with previous analyses on well-mixed populations (Ogbo et al., 2022). When commitment is present, the overall population welfare is similar in SF and lattice networks when the market competition level is high; it is however slightly lower in SF than lattice when this competition level is low. Ultimately, we find that commitments can improve population welfare and coordination in structured populations, and that these results are robust across a wide range of network structures, unlike the case when commitment is absent.

Overall, our analysis has provided further confirmation and theoretical evidence showing that arranging prior commitments can provide a flexible and efficient pathway to achieve high levels of collective behaviours (beyond social dilemmas settings) (Nesse, 2001a; Frank, 1988; Akdeniz and van Veelen, 2021; Han, 2013). In the future, we aim to expand this network analysis to study the effects of commitments in the multi-player version of the technology adoption game (Ogbo et al., 2022), where multiple firms participate in a technology market competition.

Acknowledgements

T.A.H. is supported a Leverhulme Research Fellowship (RF-2020-603/9).

References

Ahuja, G. (2000). Collaboration networks, structural holes, and innovation: A longitudinal study. *Administrative science quarterly*, 45(3):425–455.

Akdeniz, A. and van Veelen, M. (2021). The evolution of morality and the role of commitment. *Evolutionary Human Sciences*, pages 1–53.

Andras, P., Esterle, L., Guckert, M., Han, T. A., Lewis, P. R., Milanovic, K., Payne, T., Perret, C., Pitt, J., Powers, S. T., et al. (2018). Trusting intelligent machines: Deepening trust within socio-technical systems. *IEEE Technology and Society Magazine*, 37(4):76–83.

Barabasi, A.-L. (2014). *Linked-how Everything is Connected to Everything Else and what it Means F*. Perseus Books Group.

Barabási, A.-L. and Albert, R. (1999). Emergence of scaling in random networks. *science*, 286(5439):509–512.

Barrett, S. (2016). Coordination vs. voluntarism and enforcement in sustaining international environmental cooperation. *Proceedings of the National Academy of Sciences*, 113(51):14515–14522.

Bedau, M. A. (2003). Artificial life: organization, adaptation and complexity from the bottom up. *Trends in cognitive sciences*, 7(11):505–512.

Bianca, O. N. and Han, T. A. (2019). Emergence of coordination with asymmetric benefits via prior commitment. In *Artificial Life Conference Proceedings*, pages 163–170. MIT Press.

Chen, X., Sasaki, T., Brännström, Å., and Dieckmann, U. (2015). First carrot, then stick: how the adaptive hybridization of incentives promotes cooperation. *Journal of The Royal Society Interface*, 12(102):20140935.

Chen, X.-P. and Komorita, S. S. (1994). The effects of communication and commitment in a public goods social dilemma. *Organizational Behavior and Human Decision Processes*, 60(3):367–386.

Cherry, T. L. and McEvoy, D. M. (2013). Enforcing compliance with environmental agreements in the absence of strong institutions: An experimental analysis. *Environmental and Resource Economics*, 54(1):63–77.

Cimpeanu, T., Han, T. A., and Santos, F. C. (2019). Exogenous rewards for promoting cooperation in scale-free networks. In *Artificial Life Conference Proceedings*, pages 316–323. MIT Press.

Cimpeanu, T., Perret, C., and Han, T. A. (2021). Cost-efficient interventions for promoting fairness in the ultimatum game. *Knowledge-Based Systems*, page 107545.

Cimpeanu, T., Santos, F. C., Pereira, L. M., Lenaerts, T., and Han, T. A. (2022). Artificial intelligence development races in heterogeneous settings. *Scientific Reports*, 12(1):1–12.

Clemons, E. (1991). Evaluation of strategic investments in information technology. *Communications of the ACM*, 34(1):22–36.

Dall'Asta, L., Baronchelli, A., Barrat, A., and Loreto, V. (2006). Nonequilibrium dynamics of language games on complex networks. *Physical Review E*, 74(3):036105.

Dercole, F., Della Rossa, F., and Piccardi, C. (2019). Direct reciprocity and model-predictive rationality explain network reciprocity over social ties. *Scientific reports*, 9(1):5367.

Di Stefano, A., Scatà, M., Attanasio, B., La Corte, A., Liò, P., and Das, S. K. (2020). A novel methodology for designing policies in mobile crowdsensing systems. *Pervasive and Mobile Computing*, 67:101230.

Di Stefano, A., Scatà, M., La Corte, A., Liò, P., Catania, E., Guardo, E., and Pagano, S. (2015). Quantifying the role of homophily in human cooperation using multiplex evolutionary game theory. *PloS one*, 10(10):e0140646.

Dorogovtsev, S. (2010). *Complex networks*. Oxford: Oxford University Press.

Frank, R. H. (1988). *Passions Within Reason: The Strategic Role of the Emotions*. Norton and Company.

Gracia-Lázaro, C., Cuesta, J. A., Sánchez, A., and Moreno, Y. (2012a). Human behavior in prisoner's dilemma experiments suppresses network reciprocity. *Scientific reports*, 2(1):1–4.

Gracia-Lázaro, C., Ferrer, A., Ruiz, G., Tarancón, A., Cuesta, J. A., Sánchez, A., and Moreno, Y. (2012b). Heterogeneous networks do not promote cooperation when humans play a prisoner's dilemma. *Proceedings of the National Academy of Sciences*, 109(32):12922–12926.

Grujić, J. and Lenaerts, T. (2020). Do people imitate when making decisions? evidence from a spatial prisoner?s dilemma experiment. *Royal Society open science*, 7(7):200618.

Han, T. A. (2013). *Intention Recognition, Commitments and Their Roles in the Evolution of Cooperation: From Artificial Intelligence Techniques to Evolutionary Game Theory Models*, volume 9. Springer SAPERE series.

Han, T. A. (2022). Institutional incentives for the evolution of committed cooperation: Ensuring participation is as important as enhancing compliance. *Journal of the Royal Society Interface*, 19(188):20220036.

Han, T. A., Lenaerts, T., Santos, F. C., and Pereira, L. M. (2022). Voluntary safety commitments provide an escape from over-regulation in ai development. *Technology in Society*, 68:101843.

Han, T. A., Pereira, L. M., and Lenaerts, T. (2015). Avoiding or Restricting Defectors in Public Goods Games? *J. Royal Soc Interface*, 12(103):20141203.

Han, T. A., Pereira, L. M., and Lenaerts, T. (2017). Evolution of commitment and level of participation in public goods games. *Autonomous Agents and Multi-Agent Systems*, 31(3):561–583.

Han, T. A., Pereira, L. M., and Santos, F. C. (2012). Corpus-based intention recognition in cooperation dilemmas. *Artificial Life*, 18(4):365–383.

Han, T. A., Pereira, L. M., Santos, F. C., and Lenaerts, T. (2013). Good agreements make good friends. *Scientific reports*, 3(2695).

Han, T. A., Pereira, L. M., Santos, F. C., and Lenaerts, T. (2020). To Regulate or Not: A Social Dynamics Analysis of an Idealised AI Race. *Journal of Artificial Intelligence Research*, 69:881–921.

Kumar, A., Capraro, V., and Perc, M. (2020). The evolution of trust and trustworthiness. *Journal of The Royal Society Interface*, 17(169):20200491.

Martinez-Vaquero, L. A., Han, T. A., Pereira, L. M., and Lenaerts, T. (2015). Apology and forgiveness evolve to resolve failures in cooperative agreements. *Scientific reports*, 5(10639).

Nesse, R. M. (2001a). *Evolution and the capacity for commitment*. Foundation series on trust. Russell Sage.

Nesse, R. M. (2001b). *Natural selection and the capacity for subjective commitment*. New York: Russell Sage.

Newman, M. E. (2003). The structure and function of complex networks. *SIAM review*, 45(2):167–256.

Newman, M. E. (2004). Coauthorship networks and patterns of scientific collaboration. *Proceedings of the national academy of sciences*, 101(suppl 1):5200–5205.

Nowak, M. A. (2006a). *Evolutionary Dynamics: Exploring the Equations of Life*. Harvard University Press, Cambridge, MA.

Nowak, M. A. (2006b). Five rules for the evolution of cooperation. *Science*, 314(5805):1560.

Ogbo, N. B., Elragig, A., and Han, T. A. (2022). Evolution of coordination in pairwise and multi-player interactions via prior commitments. *Adaptive Behavior*, 30(3):257–277.

Ohtsuki, H., Hauert, C., Lieberman, E., and Nowak, M. A. (2006). A simple rule for the evolution of cooperation on graphs and social networks. *Nature*, 441(7092):502–505.

Ohtsuki, H., Nowak, M. A., and Pacheco, J. M. (2007). Breaking the symmetry between interaction and replacement in evolutionary dynamics on graphs. *Physical review letters*, 98(10):108106.

Ostrom, E. (1990). *Governing the commons: The evolution of institutions for collective action*. Cambridge university press.

Page, K. M., Nowak, M. A., and Sigmund, K. (2000). The spatial ultimatum game. *Proceedings of the Royal Society of London. Series B: Biological Sciences*, 267(1458):2177–2182.

Parsons, G. L. (1983). Information technology: A new competitive weapon. *Sloan management review*, 25(1):3.

Pena, J., Volken, H., Pestelacci, E., and Tomassini, M. (2009). Conformity hinders the evolution of cooperation on scale-free networks. *Physical Review E*, 80(1):016110.

Perc, M., Jordan, J. J., Rand, D. G., Wang, Z., Boccaletti, S., and Szolnoki, A. (2017). Statistical physics of human cooperation. *Physics Reports*, 687:1–51.

Perc, M. and Szolnoki, A. (2010). Coevolutionary games—a mini review. *BioSystems*, 99(2):109–125.

Pereira, L. M., Han, T. A., and Lopes, A. B. (2021). Employing ai to better understand our morals. *Entropy*, 24(1):10.

Perret, C., Krellner, M., and Han, T. A. (2021). The evolution of moral rules in a model of indirect reciprocity with private assessment. *Scientific Reports*, 11(1):1–10.

Pinheiro, F. L., Santos, F. C., and Pacheco, J. M. (2012). How selection pressure changes the nature of social dilemmas in structured populations. *New Journal of Physics*, 14(7):073035.

Pitt, J., Schaumeier, J., and Artikis, A. (2012). Axiomatization of socio-economic principles for self-organizing institutions: Concepts, experiments and challenges. *ACM Transactions on Autonomous and Adaptive Systems (TAAS)*, 7(4):39.

Poncela, J., Gómez-Gardenes, J., Floría, L., and Moreno, Y. (2007). Robustness of cooperation in the evolutionary prisoner's dilemma on complex networks. *New Journal of Physics*, 9(6):184.

Powers, S. T., Taylor, D. J., and Bryson, J. J. (2012). Punishment can promote defection in group-structured populations. *Journal of theoretical biology*, 311:107–116.

Rand, D. G., Tarnita, C. E., Ohtsuki, H., and Nowak, M. A. (2013). Evolution of fairness in the one-shot anonymous ultimatum game. *Proc. Natl. Acad. Sci. USA*, 110:2581–2586.

Sachs, G. (2000). B2b: 2b or not 2b.

Santos, F. C., Pacheco, J. M., and Lenaerts, T. (2006). Evolutionary dynamics of social dilemmas in structured heterogeneous populations. *Proceedings of the National Academy of Sciences of the United States of America*, 103:3490–3494.

Santos, F. C., Santos, M. D., and Pacheco, J. M. (2008). Social diversity promotes the emergence of cooperation in public goods games. *Nature*, 454:214–216.

Santos, F. P., Pacheco, J. M., Paiva, A., and Santos, F. C. (2017). Structural power and the evolution of collective fairness in social networks. *PLoS one*, 12(4):e0175687.

Sasaki, T., Okada, I., Uchida, S., and Chen, X. (2015). Commitment to cooperation and peer punishment: Its evolution. *Games*, 6(4):574–587.

Sayama, H. (2011). Seeking open-ended evolution in swarm chemistry. In *2011 IEEE Symposium on Artificial Life (ALIFE)*, pages 186–193. IEEE.

Schilling, M. A. and Phelps, C. C. (2007). Interfirm collaboration networks: The impact of large-scale network structure on firm innovation. *Management science*, 53(7):1113–1126.

Shipilov, A. and Gawer, A. (2020). Integrating research on interorganizational networks and ecosystems. *Academy of Management Annals*, 14(1):92–121.

Sigmund, K. (2010). *The Calculus of Selfishness*. Princeton University Press.

Sinatra, R., Iranzo, J., Gomez-Gardenes, J., Floria, L. M., Latora, V., and Moreno, Y. (2009). The ultimatum game in complex networks. *Journal of Statistical Mechanics: Theory and Experiment*, 2009(09):P09012.

Szabó, G. and Fáth, G. (2007). Evolutionary games on graphs. *Phys Rep*, 97-216(4-6).

Szolnoki, A., Perc, M., and Szabó, G. (2012). Defense mechanisms of empathetic players in the spatial ultimatum game. *Physical review letters*, 109(7):078701.

Traulsen, A., Nowak, M. A., and Pacheco, J. M. (2006). Stochastic dynamics of invasion and fixation. *Phys. Rev. E*, 74:11909.

Uchida, S., Yamamoto, H., Okada, I., and Sasaki, T. (2019). Evolution of cooperation with peer punishment under prospect theory. *Games*, 10(1):1–14.

West, S., Griffin, A., and Gardner, A. (2007). Evolutionary explanations for cooperation. *Current Biology*, 17:R661–R672.

Wu, T., Fu, F., Zhang, Y., and Wang, L. (2013). Adaptive role switching promotes fairness in networked ultimatum game. *Scientific reports*, 3:1550.

Zhu, K. and Weyant, J. P. (2003). Strategic decisions of new technology adoption under asymmetric information: a game-theoretic model. *Decision sciences*, 34(4):643–675.

Zisis, I., Guida, S. D., Han, T. A., Kirchsteiger, G., and Lenaerts, T. (2015). Generosity motivated by acceptance - evolutionary analysis of an anticipation games. *Scientific reports*, 5(18076).

A Partial Integro-Differential Equation-Based Model of Adaptive Social Network Dynamics

Hiroki Sayama[1,2]
[1]Center for Collective Dynamics of Complex Systems, Binghamton University, State University of New York, USA
[2]Waseda Innovation Lab, Waseda University, Japan
sayama@binghamton.edu

Social fragmentation and opinion polarization are the major social problems these days. While there are many studies on modeling social fragmentation (Centola et al., 2007; Kozma and Barrat, 2008; Böhme and Gross, 2011; Sayama, 2020; Blex and Yasseri, 2020), most of the earlier models typically considered only two societal outcomes: (1) fragmented society with many disconnected, incompatible social clusters and (2) well-connected society in which cultural/opinion states of individuals are homogenized. To seek the possibility of the third alternative social state that maintains *both* social connectivity and cultural/opinion diversity, we had proposed an agent-based adaptive social network model that incorporated behavioral diversity of nodes (agents) (Sayama and Yamanoi, 2020). This model described social network evolution in which cultural information was shared among agents (nodes) and the weights of their connections (edges) were updated according to acceptance or rejection of shared cultural information (Sayama et al., 2013; Yamanoi and Sayama, 2013). The model showed that, when the cultural tolerance levels of agents were diversified within society, social evolution could lead to a culturally diverse yet structurally connected state. However, this model was entirely computational and the simulation experiments were done with relatively small-sized network topologies, and hence it did not provide much theoretical insight into how a much larger social network might behave according to the same model assumptions.

In the current study (Sayama, 2022), we have converted the rather complex, discrete, *if-then*-rule-based original ABM into a smooth, continuous, analytically tractable equation-based one so that we can gain more theoretical insight into the model dynamics. To make it easier to represent the configuration of the network that involves heterogeneity in behavioral traits of nodes, we have restricted the attributes and states of each node to one-dimensional continuous values and have assumed that the size of the network is very large, which allows for continuous representation. As a result, we have represented the whole system as a set of partial integro-differential equations (PIDEs) about just two continuous state functions: population density function $p(v, d, t)$—density of nodes with opinion v and cultural tolerance d at time t, and connection density function $c(v, u, d_v, d_u, t)$—density of directed edges from nodes with opinion v and cultural tolerance d_v to nodes with opinion u and cultural tolerance d_u at time t. The resulting set of equations developed is given below (details of the model assumptions, parameters and derivation process can be found in (Sayama, 2022)):

$$\frac{\partial p}{\partial t} = D_p \frac{\partial^2 p}{\partial v^2} - \alpha \frac{\partial}{\partial v}\left[p\, T_p(v, d_v)\right] \tag{1}$$

$$\frac{\partial c}{\partial t} = D_c \left(\frac{\partial^2 c}{\partial v^2} + \frac{\partial^2 c}{\partial u^2}\right) + \beta c\, R_c(v, u, d_v, d_u) - \gamma \frac{\partial}{\partial v}\left[c\, T_c(v, u, d_v, d_u)\right] \tag{2}$$

With:

$$T_p(v, d_v) = \int c_{v,d_v}\, P_a(v - u, d_v)\, r_s(u - v)\, \mathrm{d}u \tag{3}$$

$$c_{v,d_v} = \frac{\int c\, \mathrm{d}d_u}{\int\int c\, \mathrm{d}d_u \mathrm{d}u} \tag{4}$$

$$P_a(x, d) = \left(\frac{1}{2}\right)^{\frac{|x|}{d}} \tag{5}$$

$$\begin{aligned} R_c(v, u, d_v, d_u) = {} & (1 - P_a(v - u, d_v)) \\ & (\mathrm{logistic}(\mathrm{logit}(c) - r_w) - c) \\ & + P_a(v - u, d_v) \\ & (\mathrm{logistic}(\mathrm{logit}(c) + r_w) - c) \end{aligned} \tag{6}$$

$$T_c(v, u, d_v, d_u) = P_a(v - u, d_v)\, r_s(u - v) \tag{7}$$

Eqs. (1) and (2) collectively represent the continuous dynamics of population and connection densities based on the assumptions used in the original ABM. Both equations have diffusion terms along the node state axes. The second term of Eq. (1) represents the aggregated migration of populations in the opinion space because of the acceptance of opinions shared by neighbors, whose trend is given in Eq. (3).

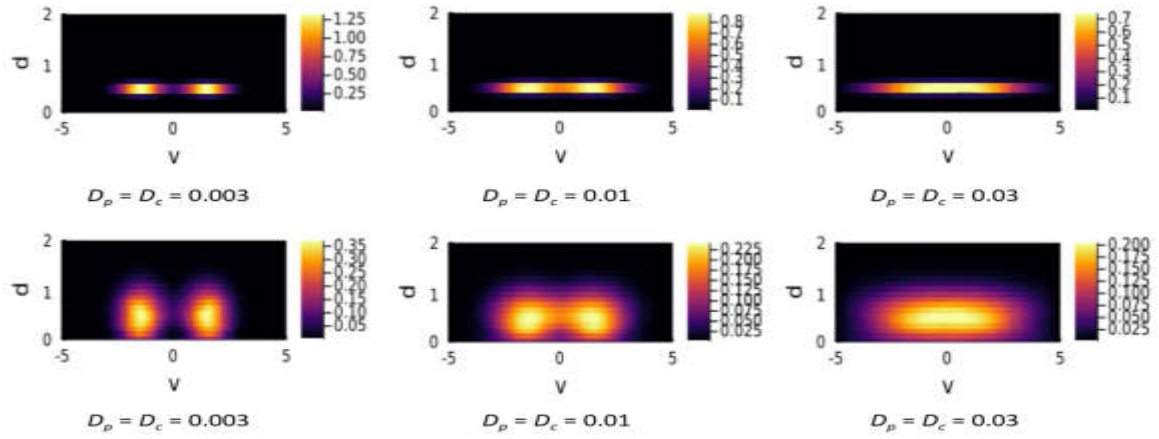


Figure 1: Final states of $p(v, d)$ at $t = 50$, where v is the opinion state and d is the cultural tolerance level. Top: With low diversity of cultural tolerance ($\sigma(d) = 0.1$). As the diffusion coefficients varied, the model exhibited typical social fragmentation transition between two separated clusters (left) and a single unified cluster (right). Bottom: With high diversity of cultural tolerance ($\sigma(d) = 0.5$). There appeared a new third state (center; $D_p = D_c = 0.01$) in which agents with greater cultural tolerance (d) approached the center and began connecting the two opinion clusters while agents with smaller d remained holding their original opinions.

The second term of Eq. (2) represents the decrease/increase of connection weights (defined in Eq. (6)) due to rejection/acceptance of neighbors' opinions. The third term of Eq. (2) represents the migration of connectivities in the opinion space because of the acceptance of opinions shared by neighbors, whose trend is given in Eq. (7). Eq. (4) is the normalized probability density function of connectivity from a node with opinion v and cultural tolerance d_v to a neighbor with opinion u (irrespective of the neighbor's cultural tolerance). Eq. (5) is the acceptance probability function for opinion difference $|x|$ and cultural tolerance d. These equations are a concrete demonstration of how one can convert rule-based complex ABM rules into continuous mathematical equations.

We have conducted numerical integrations of the developed equations using a custom-made numerical integrator written in Julia (specific details of the methods and parameter settings can be found in (Sayama, 2022)). Figure 1 shows the final states of population density function $p(v, d)$ at $t = 50$ for low (top) / high (bottom) diversity of cultural tolerance d and three different values of diffusion coefficients ($D_p = D_c = 0.003$ (left), 0.01 (center), and 0.03 (right)). The diffusion represents the stochastic nature of agents' behaviors in this otherwise deterministic PIDE model. When the diversity of cultural tolerance d is low ($\sigma(d) = 0.1$; Fig. 1 top), if diffusion is weak, the network remains separated in the initial two opinion clusters (Fig. 1 top left). If diffusion is stronger, however, the two initial opinion clusters become blurred and merge into a single cluster (Fig. 1 top right). This corresponds to the typical social fragmentation transition that has been reported in many earlier models.

Meanwhile, when the diversity of cultural tolerance d is high enough ($\sigma(d) = 0.5$; Fig. 1 bottom), a different social state emerges. The new social state is observed in the scenario with intermediate diffusion strength (Fig. 1 bottom center, $D_p = D_c = 0.01$), in which agents with greater cultural tolerance (d) approached the center and began connecting the two opinion clusters while agents with smaller d remained holding their original opinions. This resulted in a "frown ($\frown$)"-like shape of the $p(v, d)$ distribution, which represents diverse yet connected social networks. Such social states were not reported in conventional social fragmentation studies.

The main finding consistently obtained in both ABM and PIDE models is that, when the cultural tolerance levels of constituents are diverse enough ($\sigma(d) \geq 0.3$), the adaptive social network can self-organize into a configuration with multiple well-established opinion clusters together with bridges that connect them (Sayama, 2022). This is a possible alternative third state of society that goes beyond complete fragmentation or complete homogenization, the two only possible outcomes studied in typical social fragmentation models. This third state may be a more desirable state of *society-as-it-could-be*, since it maintains informational diversity and information exchange simultaneously, possibly generating more innovation. Additional insights gained from the PIDE model include the role of stochasticity (diffusion) as the main factor of social fragmentation transition and the overall "phase space" structure of social evolution outcomes that depend on the agents' stochasticity and behavioral diversity (Sayama, 2022).

Acknowledgments

We thank Junichi Yamanoi for his contribution to the original ABM-based study on which the present study is based. This work was supported by JSPS KAKENHI Grant Number 19H04220.

References

Blex, C. and Yasseri, T. (2020). Positive algorithmic bias cannot stop fragmentation in homophilic networks. *The Journal of Mathematical Sociology*, pages 1–18.

Böhme, G. A. and Gross, T. (2011). Analytical calculation of fragmentation transitions in adaptive networks. *Physical Review E*, 83(3):035101.

Centola, D., Gonzalez-Avella, J. C., Eguiluz, V. M., and San Miguel, M. (2007). Homophily, cultural drift, and the co-evolution of cultural groups. *Journal of Conflict Resolution*, 51(6):905–929.

Kozma, B. and Barrat, A. (2008). Consensus formation on adaptive networks. *Physical Review E*, 77(1):016102.

Sayama, H. (2020). Enhanced ability of information gathering may intensify disagreement among groups. *Physical Review E*, 102(1):012303.

Sayama, H. (2022). Representing and analyzing the dynamics of an agent-based adaptive social network model with partial integro-differential equations. *Northeast Journal of Complex Systems (NEJCS)*, 4(1):3.

Sayama, H., Pestov, I., Schmidt, J., Bush, B. J., Wong, C., Yamanoi, J., and Gross, T. (2013). Modeling complex systems with adaptive networks. *Computers & Mathematics with Applications*, 65(10):1645–1664.

Sayama, H. and Yamanoi, J. (2020). Beyond social fragmentation: Coexistence of cultural diversity and structural connectivity is possible with social constituent diversity. In *International Conference on Network Science*, pages 171–181. Springer.

Yamanoi, J. and Sayama, H. (2013). Post-merger cultural integration from a social network perspective: A computational modeling approach. *Computational and Mathematical Organization Theory*, 19(4):516–537.