Step Size is a Consequential Parameter in Continuous Cellular Automata

Q. Tyrell Davis[1] and Josh Bongard[1]

[1]University of Vermont, Burlington, VT 05405
qdavis & jbongard @uvm.edu

Introduction

In the 1960s, John H. Conway developed a zero-player game with simple rules. This Game of Life, a cellular automaton (CA), has had a seminal impact on the study of complex systems, computation, and art. Conway's Life followed John von Neumann's 29-state CA (von Neumann and Burks, 1966), and Life's impact on popular as well as academic imagination is unique, seeded by a 1970 article in Scientific American by Martin Gardner (Gardner, 1970).

Subsequent decades saw increasing diversity of CA research and applications. The Life framework: totalistic CA based on a Moore neighborhood and cell states, supports 262,144 different rulesets[1] CA have since been developed with larger neighborhoods (Evans, 2001; Pivato, 2007), higher dimensions (Bays, 1987; Chan, 2020), and evolving rules (McCaskill and Packard, 2019), to name but a few examples among many. In this work we are concerned with continuous CA.

Rudy Rucker developed a continuous CA framework in the 1990s called CAPOW (Rucker, 2003). Later, Stephen Rafler developed a continuous CA, SmoothLife, and discovered a persistent glider therein (Rafler, 2011). Recently, Bert Chan described the discovery of many persistent patterns in his Lenia framework (Chan, 2019, 2020), and encoding CA updates in continuously valued neural networks has been applied to models for growth (Mordvintsev et al., 2020), image recognition (Randazzo et al., 2020), and control (Variengien et al., 2021). Continuous CA can be described as:

$$A(t + dt) = A(t) + dt \cdot f(A(t)) \quad (1)$$

where A represents cell states, t is unitless time, dt is step size, and $f(\cdot)$ represents some function dependent on cell states[2]

[1]Life-like CA are defined by **B**irth and **S**urvival rules; each may contain any or all of the possible Moore neighborhood sum values 0 through 8, yielding $2^9 \cdot 2^9 = 2^{18} = 262,144$ possible rules. At each step, cells with a neighborhood sum in their S rules remain unchanged, become 1 with a sum in B; all other cells become 0.

[2]$f(\cdot)$ typically includes a neighborhood operation (*e.g.* convolution) and an arbitrary update function.

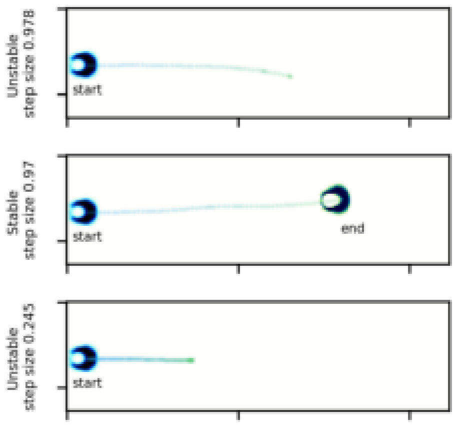


Figure 1: Stability of a minimal glider in Lenia's *Scutium gravidus* ruleset spans from approximately $dt = 0.25$ to 0.97. Kernel parameters $(\mu_K, \sigma_K) = (0.5, 0.15)$, update parameters $(\mu_G, \sigma_G) = (0.283, 0.0369)$.

Typically receiving little scrutiny, dt can have important effects. Persistent patterns require step size to be neither too large nor too small, and multiple patterns may exist their own dt ranges within an otherwise identical CA rule set (Figures 2 and 3). Step size can also lead to qualitatively different behavior in CA. Varying step size from 0.0125 to 0.13 in Figure 3 yields diverse movement types including hopping, meandering, and corkscrewing. While not a perfect analogy, the behavioral repertoire in Figure 3 seems to have more in common with a robot in a conventional physics simulation changing from jumping to pirouette movement patterns than with the expected catastrophic failure (or tedious slow-down) caused by an inappropriate time-step in a conventional differential equation-based numerical physics simulation. Supporting resources for this project are open-source [3]

[3]Links to animations, notebooks, and code used in this project are consolidated at `https://rivesunder.github.io/yuca`

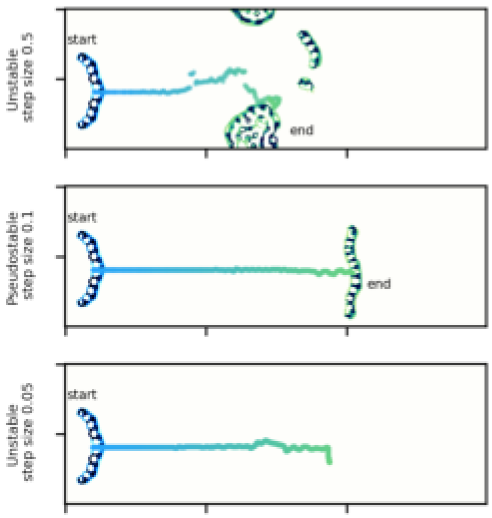


Figure 2: This wide glider pattern is pseudostable around $dt = 0.1$, but tends to change size and shape (sometimes with explosive growth) at higher dt, and to disappear at lower dt. Kernel parameters $(\mu_K, \sigma_K) = (0.5, 0.15)$, update parameters $(\mu_G, \sigma_G) = (0.283, 0.0369)$.

Pattern stability and step size

An intuitive consequence of poor step size choice is that patterns become unstable when the step size is too large, but this also occurs when step size is too small. This defies the expectation that in simulations of physical system we typically expect greater accuracy as the step size approaches zero. What would be considered systematic error in simulating billiard ball trajectories is essential for self-organization by some CA patterns.

A minimal glider *Scutium gravidus* in the Lenia framework, cousin to the SmoothLife glider (Rafler, 2011), is stable between approximately $dt = 0.245$ to 0.978, smaller or larger step sizes lead to the glider vanishing. Sample trajectories at both extremes are shown in Figure 1.

Another pattern operating under the same neighborhood and update rules, a wide glider, is most stable at much lower step sizes around 0.1. Larger or smaller step sizes yield unconstrained growth or a vanishing pattern, respectively (Figure 2). A step size of 0.1 is pseudo-stable for this pattern, but is sensitive to initial conditions (including grid dimensions, pattern placement, and floating point precision) and can eventually ($\sim$1000s of steps) become unstable.

Pattern behavior and step size

In addition to pattern instability, step size can lead to qualitatively different behavior. Figure 3 demonstrates behavioral diversity solely by changing step size. At the "natural" step size of 0.1, it moves in a "hopping" motion. Relatively large step sizes (≈ 0.13) bring the pattern to the edge of stability, with occasional explosive growth (responsible for a sharp turn in trajectory), and also occasional larger, surging hops. At step size 0.025 the pattern travels in a meandering trajectory and a step size of 0.0125 leads to corkscrew trajectories and a spiky morphology. The CA in Figure 3 uses the Glaberish CA framework (Davis and Bongard, 2022), splitting the update function into persistence and genesis functions dependent on cell state, with a neighborhood kernel of mixed Gaussians with parameters $(\mu_K, \sigma_K) = [(0.0938, 0.033), (0.2814, 0.033), (0.469, 0.033)]$ with weights $[0.5, 1.0, 0.667]$, a Gaussian genesis function $(\mu_g, \sigma_g) = (0.0621, 0.0088)$, and persistence function $(\mu_p, \sigma_p) = (0.2151, 0.0369)$.

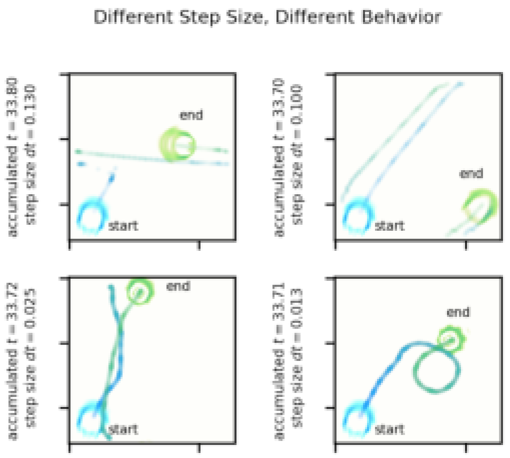


Figure 3: Identical starting conditions, here a mobile "frog" pattern, follow different trajectories with qualitatively different behavior under CA rules differing only in step size dt. Note the CA grid represents the surface of a torus: trajectories reaching one edge of the grid reappear at the opposite edge.

Conclusions

This work demonstrates that step size is a consequential parameter in continuous CA, affecting pattern stability and qualitative behavior. This is in marked contrast to remarks in (Chan, 2019), which, noting the resemblance of the Lenia update to Euler's method, suggested that decreasing step size asymptotically approaches the ideal simulation of a Lenia pattern, *Orbium*. This work demonstrates that for several patterns a lower step size does not entail a more accurate simulation, but different behavior or potential patterns entirely. Therefore, step size should be considered for optimization and learning with CA, *e.g.* to develop patterns with agency to avoid obstacles (Hamon et al., 2022) or for training neural CA such as in(Mordvintsev et al., 2020; Randazzo et al., 2020; Variengien et al., 2021).

Funding

This work was supported by the National Science Foundation under the Emerging Frontiers in Research and Innovation (EFRI) program (EFMA-1830870).

References

Bays, C. (1987). Candidates for the Game of Life in three dimensions. *Complex Syst.*, 1.

Chan, B. W.-C. (2019). Lenia - biology of artificial life. *Complex Systems*, 28(3):251–286.

Chan, B. W.-C. (2020). Lenia and expanded universe. *The 2020 Conference on Artificial Life*, pages 221–229.

Davis, Q. T. and Bongard, J. (2022). Glaberish: generalizing the continuously-valued Lenia framework to arbitrary Life-like cellular automata. *ALIFE 2022: The 2022 Conference on Artificial Life*.

Evans, K. M. (2001). Larger than Life: Digital creatures in a family of two-dimensional cellular automata. *Discrete Mathematics & Theoretical Computer Science*, DMTCS Proceedings vol. AA,...:2288.

Gardner, M. (1970). Mathematical games: The fantastic combinations of John Conway's new solitaire game "Life". *Scientific American*, 223:120–123.

Hamon, G., Etcheverry, M., Chan, B. W.-C., Moulin-Frier, C., and Oudeyer, P.-Y. (2022). Learning sensorimotor agency in cellular automata. https://developmentalsystems.org/sensorimotor-lenia/.

McCaskill, J. S. and Packard, N. H. (2019). Analysing emergent dynamics of evolving computation in 2D cellular automata. *TPNC*, pages 3–40.

Mordvintsev, A., Randazzo, E., Niklasson, E., and Levin, M. (2020). Growing neural cellular automata. *Distill*. https://distill.pub/2020/growing-ca.

Pivato, M. (2007). RealLife: the continuum limit of Larger than Life cellular automata. *Theoretical Computer Science*, 372(1):46–68.

Rafler, S. (2011). Generalization of Conway's "Game of Life" to a continuous domain - SmoothLife. *arXiv:1111.1567 [nlin]*.

Randazzo, E., Mordvintsev, A., Niklasson, E., Levin, M., and Greydanus, S. (2020). Self-classifying MNIST digits. *Distill*. https://distill.pub/2020/selforg/mnist.

Rucker, R. (2003). *Continuous-Valued Cellular Automata in Two Dimensions. in: New Constructions in Cellular Automata*. Oxford University Press. doi: 10.1093/oso/9780195137170.003.0016, ISBN: 978-0-19-513717-0 978-0-19-756165-.

Variengien, A., Pontes-Filho, Sidny abd Glover, T., and Nichele, S. (2021). Towards self-organized control: Using neural cellular automata to robustly control a cart-pole agent. *Innovations in Machine Intelligence*, 1:1–14.

von Neumann, J. and Burks, A. W. (1966). *Theory of self-reproducing automata*. Urbana, University of Illinois Press.

Lifeforms potentially useful for automated underwater monitoring systems

Wiktoria Rajewicz[1], Donato Romano[2], Joshua Varughese[1], Thomas Schmickl[1] and Ronald Thenius[1]
[1]University of Graz, Austria, [2]Scuola Superiore Sant'Anna, Pisa, Italy
corresponding author: wiktoria.rajewicz@uni-graz.at

Abstract

Biohybrids combine artificial robotic elements with living organisms. These novel technologies allow for obtaining useful data on the environment by implementing organisms as "living sensors". Natural water resources are under serious ecological threat and there is always a need for new, more efficient methods for aquatic monitoring. Project Robocoenosis introduces the use of biohybrid entities as low-cost and long-term environmental monitoring devices. This will be done by combining lifeforms with technical parts which will be powered with the use of MFCs. This concept will allow for a more well-rounded data collection and provide an insight into the water body with minimal human impact.

Introduction

Currently worsening environmental conditions pose a threat to global water supplies and communities inhabiting them. This trend causes for urgent need to gather water quality data globally and more efficiently. Traditionally, water monitoring studies are carried out with surveys, sample collecting and various sensors. This provides a precise and reliable readings on the examined water body. However, due to the immense complexity of aquatic environments, a need arose for a more extensive and continuous water monitoring. Project Robocoenosis aims to develop a novel methodology for that purpose, with the use of "biohybrid entities" (Thenius et al., 2021). These devices are made by combining the electrical and mechanical robotic parts with living organisms. This system allows for extracting useful information from the organisms which respond instantly to changes in the environment with various behavioural or physiological changes. The biohybrid will also be self-powering by being equipped with Microbial Fuel Cells (MFC). These structures consist of an anode and cathode chamber which extract electricity by relying on natural processes occurring in the sediments and the availability of organic matter. Here, the focus is placed on lifeforms and devices that can potentially provide valuable insight into the aquatic communities and environmental changes affecting them.

It is of merit to note that the concept of using organisms as biosensors is not entirely new (Kirsanov et al., 2014; Fini et al., 2009). The project Robocoenosis will combine existing technologies (Wang et al., 2015; EnviroAnalytical, 2021; Dzierżyńska-Białończyk et al., 2019) and well-researched organisms to construct a new, self-sufficient biohybrid. This concept aims to compliment traditional monitoring methods, providing an additional insight into the water quality. The combination of different lifeforms will allow a, so called, ecosystem hacking.

Methods

Lifeforms

Zebra mussel For the sake of biological safety, it is essential to use lifeforms from the respective environments to avoid introducing alien species (Stiers et al., 2011; Guzman-Novoa et al., 2020). First lifeform investigated for the underwater monitoring setup is zebra mussel *Dreissena polymorpha* (Pallas, 1771). It is a broadly distributed species considered invasive in most of Europe. Mussels have been considered useful bioindicators due to their longevity, clustered colonies and high vulnerability to the surrounding waters due to their feeding and breathing mechanisms (Grabarkiewicz and Davis, 2008). Its sensitivity to various sensors, especially low oxygen and heavy metals, often results in abnormal behaviour, such as closing the valves. This can be observed by attaching it to the robotic setup and observing its movements with continuous image analysis (Figure 1). When the valves present fluttering movements or are closed for a prolonged period of time, an alarm will be triggered to announce a potential disturbance to the environment.

Daphnia Another organism considered a useful bioindicator is the water flea *Daphnia* sp. (Müller, 1785). It is highly sensitive to many environmental stressors including changing salinity, oxygen levels, heavy metals and many others (Michels et al., 2000). Stress responses can include abnormal swimming (slower speed, increased sinking, spinning), disrupted phototaxy (light-related responses), haemoglobin accumulation and mortality (Rajewicz et al., 2021). These behaviours will be observed by continuous movement track-

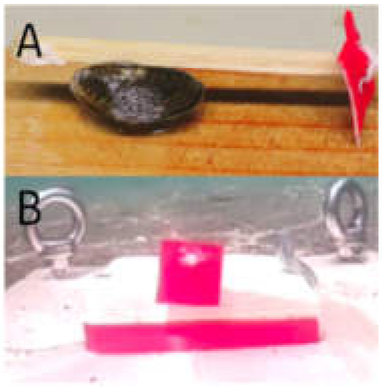


Figure 1: Zebra mussel mounted on the underwater setup A: red tag mounted on the top valve and B: view on the mussel from the camera attached to the setup.

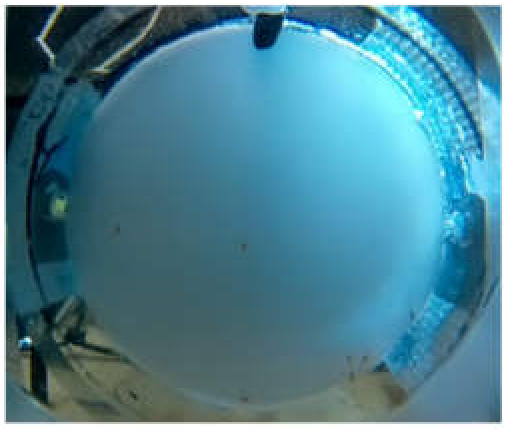


Figure 2: Daphnia swimming in the flow-through chamber with background illumination as part of the monitoring setup.

ing of the swimming animals enclosed in the setup. A flow-through cage was built where *Daphnia* can swim freely while being exposed directly to the surrounding waters (Figure 2). This will also allow the animals to sustain themselves normally by feeding on the algae present in the water. Because of the stress reactions occurring immediately after the exposure to the stressor, *Daphnia* can be a reliable source of information on the water condition.

Microbial Fuel Cells

Microbial Fuel Cell will be explored as a power source alternative as well as an additional bioindicator. Thanks to their working principles, MFCs can provide information on dissolved oxygen, toxins, bacterial activity and others. These can be monitored by tracking the electrical current produced and delivered to the biohybrid entity (Cui et al., 2019). Certain species of bacteria, such as *Geobacter sulfurreducens*, are able to perform aerobic breathing and produce an electric current (Poddar and Khurana, 2011). This proves a feasible method of a long-term production of electricity supplies that can be harvested by the MFC. The major limitation of this energy-harvesting method is that the currents provided are low in voltage and difficult to amplify. This will be resolved by using multiple MFCs connected by a super capacitor. Additionally, low-power electronics will be used for short periods of time and be put in sleep mode between the monitoring periods. This energy will be stored and used for powering micro-controllers and other low-energy devices contained within the biohybrid.

Discussion

In this work, certain ideas and concepts used in the project Robocoenosis are presented. With the use of biohybrid entities, environmental monitoring can be performed more efficiently by removing the time usually used for sampling surveys, enabling a faster detection of catastrophic events, such as leakages of toxins, heavy metals, etc. Using behavioural and physiological reactions as natural sensors requires the use of well-researched organisms and meticulously planned environmental setups. Data obtained with this methodology will come with a degree of ambiguity due to the complexity of the aquatic environment and varying degrees to which they affect the living organisms. For this reason, the method developed is meant to compliment the classical sensor approach. The use of organisms will allow for monitoring the overall quality of the water body and reduce the need for maintenance and costs usually associated with using classical probes and sensors. The big advantage of biohybrid entities compared to fully artificial systems is that lifeforms have the ability to self-calibrate and self maintain. Compared to technological systems that usually need frequent maintenance (ThermoScientific, 2007), a simple observation of lifeforms leaves a longer maintenance break. In combination with energy harvesting and low-power electronics (MFC and STM32 microcontroller), it is possible to create an ultra-long runtimes of possibly several years. This leads to long-lasting, affordable and reliable measuring system. In theory, the system works with no maintenance, although the practical limitations will be investigated.

Acknowledgements

This work was supported by EU-H2020 Project Robocoenosis, grant agreement No 899520. Furthermore this work was supported by the COLIBRI initiative at the University of Graz. We further thank the “Österreichische Bundesforste AG”, the “Burgenländische Landesregierung, Abt. 4 (Natur- und Klimaschutz)” and the “Biologische Station Illmitz” for their immense support.

References

Cui, Y., Lai, B., and Tang, X. (2019). Microbial fuel cell-based biosensors. *Biosensors*, 9(3):92.

Dzierżyńska-Białończyk, A., Jermacz, Ł., Zielska, J., and Kobak, J. (2019). What scares a mussel? Changes in valve movement pattern as an immediate response of a byssate bivalve to biotic factors. *Hydrobiologia*, 841(1):65–77.

Enviro-Analytical (2021). Daphnia toximeter II. A powerful instrument for water toxicity assessment- continuous visual analysis of *Daphnia* behaviour. http://www.enviro-analytical.com/preview/enviroproducts/daphnia_toximeter.html. Accessed on 25.02.2021.

Fini, J.-B., Pallud-Mothré, S., Le Mével, S., Palmier, K., Havens, C. M., Le Brun, M., Mataix, V., Lemkine, G. F., Demeneix, B. A., Turque, N., and Johnson, P. E. (2009). An innovative continuous flow system for monitoring heavy metal pollution in water using transgenic *Xenopus laevis* tadpoles. *Environmental Science & Technology*, 43(23):8895–8900.

Grabarkiewicz, J. and Davis, W. (2008). An introduction to freshwater mussels as biological indicators. U.S. Environmental Protection Agency, Office of Environmental Information, Washington, DC.

Guzman-Novoa, E., Morfin, N., De la Mora, A., Macías-Macías, J. O., Tapia-González, J. M., Contreras-Escareño, F., Medina-Flores, C. A., Correa-Benátez, A., and Quezada-Euán, J. J. G. (2020). The process and outcome of the africanization of honey bees in mexico: lessons and future directions. *Frontiers in Ecology and Evolution*, 8.

Kirsanov, D., Legin, E., Zagrebin, A., Ignatieva, N., Rybakin, V., and Legin, A. (2014). Mimicking *Daphnia magna* bioassay performance by an electronic tongue for urban water quality control. *Analytica chimica acta*, 824:64–70.

Michels, E., Semsari, S., Bin, C., and De Meester, L. (2000). Effect of sublethal doses of cadmium on the phototactic behavior of *Daphnia magna*. *Ecotoxicology and environmental safety*, 47:261–5.

Poddar, S. and Khurana, S. (2011). Geobacter: the electric microbe! efficient microbial fuel cells to generate clean, cheap electricity. *Indian journal of microbiology*, 51(2):240.

Rajewicz, W., Romano, D., Varughese, J. C., Vuuren, G. J. V., Campo, A., Thenius, R., and Schmickl, T. (2021). Freshwater organisms potentially useful as biosensors and power-generation mediators in biohybrid robotics. *Biological cybernetics*, 115(6):615–628.

Stiers, I., Crohain, N., Josens, G., and Triest, L. (2011). Impact of three aquatic invasive species on native plants and macroinvertebrates in temperate ponds. *Biological Invasions*, 13(12):2715–2726.

Thenius, R., Rajewicz, W., Cherian Varughese, J., Schoenwetter-Fuchs, S., Arvin, F., J. Casson, A., Wu, C., Lennox, B., Campo, A., Jansen van Vuuren, G., Stefanini, C., Romano, D., and Schmickl, T. (2021). Biohybrid entities for environmental monitoring. In *Proceedings of The 2021 Conference on Artificial Life, ALIFE, University of Chemistry and Technology Prague*.

ThermoScientific (2007). Dissolved oxygen probe maintenance. https://www.fondriest.com/pdf/thermo_do_maint_manual.pdf. Accessed on 17.05.2022.

Wang, H., Park, J.-D., and Ren, Z. J. (2015). Practical energy harvesting for microbial fuel cells: A review. *Environmental Science & Technology*, 49(6):3267–3277.

Network Diversity Promotes Safety Adoption in Swift Artificial Intelligence Development

Theodor Cimpeanu[1], Francisco C. Santos[3], Luís Moniz Pereira[2], Tom Lenaerts[4,5], and The Anh Han[1,⋆]
[1] School of Computing, Engineering and Digital Technologies, Teesside University
[2] NOVA Laboratory for Computer Science and Informatics (NOVA-LINCS), Universidade Nova de Lisboa
[3]INESC-ID and Instituto Superior Técnico, Universidade de Lisboa
[4] Machine Learning Group, Université Libre de Bruxelles
[5] Artificial Intelligence Lab, Vrije Universiteit Brussel

Abstract

Regulating the development of advanced technology such as Artificial Intelligence (AI) has become a principal topic, given the potential threat they pose to humanity's long term future. First deploying such technology promises innumerable benefits, which might lead to the disregard of safety precautions or societal consequences in favour of speedy development, engendering a race narrative among firms and stakeholders due to value erosion. Building upon a previously proposed game-theoretical model describing an idealised technology race, we investigated how various structures of interaction among race participants can alter collective choices and requirements for regulatory actions. Our findings indicate that strong diversity among race participants, both in terms of connections and peer-influence, can reduce the conflicts which arise in purely homogeneous settings, thereby lessening the need for regulation.

Introduction

Researchers and stakeholders alike have urged for due diligence in regard to AI development on the basis of several concerns. The desire to be at the foreground of the state-of-the-art, or the pressures imposed by upper management, might tempt developers to ignore safety procedures or apprehensions about ethical consequences (Armstrong et al., 2016; Cave and ÓhÉigeartaigh, 2018). Regulation and governance of advanced technologies such as Artificial Intelligence (AI) have become increasingly more important given their potential implications, such as for associated risks and ethical issues (European Commission, 2020; Declaration, 2018; Russell et al., 2015; Future of Life Institute, 2015, 2019). With the tremendous benefits promised from being first able to supply such technologies, stake-holders might cut corners on safety precautions in order to ensure rapid deployment, in a race towards AI market supremacy (AIS) (Armstrong et al., 2016; Cave and ÓhÉigeartaigh, 2018).

With this aim in mind, a baseline model of an innovation race has been recently proposed (Han et al., 2020), in which innovation dynamics are pictured through the lens of Evolutionary Game Theory (EGT) and where all race participants are equally well-connected in the system. The baseline results have showed the importance of accounting for different time-scales of development, and also exposed the dilemmas that arise when what is individually preferred by developers differs from what is globally beneficial. However, real-world stakeholders and their interactions are far from homogeneous (Schilling and Phelps, 2007; Newman, 2004; Barabasi, 2014). Some individuals are more influential than others, or play different roles in the unfolding of new technologies. Technology races are shaped by complex networks of exchange, influence, and competition where diversity abounds. Here we summarise a recent work (Cimpeanu et al., 2022) studying impacts of network topology on the adoption of safety measures in innovation dynamics.

Models and Methods

Assuming that winning the race towards supremacy is the goal of the development teams and that a number of development steps are required, the players have two strategic options at each step: to follow safety precautions (denoted by SAFE) or to ignore them (denoted by UNSAFE) (Han et al., 2020). As it takes more time and effort to comply with the precautionary requirements, playing SAFE is not only costlier but also implies a slower development speed, compared to playing UNSAFE. Let us also assume that to play SAFE players need to pay additional costs. The interactions are iterated until one or more teams achieve a designated objective, after having completed W development steps. As a result, the players obtain a large benefit B, shared among those who reach the target objective at the same time. However, a setback or disaster can happen with some probability, which is assumed to increase with the number of times the safety requirements have been omitted by the winning team(s). Although many potential AI disaster scenarios have been sketched (Armstrong et al., 2016; Pamlin and Arm-

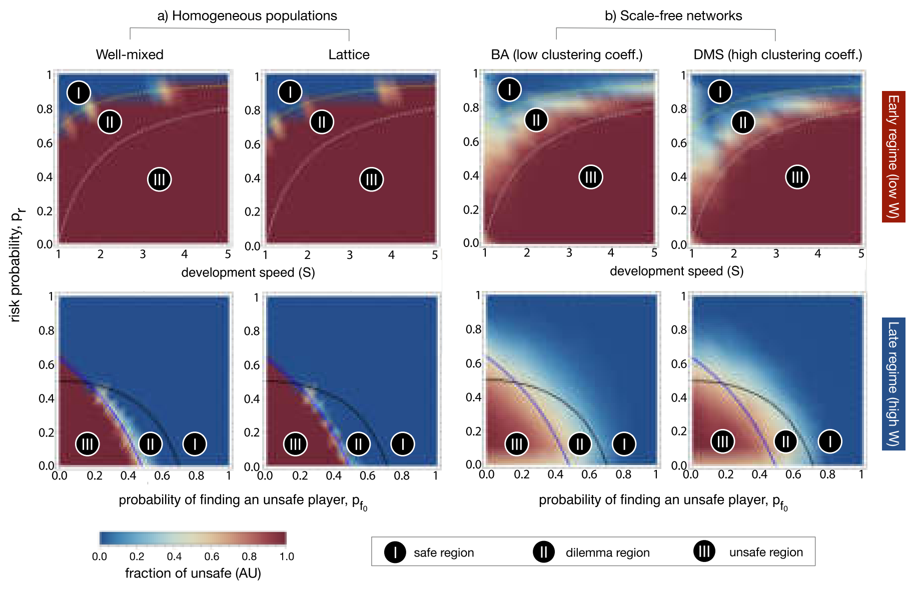


Figure 1: Colour gradients indicating the average fraction of AU (unsafe strategy) for (a) homogeneous (well-mixed and lattices) populations and (b) scale-free networks (BA and DMS models). In the early regime, region II indicates the parameters in which safe AI development is the preferred collective outcome, but unsafe development is expected to emerge and regulation may be needed. In regions I and III, safe and unsafe AI development, respectively, are both preferred collective outcomes and expected to emerge from self-organization. In the late regime, the solid black line marks the boundary above which safety is the preferred outcome, whereas the blue line indicates the boundary above which safety becomes risk dominant against unsafe development.

strong, 2015; Han et al., 2019, 2022), the uncertainties in accurately predicting these outcomes are high. When such a disaster occurs, risk-taking participants lose all their benefits. We denote by p_r the risk probability of such a disaster occurring when no safety precaution is followed at all.

To study the effect of network structures on the safety outcome, we have analysed the following types of networks, from simple to more complex (Cimpeanu et al., 2022): well-mixed populations (complete graph), where each agent interacts with all other agents in a population; square lattice of size with periodic boundary conditions; and scale-free (SF) networks (Barabási and Albert, 1999; Dorogovtsev, 2010; Newman, 2003), generated by means of two growing network models — the widely-adopted Barabási-Albert (BA) model (Barabási and Albert, 1999; Albert and Barabási, 2002) and the Dorogovtsev-Mendes-Samukhin (DMS) model (Dorogovtsev, 2010), the latter of which allowed us to assess the role of a large number of triangular motifs (i.e. high clustering coefficient).

Results and Conclusions

We initially considered the roles of degree-homogeneous graphs in the evolution of safety in the AI race game. They simulated the AI race game in well-mixed populations (Figure 1, first column), and then explored the same game on a square lattice, where each agent can interact with its four edge neighbours (Figure 1, second column). They show that the trends remain the same when compared with well-mixed populations, with very slight differences in numerical values between the two. That is, homogeneous spatial variation is not enough to influence safe technological development.

Investigating beyond homogeneous structures, we make use of two SF network models. Contrary to the findings on homogeneous networks, SF structures produce marked improvements in almost all parameter regions of the AI race game (Figure 1). Given that innovation in the field of AI (more broadly, technological advancement), should be profitable (and robust) to developers, shareholders and society altogether, it is important to discuss the analytical loci where these objectives can be fulfilled. Assuredly, it is observed that diversity in players introduces two marked improvements in both early and late safety regimes. Firstly, very little regulation is required in the case of a late AI race, principally concerning the existing observations in homogeneous settings. Intuitively, this suggests that there is little encouragement needed to promote risk-taking in late AIS regimes: diversity enables beneficial innovation. Secondly, the region for early AIS regimes in which regulation must be enforced is diminished, but not completely eliminated. Consequently, governance should still be prescribed when developers are racing towards an early or otherwise uncertain timeline to reaching transformative AI. It stands to reason that insight into what regime type the AI race operates in is therefore paramount to the success of any potential regulatory actions.

References

Albert, R. and Barabási, A.-L. (2002). Statistical mechanics of complex networks. *Reviews of Modern Physics*, 74(1):47.

Armstrong, S., Bostrom, N., and Shulman, C. (2016). Racing to the precipice: a model of artificial intelligence development. *AI & SOCIETY*, 31(2):201–206.

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.

Cave, S. and ÓhÉigeartaigh, S. S. (2018). An AI Race for Strategic Advantage: Rhetoric and Risks. In *Proceedings of the 2018 AAAI/ACM Conference on AI, Ethics, and Society*, pages 36–40.

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.

Declaration, M. (2018). The Montreal Declaration for the Responsible Development of Artificial Intelligence Launched. https://www.canasean.com/the-montreal-declaration-for-the-responsible-development-of-artificial-intelligence-launched/.

Dorogovtsev, S. (2010). *Complex networks*. Oxford: Oxford University Press.

European Commission (2020). White paper on Artificial Intelligence – An European approach to excellence and trust. Technical report, European Commission.

Future of Life Institute (2015). Autonomous Weapons: An Open Letter from AI \& Robotics Researchers. Technical report, Future of Life Institute, Cambridge, MA.

Future of Life Institute (2019). Lethal Autonomous Weapons Pledge. https://futureoflife.org/lethal-autonomous-weapons-pledge/.

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. (2019). Modelling and Influencing the AI Bidding War: A Research Agenda. In *Proceedings of the AAAI/ACM conference AI, Ethics and Society*, pages 5–11.

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.

Newman, M. E. J. (2003). The structure and function of complex networks. *SIAM review*, 45(2):167–256.

Newman, M. E. J. (2004). Coauthorship networks and patterns of scientific collaboration. *Proceedings of the national academy of sciences*, 101(suppl 1):5200–5205.

Pamlin, D. and Armstrong, S. (2015). Global challenges: 12 risks that threaten human civilization. *Global Challenges Foundation, Stockholm.*

Russell, S., Hauert, S., Altman, R., and Veloso, M. (2015). Ethics of artificial intelligence. *Nature*, 521(7553):415–416.

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.

Generation of Complex Patterns using Coupled Generative Adversarial Networks

Hiroyuki Iizuka[1,2], Taiki Sasaki[3], Wataru Noguchi[1], and Masahito Yamamoto[1,2]

[1]Faculty of Information Science and Technology, Hokkaido University, Japan
[2]Center for Human Nature, Artificial Intelligence, and Neuroscience, Hokkaido University, Japan
[3]Graduate School of Information Science and Technology, Hokkaido University, Japan
iizuka@ist.hokudai.ac.jp

Abstract

This study reveals what kind of temporal and spatial patterns form when learning in an adversarial relationship between two individuals. The model was implemented by coupling generative adversarial networks, which are well-known in the field of machine learning. The obtained temporal patterns resulted in chaos with a positive Lyapunov exponent for time-series learning, whereas spatial pattern learning produced structured patterns with a higher fractal dimension, not just more complexity with a higher entropy.

Introduction

The simulation studies for communication emergence in Alife aim to reveal how communication signals are organized to make sense with other individuals (Marocco et al., 2003; Hashimoto and Ikegami, 1996; Shibuya et al., 2018). In those simulations, other individuals are always cooperative and the signals tend to converge on simple symbolic usages, such as alarm sounds used by animals. How, then, can human language and some bird songs, which form complex signal patterns, emerge?

The studies in which complex patterns are produced involve dilemmatic or competitive environments instead of simple cooperative relationships (Suzuki and Kaneko, 1994; Hashimoto and Ikegami, 1996; Iizuka and Ikegami, 2004; Moran and Pollack, 2017). In particular, Suzuki and Kaneko (1994) showed that the parameters of the logistic function evolved to the edge of chaos in generating time series in a situation where individuals want to imitate but not be imitated. We call this situation an adversarial imitation. The ability to imitate profitable and useful behaviors is beneficial. However, for those performing the behavior, being imitated loses the benefit. Brood parasitism, used by some birds, is a good example. The strategy of imitating and not-being-imitated can be evolutionarily dominant. In this study, we use deep learning methods to investigate what kind of time series or spatial patterns adversarial imitation generates and how the patterns can be structured using a simulation-based synthetic approach.

Coupled Generative Adversarial Networks

Generative Adversarial Networks (GANs) are a technique that imitates real data (e.g., images) to create artificial data that does not actually exist but is realistic (Goodfellow et al., 2014). We use GANs to model adversarial imitation.

Time series generation

We first describe the model of applying coupled GANs to time series generation, simulating a situation in which two agents engage in adversarial imitation. Figure 1 presents an overview of the simulation model. Each agent consists of a time series generator and discriminator. The generator generates its own time series, feeding back its output to the input at each step. The imitation time series are generated while inputting the opponent's time series at each step. The discriminator receives each generated time series as input and outputs whether it is its own or the opponent's. The discriminator is trained such that it can recognize the time series generated by its own generator as its own and the imitation time series as the opponent's. The teaching signals are given as to who outputs the time series. The generator is trained such that its discriminator recognizes the own time series as its own, and the opponent's discriminator is fooled into recognizing the generated imitation time series as the opponent's own time series.

The top graphs in Fig.2 are bifurcation diagrams showing the convergence points of the time series generated by the trained generators at each learning epoch. The random initialization of the generators resulted in monotonically converging dynamics; however, as the learning progressed, we observe bifurcations of the generated time series and dynamics changing from periodic to more complex trajectories. The graphs on the lower right of Fig.2 present examples of the complex trajectories. The Lyapunov exponents calculated by differentiating the generator networks show that generators produced a chaotic trajectory (see lower left of Fig.2).

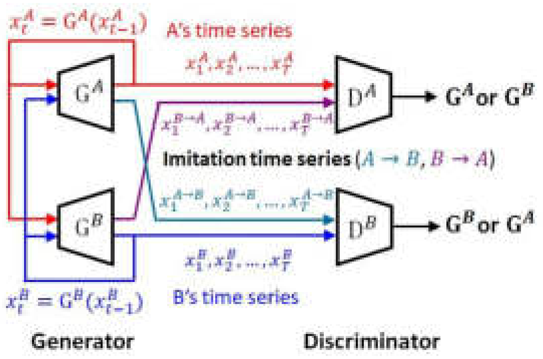

$x_t^A = G^A(x_{t-1}^A)$
A's time series
$x_1^A, x_2^A, \dots, x_T^A$
G^A
D^A
G^A or G^B
$x_1^{B\to A}, x_2^{B\to A}, \dots, x_T^{B\to A}$
Imitation time series ($A \to B$, $B \to A$)
$x_1^{A\to B}, x_2^{A\to B}, \dots, x_T^{A\to B}$
G^B
D^B
G^B or G^A
$x_1^B, x_2^B, \dots, x_T^B$
$x_t^B = G^B(x_{t-1}^B)$
B's time series
Generator
Discriminator

Figure 1: Coupled GANs for time series generation

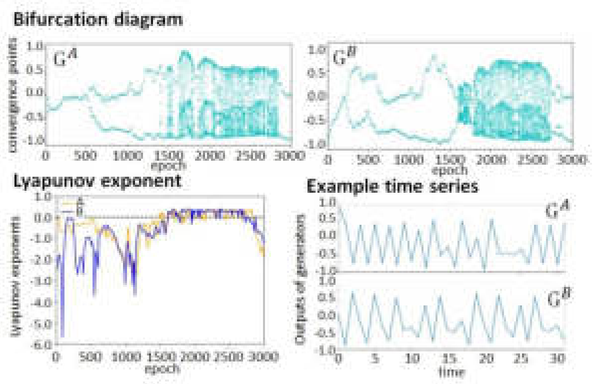


Figure 2: Learning results of time series generation

Spatial pattern generation

We then applied the Coupled GANs to generate spatial patterns. Figure 3 presents an overview of our simulation model. The spatial pattern generators were implemented using feed-forward networks with convolution layers, whereas the time series in the previous experiment was generated in a recurrent manner. The generators generated grayscale images of size 128 $\times$ 128 from random values, similar to the original GANs. These generators and discriminators were trained using adversarial imitation in the same manner as the previous experiment. To show that mutual adversarial imitation learning produced structural patterns, we compared it to the unidirectional condition, in which only one individual performed adversarial imitation learning.

The graphs on the left in Fig.4 show the patterns generated by generators $\mathrm{G^A}$ and $\mathrm{G^B}$ with eight different random z values under the mutual conditions. The generations of images by $\mathrm{G^A}$ and $\mathrm{G^B}$ are independent, but they generated similar patterns owing to the imitation effect. However, they did not maintain the same pattern. The generated patterns became equally cluttered at the beginning because all network parameters were initialized with random values. In addition,

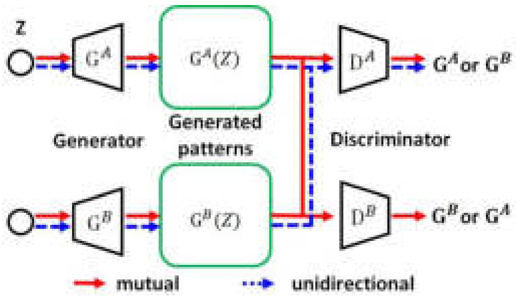


Figure 3: Coupled GANs for spatial pattern generation

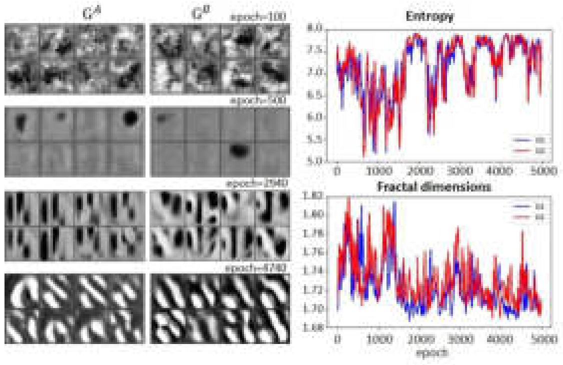


Figure 4: Generated patterns and changes of entropy and fractal dimensions

as the learning process progressed, in the mutual adversarial imitation learning, the generated patterns had a type of global pattern instead of a locally cluttered pattern. Conversely, in the unidirectional adversarial imitation learning, the generated patterns became locally detailed and cluttered (not shown).

To show the progress of complication and structuring under mutual adversarial imitation learning, entropy and fractal dimensions were computed (see the images on the right in Fig.4). In the mutual adversarial imitation learning, the entropy and fractal dimension oscillated. The entropy sometimes decreased, and the pattern became less cluttered. The fractal dimension temporarily increased. In the unidirectional adversarial imitation learning, we observed that entropy simply increased and that the fractal dimension stagnated at a relatively low value. These results indicate that mutual adversarial imitation learning generates not only complex patterns, but also formed structural patterns measurable in the fractal dimension.

Conclusion

We showed that adversarial imitation learning can increase the complexity of patterns and structure them both temporally and spatially. If only one side of the discriminators was trained, it simply produced a messy spatial pattern. This suggests that mutual learning is necessary for the formation of patterns with structure rather than mere clutter.

Acknowledgements

This work was partially supported by JSPS KAKENHI Grant Number JP20H04989.

References

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014). Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680.

Hashimoto, T. and Ikegami, T. (1996). Emergence of net-grammar in communicating agents. *BioSystems*, 38(1):1–14.

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

Marocco, D., Cangelosi, A., and Nolfi, S. (2003). The emergence of communication in evolutionary robots. *Philosophical Transactions of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences*, 361(1811):2397–2421.

Moran, N. and Pollack, J. (2017). Effects of cooperative and competitive coevolution on complexity in a linguistic prediction game. In *Proceedings of the 14th European conference on artificial life*, pages 298–205.

Shibuya, N., Iizuka, H., and Yamamoto, M. (2018). Evolution of communication through differentiation of communicative and goal-directed behaviors. *Artificial life and robotics*, 23(2):225–234.

Suzuki, J. and Kaneko, K. (1994). Imitation games. *Physica D: Nonlinear Phenomena*, 75(1-3):328–342.

Glaberish: Generalizing the Continuously-Valued Lenia Framework to Arbitrary Life-Like Cellular Automata

Q. Tyrell Davis[1] and Josh Bongard[1]

[1]University of Vermont, Burlington, VT 05405
qdavis & jbongard @uvm.edu

Abstract

Recent work with Lenia, a continuously-valued cellular automata (CA) framework, has yielded ∼100s of compelling, bioreminiscent and mobile patterns. Lenia can be viewed as a continuously-valued generalization of the Game of Life, a seminal cellular automaton developed by John Conway that exhibits complex and universal behavior based on simple birth and survival rules. Life's framework of totalistic CA based on the Moore neighborhood includes many other interesting, Life-like, CA. A simplification introduced in Lenia limits the types of Life-like CA that are expressible in Lenia to a specific subset. This work recovers the ability to easily implement any Life-like CA by splitting Lenia's growth function into genesis and persistence functions, analogous to Life's birth and survival rules. We demonstrate the capabilities of this new CA variant by implementing a puffer pattern from Life-like CA Morley/Move, and examine differences between related CA in Lenia and Glaberish frameworks: Hydrogeminium natans and s613, respectively. These CA exhibit marked differences in dynamics and character based on spatial entropy over time, and both support several persistent mobile patterns. The CA s613, implemented in the Glaberish framework, is more dynamic than the Hydrogeminium CA (and likely most Lenia-based CA) in terms of a consistently high variance in spatial entropy over time. These results suggest there may be a wide variety of interesting CA that can be implemented in the Glaberish variant of the Lenia framework, analogous to the many interesting Life-like CA outside of Conway's Life. Supporting information and resources are open-source[1].

Introduction

John Conway's Game of Life is a cellular automaton (CA, plural: cellular automata) and archetypical example of a complex and computationally universal system arising from simple rules (Gardner, 1970; Berlekamp et al., 2004). Life is a "zero-player" (or simulation) game invented by Conway during coffee breaks with help from colleagues; its invention laid the foundations for a vibrant community of hobbyists and professional researchers to study Life and related CA, with some of the most interesting and beautiful work in CA the work of non-academic artists, engineers, and tinkerers. Life consists of binary cells on a grid, with each cell's dynamics defined by the sum of neighbors and the cell's own state. The dynamics of Life depend entirely on cell states and neighborhood values and follow a simple set of rules.

[1]Links to supporting resources, including notebooks for replicating this paper's figures and additional animations are consolidated at `https://rivesunder.github.io/yuca`

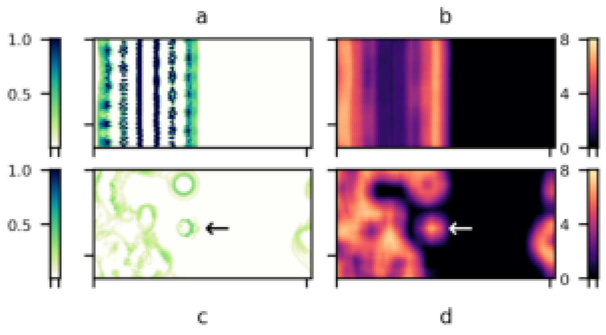


Figure 1: Hydrogeminium natans (Lenia) and s613 (Glaberish) CA after 47 time steps. Both CA started with the same 64-wide vertical strip of cells with random uniform intial values. a) Hydrogeminium grid state after 47 time steps. b) Spatial entropy map of a. c) s613 grid state after 47 time steps, with emerged "frog" glider pattern. d) Spatial entropy map of c. Arrows call out glider pattern in s613. Spatial entropy was computed with a window size of 23, entropy units are in bits.

Life is one of 262,144 possible rules in the framework of Life-like CA. Rules can be written as Bx/Sy, where x and y are any, none, or all integers from 0 through 8. Life is written as B3/S23: empty cells with 3 neighbors are **B**orn, or become 1, and active cells with 2 or 3 neighbors **S**urvive, and retain a state of 1. Life-like CA update synchronously and discretely. Because B rules and S rules are 9-bits each, and can be implemented in any combination, there are $2^9 \cdot 2^9 = 2^{18}$ possible rule combinations of Life-like CA.

In Life cells can have a state of 0 or 1, and each cell has a neighborhood consisting of the cells orthogonally and diagonally adjacent to it. With exactly 3 neighbors cell state becomes 1. With 2 neighbors, cell state remains unchanged, regardless of whether the current state is 0 or 1. All other cells take on a state of 0. Note that it is possible to write a description of Life rules without referring to the particular value of the cell state. This distinction enables facile implementation of Life in the Lenia framework, but is not a general characteristic for every Life-like CA.

Inspired by Life, CA frameworks have subsequently been expanded to larger neighborhoods (Evans, 2001; Pivato, 2007), higher dimensions (Bays, 1987; Chan, 2020), multiple channels (Chan, 2020)[2], and continuous-value states and updates (Rucker, 2003; Chan, 2019)[3] to give just a few examples of the many Life-inspired CA projects.

An early framework for continuous CA, developed by Rudy Rucker in the 1990s, was CAPOW (Rucker, 2003). Later, Stephan Rafler described the SmoothLife CA based on sharply defined inner and outer neighborhoods, with birth and survival rules defined by a pair of intervals comprised of smooth step functions[4]. Bert Chan developed the continuous Lenia CA framework with a different formulation: neighborhoods are defined by smooth convolution kernels and a single update function (called the growth function) (Chan, 2019).

Neither SmoothLife nor Lenia explicitly considered Life-like CA in general: SmoothLife was developed with a single continuous interval each for birth and survival (*i.e.* SmoothLife used "Bays space" rules (Bays, 1987)), and Lenia updates do not depend on cell state. This potentially leaves a vast volume of continuous CA with interesting dynamics that are not readily implementable in Lenia or SmoothLife.

A Life-like CA of particular interest is called Move or Morley, with rules B368/S245 (Figure 4). Morley supports a commonly occurring puffer, a mobile pattern that continuously generates persistent patterns along its trajectory. The Morley puffer is a simple example of an infinite growth pattern, of particular interest as an indicator of complexity in CA. Morley has multiple B and S conditions, none of which overlap and most of which are not contiguous, making it an ideal Life-like CA for exploring the limitations of previous continuous CA frameworks.

In the next section we demonstrate Lenia does not have a simple implementation of Life-like CA Morley. This limitation is exemplified in Figures 4 and 5. Choosing to view this as a problem, we incorporate conditional updates based on cell state, and name the resulting CA framework *Glaberish*, a nod to the Latin etymology of Lenia and a reference to the role of Life-like CA beyond Conway's Life in studying complex systems. Glaberish takes its root from another Latin word for smooth[5], and Glaberish is to Lenia as Life-like CA are to Conway's Life.

[2]See also `https://softologyblog.wordpress.com/2018/03/09` and `https://github.com/slackermanz/vulkanautomata`

[3]See also `https://github.com/rudyrucker/capow`, `https://www.rudyrucker.com/capow`, and `https://arxiv.org/abs/1111.1567`

[4]Rafler, S. (2011). Generalization of Conway's "Game of Life" to a continuous domain - SmoothLife. pre-print on ArXiv `https://arxiv.org/abs/1111.1567`

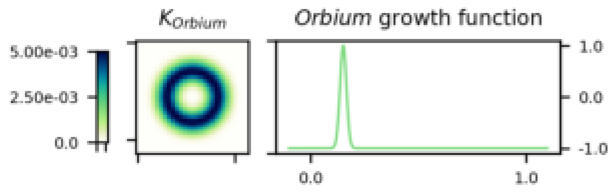


Figure 2: Lenia neighborhood kernel and growth function for Orbium CA. A neighborhood kernel and a growth function gives rise to complex behavior in Lenia. The Gaussian kernel defined by ($\mu = 0.5, \sigma = 0.15$) and Gaussian growth function defined by ($\mu = 0.15, \sigma = 0.015$), shown here, supports the exemplary *Orbium* glider pattern.

In Results several techniques from the literature are used to assess CA implemented in the Lenia framework and in Glaberish. A spatial entropy metric, similar to that in (Wuensche, 1999), is shown applied to CA grid states in Lenia and Glaberish in Figure 1, with an emergent glider pattern in the Glaberish CA (s613). This metric is further considered in the context of previous work in the Discussion section. Speculation about the potential value of increasingly complicated systems, *i.e.* continuous CA, forms the basis for possible future research directions discussed in our Conclusions.

Glaberish is Lenia with conditional updates

Like Life, Lenia defines CA dynamics with a neighborhood and an update function. Neighborhood values are the result of 2D convolution with kernel K, which become input to growth function G. CA dynamics in the Lenia system are defined as

$$A^{t+dt} = \psi(A^{t}) + dt \cdot G(K * A^{t})) \qquad (1)$$

where ψ is a squashing or clipping function that limits values to between 0.0 and 1.0, and A^t is the CA grid at time t.

The kernel K and growth function G can take various forms, but a typical Lenia CA uses Gaussian functions of the form

$$f(x) = exp\left(-\left(\frac{(x-\mu)}{2\sigma}\right)^2\right) \qquad (2)$$

[5]Lenia and Glaberish are based on the Latin root words *lenis* and *glaber*, respectively.

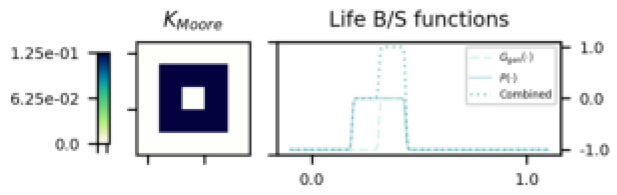


Figure 3: Conway's Game of Life implemented in the Lenia framework. Kernel K is a Moore neighborhood, and a stepwise growth function corresponds to B3/S23.

Figure 2 visualizes K and the growth function for a Lenia CA, Orbium. The neighborhood kernel is a ($\mu = 0.5, \sigma = 0.15$) Gaussian with a 2D grid of radial distance from center as input x. K is normalized to have a sum of 1.0, and the output of convolution with K is input for the growth function. In the same CA, the growth function $G(\cdot)$ is a Gaussian with ($\mu = 0.15, \sigma = 0.015$). Note that for use as a growth function, $f(x)$ is stretched to yield values from -1 to 1, *i.e.* $G(x) = 2 \cdot f(x) - 1$. The Orbium CA supports an iconic glider pattern of the same name, described in (Chan, 2019).

Figure 3a shows how Conway's Life can be implemented in the Lenia framework. The B and S rules overlap, so if we sum the corresponding continuous intervals we get a two-step staircase function; the default value is -1, cells go to 0, values in the survival interval yield a value of 0, no change, and values in the birth interval yield an update of +1. Growth occurs only where B and S rules overlap.

Lenia can only readily implement Life-like CA where values in the B interval are a subset of those in the S interval, to enable growth and avoid survival outside of survival intervals. B368/S245, aka Morley (Figure 4), is an example of a CA with no overlap between B and S rules and is not readily implementable in the Lenia framework. Combining the B/S rules as in Figure 3 results in a growth function that never returns values greater than 0 (Figure 4a), yielding a CA that is not capable of growth.

To extend Lenia to a full generalization of Life-like CA, we add conditional updates, splitting the growth function into genesis and persistence functions that depend on the cell state. Equation 3 defines the Glaberish update.

$$A^{t+dt} = \psi\left(A^t + dt[(1 - A^t)G_{gen}(N) + A^t P(N)]\right) \quad (3)$$

In Equation 3 we use N to denote the result of neighborhood convolution $K * A^t$. The growth function $G(\cdot)$ is replaced in Equation 3 by genesis function $G_{gen}(\cdot)$ and persistence function $P(\cdot)$. Conditional update dynamics rescue the ability to implement B368/S245, and a timelapse of the common Morley puffer pattern is shown in Figure 5.

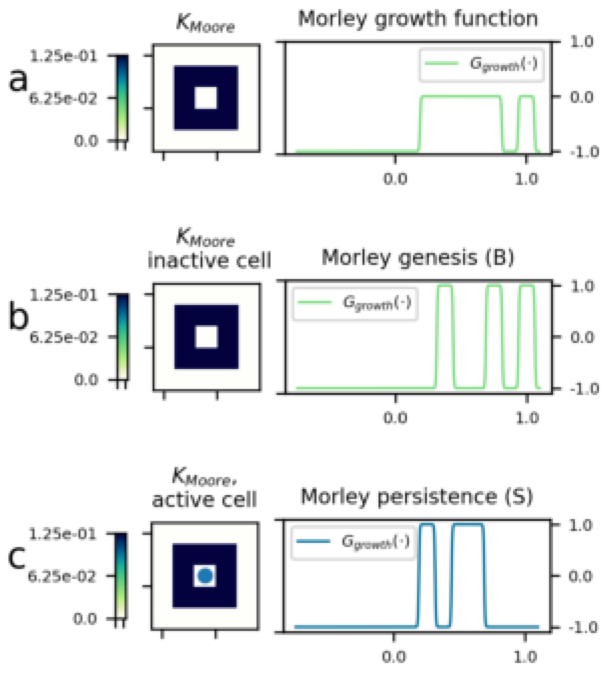


Figure 4: Continuous implementations of the Life-like Morley/Move CA (B368/S245). a) Implemented in Lenia with updates defined by a single growth function. b) and c) Implemented in Glaberish with separate genesis and persistence functions.

Results

We compare the Hydrogeminium CA from Lenia (Chan, 2019) to an evolved CA rule set under the Glaberish framework. We refer to the Glaberish CA as "s613" for the random seed used to evolve it, selecting for poor performance of neural networks trained to predict whether all cell states fall to 0 after a set number of time steps, described in (Davis and Bongard, 2022). These CA share a neighborhood kernel, and the persistence function from s613 resembles a slightly shifted version of the growth function from Hydrogeminium natans (Figure 4). The most notable difference between the two sets of rules, the s613 genesis function (a Gaussian with $\mu = 0.063$ and $\sigma = 0.0088$) makes for markedly different dynamics between the two CA.

Hydrogeminium is defined by a (shared with s613) neighborhood kernel with three rings, the weighted sum of three Gaussians acting on the radial distance from center in a 2D grid, with parameters (μ, σ) = $[(0.0938, 0.033), (0.2814, 0.0330), (0.469, 0.033)]$ and weights $[0.5, 1.0, 0.667]$. The growth function has parameters $(\mu, \sigma) = (0.26, 0.036)$. s613 shares the neighborhood function (as well as the same step size $dt = 0.1$), but has genesis and persistence functions with parameters $(\mu, \sigma) =$

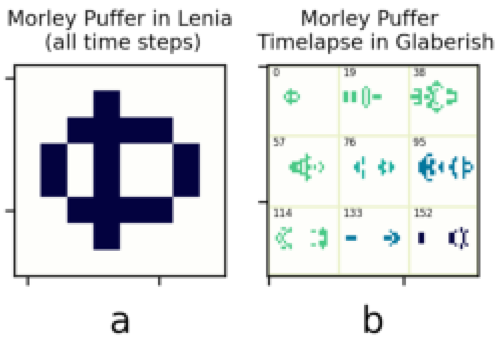

Morley Puffer in Lenia (all time steps)
Morley Puffer Timelapse in Glaberish
a
b

Figure 5: A common puffer in Life-like CA Morley/Move (B368/S245). a) The puffer pattern is static in Lenia, under the growth function in Figure 4a. b) Implemented in Glaberish, the pattern exhibits expected behavior: moving across the grid while leaving a trail of oscillating patterns.

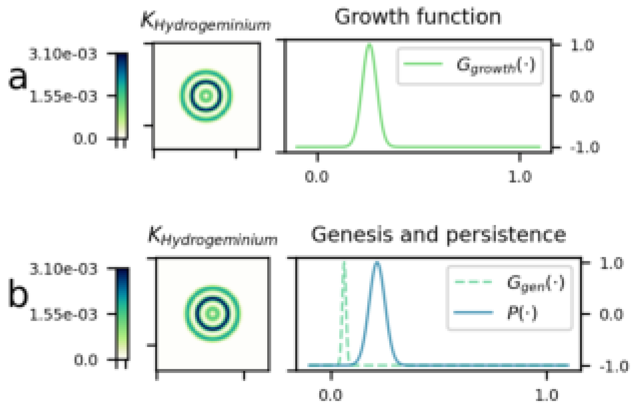


Figure 6: Rule visualization in Lenia and Glaberish. a) Hydrogeminium, a CA based on the Lenia framework. b) s613, a Glaberish CA. These use the same neighborhood kernel but differ in their update functions. Most notably s613 splits updates into genesis and persistence dependent on current cell state. Despite minor differences in the update functions and marked differences in CA dynamics, both systems readily support mobile patterns.

$(0.0621, 0.0088)$ and $(0.2151, 0.0369)$, respectively.

Wolfram (1983) suggested a subjective classification scheme for complexity in 1D CA, later applied to 2D CA in (Packard and Wolfram, 1985), and often referenced to as a general rubric for CA. Under the Wolfram classes, briefly, Class I CA cells quickly reach a uniform state (usually all 0), Class II CA typically produce an equilibrium grid of static and/or oscillating patterns, Class III CA produce chaotic dynamics with aperiodic patterns, and Class IV CA exhibit long-lived, complex, and localized patterns. These categories remain subjective, but the sedentary patterns produced by Hydrogeminium might fall under Class II, while the open-ended dynamics of s613 could be categorized as Class III or class IV. Class III CA are suggested in Wolfram (1983) to tend toward a consistent density value, but in Figure 8 we see substantial variation in average cell value (normalized to the mean) in s613 over 8 runs, much more than for Hydrogeminium.

Eppstein (2010) introduced a set of heuristic requirements for predicting complexity and universality in CA: mortality and fertility. CA are *mortal* if a pattern can be found that disappears after several time steps, and *fertile* if a pattern exists that displays the capacity for infinite growth (estimated by finding patterns that escape an initial bounding box). Unlike Wolfram's CA classes and their application to 2D CA (Wolfram, 1983; Packard and Wolfram, 1985) these metrics are not subjective and capture more of the Life-like CA that support universal computation.

Although fertility and mortality were not developed with continuous CA in mind, Hydrogeminium and s613 meet both criteria. Random uniform conditions typically escape an initial bounding box, as demonstrated in Figure 7 (bounding box, in gray, is twice the width and height of the initialization area). A vanishing pattern for either CA is trivial: a single active pixel with a value smaller than any update function zero-crossing must vanish. Starting from a random uniform initial state (range 0.0 to 1.0), Hydrogeminium typically settles into a mostly static state resembling a Turing pattern (Turing, 1952). CA s613 remains dynamic, continuously remodeling local structures.

An important motivation for studying artificial life is to explore how to recognize living systems vastly different than found in Earth ecosystems. One approach is to take an abstract, information theoretic approach to defining the activities of life. In broad strokes, we can look for systems that generate local departure from thermal equilibrium (Popescu, 2011) and generate statistically unlikely structures. Entropy-based measures find common application in analyzing life and computation alike. Indeed many of the key activities of biological life including replication, transcription, and translation have been analyzed as computation with respect to their thermodynamic efficiency (Kempes et al., 2017). Cellular automata, important models of both computation and artificial life, have been been subject to several entropy-based metrics including input entropy (Wuensche, 1999), local entropy (Helvik et al., 2004), conditional entropy (Peña and Sayama, 2021), and transfer entropy (Lizier et al., 2008).

We consider spatial entropy of CA grids, calculating the image entropy under a sliding window with the same dimensions as the convolution kernel used to compute neighborhood values. Given the neighborhood kernel window size,

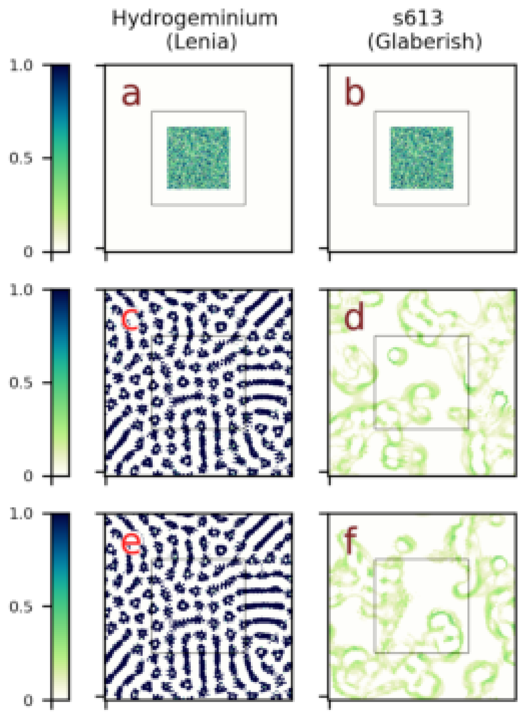


Figure 7: Grid state progression from random uniform initial states. a) Hydrogeminium at step 0. c) Hydrogeminium at step 512. c) Hydrogeminium at step 1024. b) s613 at step 0 (same as in 7a). d) s613 at step 512. f) s613 at step 1024.

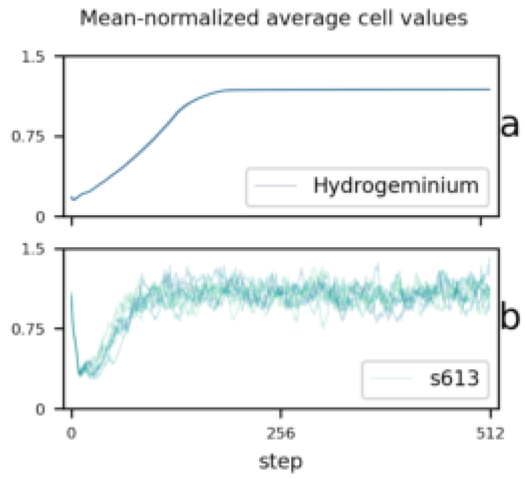


Figure 8: Mean-normalized mean cell value over time. b) After 256 time steps, the normalized average cell value for s613 varies from its overall mean value by a standard deviation of 0.0947, while Hydrogeminium (a) has a standard deviation of 0.00204. As discussed in (Wolfram, 1983), class III CA are characterized by chaos but tend to an equilibrium cell density, which we don't see in the case of s613.

our spatial entropy measure is conceptually similar to the input entropy in (Wuensche, 1999), which considers the discrete states of cells defining 1D CA rules of varying input size. The 2D spatial entropy used here considers the cell values (discretized to 8 bits for computational tractability) in the neighborhood kernel window. In the window around each cell in the CA grid we compute entropy as for a 2D 8-bit image.

$$H = -\sum_{s=0}^{S} P(s) \log_2(P(s)) \tag{4}$$

where H is the entropy of the sub-grid and $P(s)$ is the empirical proportion of cells in the sub-grid in state s. Computed at each cell location (or pixel), Equation 4 yields a spatial entropy map. In Figure 9 spatial entropy is visualized for a random uniform initial state (confined to a starting box), and after several hundred time steps of change under Hydrogeminium and s613 rules, starting from the same random uniform initialization. After 1024 steps, Hydrogeminium has very little spatial entropy variance while s613 has higher average entropy and much higher variance. Over 8 runs with bounded random uniform initial cell states, Hydrogeminium had final average spatial entropy of 1.18 bits $\pm$ 0.065 (standard deviation), while s613 had final average spatial entropy of 3.95 bits $\pm$ 1.25.

In addition to variation across the CA grid at a given time step, s613 spatial entropy varies substantially over time, especially as compared to Hydrogeminium in Figure 10. The upper and lower bounds in Figure 10, representing standard deviation from the mean, remain consistent and wide over time in s613. Hydrogeminium approaches a static mean value and vanishingly small standard deviation of spatial entropy after about 200 time steps.

Mortality and fertility heuristics, subjective categorization, and entropy-based metrics are all attempts to measure and/or predict complexity and universality in CA. They do not fully substitute for finding coherent, self-organizing, traveling patterns known as gliders (or, often, particles in 1D CA). They may serve as effective tools for finding gliders, however, especially when information theoretic metrics are used as filters to highlight the presence of gliders/particles (Lizier et al., 2008; Shalizi et al., 2006; Wuensche, 1999; Helvik et al., 2004). In 2D CA gliders often are readily apparent from observation, as in the discovery of the classic reflex glider in Life (Berlekamp et al., 2004). In this work we present a few examples of gliders in Hydrogeminium and s613, found by evolving synthesis patterns encoded as compositional pattern producing networks (Stanley, 2007) in (Davis and Bongard, 2022). These patterns were selected for a combination of center-of-mass displacement and con-

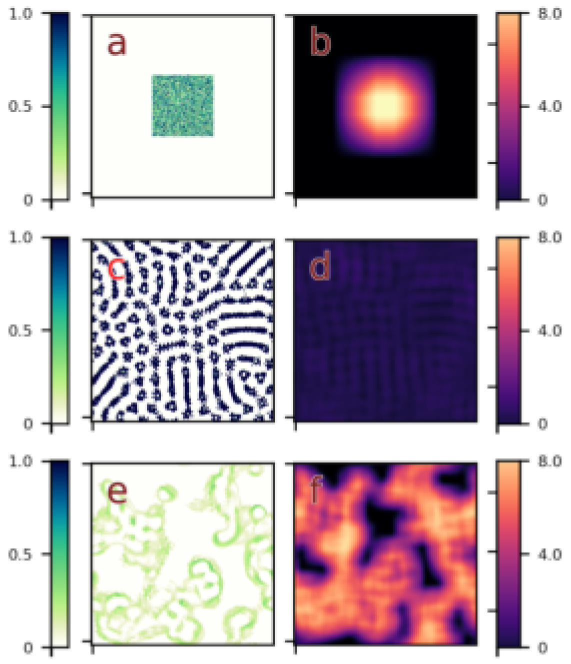


Figure 9: Spatial entropy maps of step 512 of CA simulation, starting from a bounded random uniform initial grid. a) Random uniform initial grid state. b) Spatial entropy for a. c) Hydrogeminium grid state after 1024 steps and d) spatial entropy of c. e) Grid state of s613 after 1024 steps and f) corresponding spatial entropy map.

sistent average cell value (a proxy for homeostasis).

Traveling glider patterns from each CA are shown along with trajectories in Figure 11, and include a slow-moving "wrinkled cucumber" in Hydrogeminium and a hopping "frog" pattern in s613. Both CA support similar broad gliders that initially have a straight traveling behavior, but begin to wobble after several hundred time steps and often eventually become unstable.

Figure 1 shows the Figure 11d pattern emerging from s613 dynamics, and corresponding spatial entropy, with the same grid state simulated for an equal number of steps in Hydrogeminium for comparison. This pattern emerges comparatively often from random uniform initial states in s613, frequently created and destroyed in collisions with other active cells.

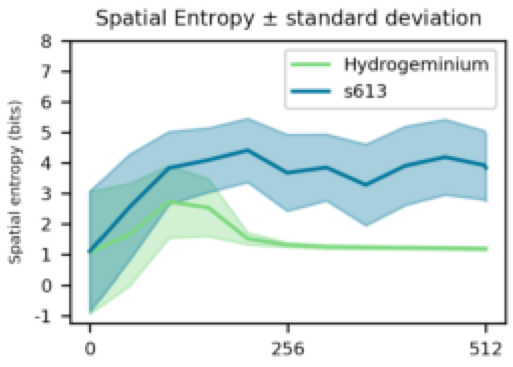


Figure 10: Spatial entropy over time. Values represent 11 snapshots taken every 25 time steps in Hydrogeminium and s613 CA simulations. Starting grid states were identical and sample from a random uniform distribution. Error fill is ± standard deviation.

Discussion

By extending Conway's Life to continuous values and time steps, Lenia introduced a combinatorial increase in possible configurations. By focusing on Life, Lenia also introduced a significant simplification over Life-like CA, opting for a single growth function instead of mirroring the cell value-dependent birth and survival rules of Life-like CA. We showed that this simplification limits Lenia's ability to implement arbitrary Life-like CA: there is no straightforward implementation of Life-like CA with birth rules that are not a subset of its survival rules[6].

Aside from the lack of straightforward implementations of many of the 262,144 possible Life-like CA in Lenia, any significant limitation as a model for artificial life as a result of the growth function simplification in Lenia is not readily apparent: work with the framework has generated a large taxonomy of bioreminiscent patterns (Chan, 2019) and Lenia has continued to be the substrate for increasingly automated exploration of bioreminiscent patterns (Reinke et al., 2020; Davis and Bongard, 2022) and modifications to the framework itself (Chan, 2020; Kawaguchi et al., 2021). Nonetheless, there is a vast number of Life-like CA with non-overlapping and/or non-contiguous B/S rules, including the Morley/Move CA we focused on in this article and the B356/S23 CA found in Peña and Sayama (2021) to score the highest on a complexity metric based on conditional entropy[7], and the increased rule-space enabled with Glaberish likely contains many interesting CA.

[6]But note that Lenia can readily implement Life, which in turn can simulate any other Life-like CA. See for example Brice Due's OTCA metapixel `https://otcametapixel.blogspot.com/`

[7]Like Morley, B356/S23 has a frequently-occurring puffer pattern, a reflex puffer with period 72

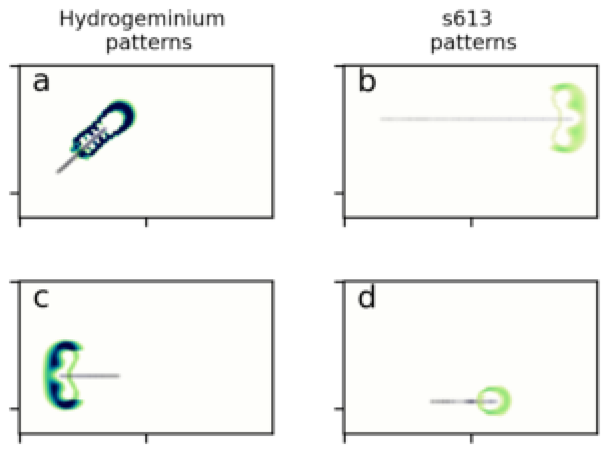


Figure 11: Self-organizing, traveling patterns in Hydrogeminium and s613. a) Large, slow-moving, and eponymous pattern in Hydrogeminum natans. b) Fast-moving glider in s613. This pattern begins to wobble afer some time steps and eventually becomes unstable. It has a similar pattern in c) glider in Hydrogeminium natans. This pattern starts to wobble after a few hundred time steps, and sometime becomes unstable over long time periods. d) "Frog" pattern in s613. This pattern moves in a straight, hopping motion. Trajectories correspond to 64 time steps for all patterns, except a) which shows 128 time steps.

In comparing related CA in Lenia and Glaberish, we observed that Hydrogeminium tends to eventually produce mostly sedentary Turing patterns, an equilibrium characteristic of class II CA in Wolfram's classification scheme. The Glaberish CA s613 continuously remodels local structures and remains difficult to predict, characteristic of Class III or IV (chaotic or complex) CA under Wolfram's classification (Wolfram, 1983). The markedly different dynamics of the two related CA make predicting future states in Hydrogeminum significantly easier than in s613. It is not clear how much the framework influences the CA dynamics in comparison to the way each rule set was developed: s613 was evolved to be difficult for convolutional neural networks to predict whether a given pattern would disappear after a number of time steps (Davis and Bongard, 2022), whereas Hydrogeminium was developed with a combination of manual and semi-automated evolution (Chan, 2019). Also, not all Lenia CA share the characteristic end-point of a static Turing pattern[8].

We also considered variance in spatial entropy as a life-like characteristic for assessing CA. One hallmark of life seems to be unusual or improbable structure, *e.g.* living eukaryotic cells maintain an improbable distinction between the external environment and their internal milieu, as well as distinct environments within organelles that quickly return to equilibrium when life stops. In terms of spatial entropy, s613 is more varied over both space and time than Hydrogeminium (Figures 9 and 10). Spatial entropy considered here is a nearly analogous metric to the input entropy described in (Wuensche, 1999). Wuensche described a characteristic signature of complex CA in entropy variance, which we also see in s613, across space as well as time. Other information theoretic measures of complexity have been applied to CA, including local entropy (Helvik et al., 2004), conditional entropy (Peña and Sayama, 2021), transfer entropy (Lizier et al., 2008), and local sensitivity and statistical complexity (Shalizi et al., 2006). While not considered here, the project repository includes tools for computing conditional entropy on continuous CA.

We also demonstrated that both Hydrogeminium and s613 support gliders. Gliders form a foundational basis of computation in CA, readily performing the essential functions of information transfer by their ability to travel, and information modification in the consequences of their collisions (Berlekamp et al., 2004; Lizier et al., 2008, 2010). Gliders are essential components of engineered computing devices in CA, such as Turing machines (Rendell, 2011) or Life metacells capable of simulating arbitrary Life-like CA[9].

Information theoretic filters have been shown to be effective for identifying gliders and other coherent structures in CA (Wuensche, 1999; Shalizi et al., 2006; Helvik et al., 2004; Lizier et al., 2008). Our gliders, in contrast, were obtained from synthesis pattern evolution, selected for center-of-mass displacement and mean cell value homeostasis (Davis and Bongard, 2022).

Conclusions

In this paper we discussed a potentially important limitation of the Lenia framework: updates are applied without regard to current cell values, which limits facile implementation of Life-like CA to those with birth rules wholly contained as a subset of the survival rules. Life, with its B3/S23 rule set, can be readily implemented in Lenia, but not Morley or other CA with non-overlapping B/S rules. We demonstrated that by splitting the update function into conditional genesis and persistence functions (analogous to B and S rules), the ability to implement Life-like CA with distinct B/S rules can be recovered, as shown for the Morley puffer pattern.

In a direct comparison between related CA implemented in the Lenia and Glaberish frameworks, the Glaberish CA is more active and unpredictable. Both CA meet the mortality and fertility criteria predicting universality from (Eppstein, 2010), but s613 exhibits greater variance in entropy across space and time.

[8]Examples of more dynamic Lenia CA include *Astrium scintillans* and two variants of *Pentafolium incarceratus*.

[9]See, for example, Brice Due's OTCA metacell `https://otcametapixel.blogspot.com/`

More importantly, both CA support gliders. Gliders are important components of computational ability, and drivers of life-like characteristics, in CA because they transmit information and can perform computation in the results of their collisions. Substantial efforts have been made to develop formal classification or heuristics that are predictive of CA that are computationally universal and compelling, but there seems to be no comprehensive substitute for finding gliders.

As noted in (Packard and Wolfram, 1985), increasing degrees of freedom in CA can increase the tendency to chaos, and as noted in (Chan, 2020) expanding CA to greater numbers of channels and dimensions makes persistent self-organizing patterns rare, but potentially more interesting. Continuously-valued CA implementations certainly fall under "additional degrees of freedom" as compared to their discrete counterparts. In addition, Glaberish at least doubles the number of functions used to calculate updates, but if applied to the Lenia's expanded universe extension (Chan, 2020), where multiple channels may interact in different ways, Glaberish may lead to an exponential increase in rule functions. Determining whether the trade-off of a more complicated implementation is worth the greater expressiveness of the Glaberish formulation, specifically whether the framework is meaningfully more capable for a given task, remains the subject of future work. While continuous dynamics and strong localization of structure that we demonstrated for the Glaberish automaton s613 are promising, our comparisons were limited to one pair of exemplary (and related) CA from the Lenia and Glaberish frameworks and do not claim superiority of one CA framework over the other.

Similar arguments for parsimony could be made against Lenia and other extended CA frameworks with additional degrees of freedom, compared to simple CA like Life. Even Life may seem too complicated if we also have simpler systems available, like the Turing-complete rule 110 elementary CA (Cook et al., 2004).

A strong argument in favor of more complicated CA is the potential for agents to exist fully embodied in simulated environments that follow the same physics as the agents themselves. Continuous CA, complicated as they may be in comparison to their simple antecessors, may fill the gap between real-world robots and simulated agents. In typical simulated learning environments, *e.g.* in reinforcement learning research, there is a sharp distinction between the mechanics of control policies and that of the simulated environment. Recent work, described online[10] demonstrated a hint of the potential for embodied agents under consistent physics in continuous CA. In a special version of Lenia's expanded universe, the authors used gradient descent methods to optimize glider-supporting CA to improve the gliders' ability to survive interaction with obstacles in an immutable channel. Several CA in that work each give rise to a particular glider pattern that by all appearances skirts obstacles contained in the immutable layer.

[10]https://developmentalsystems.org/sensorimotor-lenia/

Spatial entropy maps with window size 23 are visualized in Figure 1 for a comparison of Hydrogeminium and s613, with s613 exhibiting a glider pattern emerging from CA dynamics. An aim of future work is to discover more complex patterns with compartmentalization of internal structures and a clear distinction in internal/external entropy values (or another information theoretic complexity measure), making for an intriguing analog to biological cells.

Glaberish expands the search space of possible CA in Lenia in the same way as Life-like CA expanded the possible CA rules beyond Life itself, providing a more expansive substrate for developing agency and autopoiesis in continuous CA. While aesthetics and human pareidolia play a role in the attractiveness of continuous CA patterns, they have the potential to offer increasingly life-like systems for studying life-like computation.

Funding

This work was supported by the National Science Foundation under the Emerging Frontiers in Research and Innovation (EFRI) program (EFMA-1830870).

References

Bays, C. (1987). Candidates for the Game of Life in three dimensions. *Complex Systems*, 1:373–400.

Berlekamp, E. R., Conway, J. H., and Guy, R. K. (2004). *Winning Ways for Your Mathematical Plays Volume 4. Second Edition*. A K Peters, Wellesly, Massachusetts.

Chan, B. W.-C. (2019). Lenia - biology of artificial life. *Complex Systems*, 28(3):251–286.

Chan, B. W.-C. (2020). Lenia and expanded universe. *The 2020 Conference on Artificial Life*, pages 221–229.

Cook, M. et al. (2004). Universality in elementary cellular automata. *Complex systems*, 15(1):1–40.

Davis, Q. T. and Bongard, J. C. (2022). Selecting continuous Life-like cellular automata for halting unpredictability: Evolving for abiogenesis. In Fieldsend, J., editor, *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, New York, NY, USA. Association for Computing Machinery.

Eppstein, D. (2010). Growth and decay in life-like cellular automata. In *Game of Life cellular automata*, pages 71–97. Springer.

Evans, K. M. (2001). Larger than Life: Digital creatures in a family of two-dimensional cellular automata. *Discrete Mathematics and Theoretical Computer Science AA (DM-CCG)*, pages 177–192.

Gardner, M. (1970). Mathematical games: The fantastic combinations of John Conway's new solitaire game "Life". *Scientific American*, 223:120–123.

Helvik, T., Lindgren, K., and Nordahl, M. G. (2004). Local information in one-dimensional cellular automata. In Sloot, P., Chopard, B., and Hoekstra, A., editors, *Cellular Automata: 6th International Conference on Cellular Automata for Research and Industry, ACRI 2004, Amsterdam, The Netherlands, October 25-28, 2004. Proceedings*, volume 3305, page 121. Springer.

Kawaguchi, T., Suzuki, R., Arita, T., and Chan, B. (2021). Introducing asymptotics to the state-updating rule in Lenia. In Čejková, J., Holler, S., Soros, L., and Witkowski, O., editors, *ALIFE 2021: The 2021 Conference on Artificial Life*. 91.

Kempes, C. P., Wolpert, D., Cohen, Z., and Pérez-Mercader, J. (2017). The thermodynamic efficiency of computations made in cells across the range of life. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 375(2109):20160343.

Lizier, J. T., Prokopenko, M., and Zomaya, A. Y. (2008). Local information transfer as a spatiotemporal filter for complex systems. *Physical Review E*, 77(2):026110.

Lizier, J. T., Prokopenko, M., and Zomaya, A. Y. (2010). Information modification and particle collisions in distributed computation. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 20(3):037109.

Packard, N. H. and Wolfram, S. (1985). Two-dimensional cellular automata. *Journal of Statistical Physics*, 38:901–946.

Peña, E. and Sayama, H. (2021). Life worth mentioning: Complexity in Life-like cellular automata. *Artificial Life*, 27(2):105–112.

Pivato, M. (2007). RealLife: the continuum limit of Larger than Life cellular automata. *Theoretical Computer Science*, 372(1):46–68.

Popescu, G. (2011). *Quantitative Phase Imaging of Cells and Tissues*, pages 287–288. McGraw-Hill biophotonics. McGraw-Hill Education.

Reinke, C., Etcheverry, M., and Oudeyer, P.-Y. (2020). Intrinsically motivated discovery of diverse patterns in self-organizing systems. In *International Conference on Learning Representations (ICLR)*.

Rendell, P. W. (2011). A universal Turing machine in Conway's Game of Life. *2011 International Conference on High Performance Computing & Simulation*, pages 764–772.

Rucker, R. (2003). *Continuous-Valued Cellular Automata in Two Dimensions*, chapter 12. Oxford University Press.

Shalizi, C. R., Haslinger, R., Rouquier, J.-B., Klinkner, K. L., and Moore, C. (2006). Automatic filters for the detection of coherent structure in spatiotemporal systems. *Physical Review E*, 73(3):036104.

Stanley, K. O. (2007). Compositional pattern producing networks: A novel abstraction of development. *Genetic Programming and Evolvable Machines*, 8:131–162.

Turing, A. M. (1952). The chemical basis of morphogenesis. *Philosophical Transactions of the Royal Society of London. Series B, Biological Sciences*, 237:37 – 72.

Wolfram, S. (1983). Universality and complexity in cellular automata. *Physica D: Nonlinear Phenomena*, 10:1–35.

Wuensche, A. (1999). Classifying cellular automata automatically: Finding gliders, filtering, and relating space-time patterns, attractor basins, and the Z parameter. *Complexity*, 4(3):47–66.

Ethics of Artificial Life: The Moral Status of Life as It Could Be

Olaf Witkowski[1,2], Eric Schwitzgebel[3]

[1]Cross Labs, Cross Compass Ltd., Japan
[2]College of Arts and Sciences, University of Tokyo, Japan
[3]University of California, Riverside, United States
olaf@crosslabs.org

Abstract

The field of Artificial Life studies the nature of the living state, by modeling and synthesizing living systems. Such systems, under certain conditions, may come to deserve moral consideration similar to that of non-human vertebrates or even human beings. The fact that these systems are non-human and evolve in a potentially radically different substrate should not be seen as an insurmountable obstacle to their potentially having rights equivalent to non-human vertebrates or even human beings, if they are sufficiently sophisticated in other respects. Nor should the fact that they owe their existence to us be seen as reducing their status as targets of moral concern. On the contrary, creators of artificial life may have special obligations to their creations, resembling those of an owner to their pet or a parent to their child. For a field that aims to create artificial lifeforms with increasing levels of sophistication, it is crucial to consider the possible implications of our activities under an ethical perspective, and assess the moral obligations for which we should be prepared. If artificial life is "larger than life", then the ethics of artificial beings should be "larger than human ethics".

Introduction

Under what conditions does a system deserve moral consideration intrinsically, or for its own sake, as opposed to extrinsically or derivatively? While human beings are ordinarily regarded as having "full moral status" or the highest level of moral considerability (Jaworska and Tannenbaum, 2018), other types of entities are sometimes regarded as having intrinsic moral considerability, though often to a lesser extent, such as nonhuman primates (Zimmer, 2016), other animals (Singer, 1975; Regan, 1997; Cohen and Regan, 2001), and even rivers (Zimmer, 2016). Institutional Animal Care and Use Committees regulate the treatment of all vertebrates. Artificial living systems, which include software simulations, robots, biochemical systems, ecosystems, and a wide variety of hybrids, may also under some conditions deserve some moral consideration. In spite of differing from humans, future artificial life might possess features that warrant giving it intrinsic moral consideration, whether superior, inferior, or of a different type than that of human beings or non-human vertebrates.

In addition to being non-human, possibly non-biochemical, and built from radically different blocks on different physical substrates, artificial lifeforms may be designed and engineered by humans, giving us at least partial control and thus arguably responsibility for their well being, if they are capable of well being. Because of our potential control and responsibility, artificial life forms created by us might be due additional moral obligations, resembling the obligations of an owner to a pet or a parent to a child.

Research in artificial life aims to understand the fundamental mechanisms of life by creating and studying artificial lifeforms with increasing levels of sophistication from the bottom up. The field ought to seriously consider possible implications of its activities under an ethical viewpoint, to assess the moral obligations for which society should be prepared, enlarging its perspective to include new ethical research discoveries. If artificial life is "life as it could be" (quoting Chris Langton (Langton, 1998)) and "larger than biological life" (quoting Takashi Ikegami (Witkowski et al., 2020)), then the ethics of artificial life is "ethics as it could be" and "larger than human ethics".

In this paper, we attempt to shed some light on the moral status of artificial life, and propose ways to tackle a difficult problem under transdisciplinary perspectives. We first approach the topic from the human perspective, then progressively extending to non-human entities. We then approach artificial systems to identify key ethical parameters before discussing possible criteria, stakes, and challenges looking into the future.

From human to non-human rights

Humans are often recognized as having rights that belong to all individuals simply because they are human beings. Some theorists focus on inalienable rights, a set of human rights that are fundamental, are not awarded nor can be surrendered or taken away by any human power, embodying central values such as fairness, dignity, equality, and respect, reflected in many documents including the United Nations Universal Declaration of Human Rights (Assembly et al.,

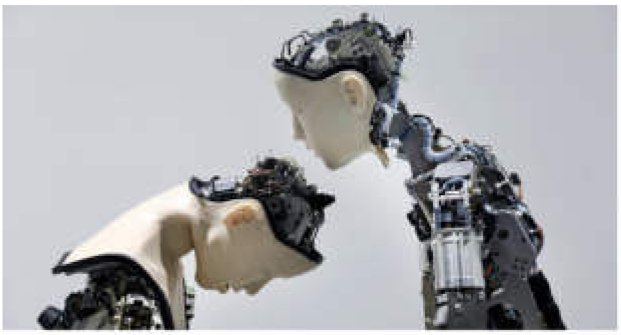


Figure 1: These Alter robots, designed for the purpose of exploring what it means to be "life-like". This proximity to humans raises the question of what the moral consideration such entities might deserve. Development by Itsuki Doi, Kohei Ogawa, Takashi Ikegami, and Hiroshi Ishiguro (2016). *Still image from the short video Soul Shift. ©2018 Justine Emard*

1948). However, "rights" talk does not need to be construed in terms of lists of inalienable rights. Virtually all current ethical perspectives recognize that human beings deserve a very high degree of "moral considerability" either simply in virtue of being human or in virtue of sets of properties or social relations that humans typically possess (Jaworska and Tannenbaum, 2018). It is unethical to kill, enslave, or torture human beings absent extremely compelling overriding considerations.

Although there exists a global agreement that humans deserve basic rights or moral consideration, other entities are far from having reached a comparable status. The main category of candidates to such rights are animals, ranging from non-human primates to animals biologically and behaviorally remote from humans. Defenders of animal rights hold that sentient animals have moral worth that is independent of their utility for humans, and that their most basic interests (life, liberty, and freedom from torture) should be given substantial consideration (Singer, 1975; Ryder, 1989; Wolff, 2012; McDonald, 2012; Korsgaard, 2018; Gruen, 2021). While views vary on exactly how much moral consideration is due to non-human animals, both ethics researchers and the general public tend to agree that some non-human animals deserve substantial protections. Many jurisdictions, for example, award prison sentences for the abuse of dogs. The United Kingdom has recently extended its Animal Welfare law to include some nonvertebrate species, particularly cephalopod molluscs and decapod crustaceans, after a report documenting the scientific consensus that the latter types of animals are sentient in the sense that they can and do feel pain and pleasure (Birch et al., 2021).

Rights or moral considerability may be considered for artificial entities as well, such as robots or AI software. Widespread agreement can be found among scholars that some artificial entities could potentially warrant moral consideration in the future (Gunkel, 2018; Harris and Anthis, 2021), some of them at least to the same degree as human beings (Schwitzgebel and Garza, 2015). One argument for this is based on the potential similarity between such AI and human beings, in all aspects that may matter, including psychological features such as consciousness, sociality, freedom, creativity, or irreplaceability. These considerations have led to further reflections about the design of policies to ensure cautious engineering that ensures that AI systems have self-respect, the freedom to explore other values, and avoid unhealthy forms of artificial altruism (Schwitzgebel and Garza, 2020).

Of the (non-)uniqueness of human life

Humans have long thought of themselves as unique, and societies have designed stories to explain why they are poised at the center of the universe (Hawking and Mlodinow, 2008). At least since the time of Galileo, evidence has been available that we are but ordinary inhabitants of the universe, living on an earth much smaller than the sun. However, the idea remains that there is something unique about being a human, often centered on two main concepts: human intelligence and human consciousness.

Research on human intelligence is fascinating and important, but we might turn out not to be as intelligent as we think. How intelligent we appear to be depends largely on the criteria for intelligence we use in studying the question. The usual approach – which makes sense from the perspective of humans, since ultimately all science is conducted by humans – is to compare the nature and capacities of human intelligence with other animal species. In that case we appear highly intelligent (Martınez-Miranda and Aldea, 2005). However, if one views intelligence in terms of physical computation (Tegmark, 2017), there are certainly ways to build computers that would be less limited than humans are in physical computing capacity (Kahle, 1979), able to process memories faster (Simon, 1955; Tegmark, 2017), capable of large-scale parallelization (Rogers and Monsell, 1995; Rubinstein et al., 2001), and able to manage memories more efficiently (Wingfield and Byrnes, 1981).

Even our degree of consciousness may not be so unique (Boly et al., 2013; Shevlin, 2021b), and some speak of artificial consciousness (Basl, 2013) or artificial sentience (Ziesche and Yampolskiy, 2019). Few nowadays agree with René Descartes that thought or consciousness is a uniquely human attribute, and nonhuman animals merely cleverly designed automatons with a toolkit of preprogrammed behaviors, each triggered by certain environmental stimuli (Chittka and Wilson, 2019). However, some still defend the idea that consciousness and higher cognitive functions are closely linked and that it's unclear whether even relatively cognitively sophisticated non-human animals have conscious experiences (Dennett, 1978; Carruthers, 2003, 2019; Rosenthal, 1993; Dennett, 2008; Papineau, 2003). However, a large part of the field of consciousness research rejects this perspective and views consciousness as a property or ensemble of mechanisms which may potentially exist in non-human beings (Birch, 2020; Shevlin, 2021b). Even if other entities differ considerably from humans in their cognitive architecture, for example having internal representations more map-like than conceptual, a different set of memory mechanisms, or even radically different types of computation (Hoffman, 2014), they might have properties sufficient for conscious experience, such as high degrees of information integration (Oizumi et al., 2014) or a cognitive "global workspace" (Dehaene, 2014).

Humans may appear to have unique properties in the animal kingdom, but the uniqueness of particular properties has been challenged again and again over the course of the history of scientific research, pointing rather at a set of evolutionary processes which made certain properties emerge and evolve, often in parallel. Most behavioral properties and cognitive mechanisms typically thought of as uniquely human, turned out to be found in other animals as well, such as culture (Kawai, 1965) and combinatorial communication (Scott-Phillips et al., 2014).

If human beings have no unique capacities that ground their high moral status, they might also not in principle be uniquely deserving of the highest moral status. A natural candidate to study is AI, for its behavior is growing more sophisticated every year. Beyond AI, one might also look at a larger range of systems, including a diverse set of intelligences, metabolisms, and functional mechanisms, which may exist over various physical substrates.

Artificial intelligence

If the field of artificial intelligence (AI) is defined as the simulation of human intelligence processes by engineered machines, it is probably still in its infancy. We might regard some AI systems as similar to children who are learning to understand causality in the physical world (Gopnik, 2017). Computers become increasingly proficient at a wide range of tasks, from simple counting and arithmetic to complex classifications such as face recognition. However, besides a set of exceptional cherry-picked outliers, the state of the art in AI is, in the words of Yoshua Bengio: "not anywhere close today to the level of intelligence of a two-year-old child" (interviewed by Eliza Strickland, on December 10, 2019). Current chatbots butcher basic sentences of common language; AI decision making fails at what we consider basic common sense; robots cannot yet balance themselves properly.

These failings reflect Moravec's Paradox, named after Hans Moravec who wrote: "It is comparatively easy to make computers exhibit adult level performance on intelligence tests or playing checkers, and difficult or impossible to give them the skills of a one-year-old when it comes to perception and mobility" (Moravec, 1988). We need to recognize that various machines are more or less optimal at various tasks (Williams Korteling, 2018). The difficulty of performing a task is by no means an intrinsic measure of task complexity, nor is an agent's performance on a specific task a measure of its general level of skill or intelligence.

Nevertheless, some successful algorithms and models do reach the level of performance of some animals, and if we allow ourselves to focus on narrow examples of what humans consider as complex tasks, progress has been extremely impressive over the last decade. Ever since IBM Deep Blue prevailed against world chess champion Garry Kasparov in a series of chess matches back in 1997, it has become clear that artificial intelligence is increasingly able to explore and react with seeming-intelligence to its environment. Some futurists envision robots with human-level intelligence, virtual simulations of humans, cyborgs, advanced brain-machine interfaces, and other emerging technologies that would blur the line between humans and machines. The science and technology of intelligence in the future may make it hard to distinguish AI from humans. Artificial life systems appear to be a good candidate for a category of systems that might, if technology evolves in a certain direction, arguably deserve moral consideration. On the other hand, as AI systems become more powerful, the line separating them from artificial living systems might become blurrier, to the point that this classification may no longer be relevant.

Narrow AI is typically distinguished from General AI (or AGI, for Artificial General Intelligence). The former is defined as the production of systems displaying intelligence regarding specific, highly constrained tasks, like playing chess, facial recognition, autonomous navigation, or locomotion (Goertzel, 2014). AGI, on the other hand, is often thought to involve the achievement of at least human-level general intelligence. However, human intelligence may not be as general nor high level as it is widely thought to be. Nor is it as unique as it is claimed to be. The widespread conception of intelligence has an intrinsic anthropocentric component, as it is only natural to view the human mind as a reference. We tend to use it as a basis for reasoning about other, less familiar phenomena of intelligence, such as other forms of biological and artificial intelligence (Coley and Tanner, 2012).

Technological beings are also likely to be the only future space travelers in a further future (Schneider, 2017), and it has been argued that we might fuse with future artificial technologies, as we have already to some extent by our interaction with them, by transmitting to them large parts of our knowledge and even values, increasingly relying upon them. Considering the long-term future of potential space travel, it's likely that our biological particles won't be able to efficiently travel in space, and thus it's likely that only our biological information in a larger sense could be distributed beyond the Solar System, either in different forms of physical embodiment or simply as information transmitted in electromagnetic waves towards new colonies. The potential ubiquity of AI invites questions of their potential future moral consideration (Dameski, 2018), although they are far from being the only category of systems at stake.

Artificial life, beyond AI

Artificial life (ALife) is a broad field of study about the synthesis and simulation of living systems. The field exists at the intersection of many other fields, including biology, chemistry, computer science, art, philosophy, engineering, astrophysics, and more. As its name indicates, the purpose of ALife is to understand the fundamental mechanisms and properties of "life as it could be", instead of "life as we know it". Its scope includes natural life with its processes and evolution, but also instances in computational models, robotics, biochemistry, and any other forms of life, discovered or designed, in the past or the future.

The ALife approach must deal with a large set of fundamental problems including the lack of a formal definition of life over a very diverse distribution of instances and radically different substrates (Bedau et al., 2000). However, since its inception, the field has proposed numerous metrics to identify conditions which may be more suitable for life than others, which include for example measures of complexity, computational capabilities, or open-endedness (Frans et al., 2021; Stepney, 2021; Stanley, 2019).

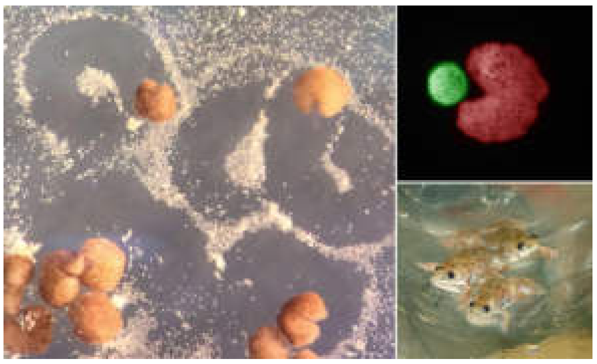


Figure 2: An example of hybrid entities is the Xenobot, an in vitro self-replicating biological robot. The version 3.0 follows the original Xenobots reported in 2020 as the first living robots, and Xenobots 2.0, capable of self-propelling using cilia and maintaining memories. Synthesized from frog cells, these computer-designed living machines are able to navigate aqueous environments in different ways, forage for single cells, heal after damage, and show emergent group behaviors (Blackiston et al., 2021). *Image source: Daily Mail ©2021 Douglas Blackiston & Sam Kriegman.*

A particular topic of interest in ALife, although it is not only limited to artificial systems, is the one of hybrid systems. Hybridity may involve an external designer, a mixture of mechanical, electrical, chemical, or biological components. There exists some tension in defining such hybrid machines, as they tend to escape the simple dichotomy between machines versus living organisms. A recent example of such hybrid robot (or hybrot) can be found in Xenobots (Blackiston et al., 2021) (see Figure 2) are made of frog skin and heart cells but are designed via a genetic algorithm. Such hybrid systems may be considered life forms, but some have been reticent to call them organisms because of the absence of certain properties such as reproduction (Coghlan and Leins, 2020). This hybridity brings about novel challenges, such as the apparent oxymoron of a designed yet autonomous agent (Siqueiros, 2021). Our limited understanding of potential hybrid systems invites concerns about our capacity to understand what their moral status might be. From the angle of environmental ethics, Holy-Luczaj and Blok (2021) argue that it may primarily be the ability to serve other beings by performing certain of their functions, intrinsic to their identity, that qualify a being for moral considerability.

Although a complete review of the current state of research on this topic is challenging due to the exponential growth of a diverse range of research, Harris and Anthis (2021) is a valuable coverage and synthesis. They show how, despite some scholars dismissing the question of moral considerability as premature or frivolous for artificial beings, an increasing number believe the topic is worth addressing urgently, even proposing the development and formalization of a field of "AI welfare science" (Ziesche and Yampolskiy, 2018).

Three approaches to moral status

Although rights are an important and effective way of conceptualizing moral status, a more fundamental starting point is to consider whether an entity's interests morally matter to some degree for the entity's own sake. More precisely, we focus on moral considerability. An entity may be said to have moral considerability if its suffering is morally bad, on account of this animal itself and regardless of the consequences for other beings. Specific terminologies depend on the type of framework or approach adopted (Jaworska and Tannenbaum, 2018). There are generally three approaches to this question: consequentialist approaches, especially utilitarianism, which focuses on the capacity for pleasure or suffering, deontological approaches, which focus on the intrinsic value of an entity, and eudaimonistic approaches, which focus on the flourishing of an entity.

The utilitarian approach views moral considerability in terms of a calculation of each agent's interests to determine which action maximizes a utility function, based on as many factors as necessary, including for example the intensity, duration, and probability of an entity's pleasure or pain (Singer, 1993; DeGrazia, 2008; Mill, 2001; Bentham, 1988). Alternatively, others favor an individual-rights-based or deontological approach, arising from the traditions of

Kant and social contract theory (Kant, 1785). This non-utilitarian approach views moral status such that there are reasons to act for the sake of the entity or its interest, reasons which are prior to, and may clash with, what the calculation of the overall best consequences would dictate (Regan, 2004). A third approach, rooted in the Aristotelian tradition, emphasizes human, or alternatively non-human, flourishing, including acting virtuously as a type of human flourishing (Nussbaum, 2009).

One immediate concern might come to mind for utilitarian or eudaimonistic approaches: What if AI systems were capable of superhuman levels of pleasure and suffering, or flourishing? Would we then owe them more moral consideration than we owe to our fellow human beings? We don't rule out this possibility, but some theorists might find it unappealing or unintuitive. Conversely, approaches that focus on individual rights might struggle if future artificial systems can merge and divide at will, or if the boundaries of individuality become vague and permeable.

What criteria for moral considerability?

The main parameters at play in determining whether a being is deserving of rights usually belong to the following list – presented to illustrate what an arbitrary cut of plausible criteria for moral status may resemble, and by no means to be regarded as exhaustive or final. Nevertheless, these may be considered as a starting reflective structure to think of moral rights for artificial systems.

Embodiment

The physical embodiment refers to the biology, dynamical properties, or architecture of an entity. The nature of the living state is not well defined in the literature, or at least it possesses many opposing definitions. Nevertheless, many would rely on the embodiment as a criterion to attribute moral status to an entity. Biological or physical criteria may include some characterization of mechanisms, metabolism, behavior, and other biochemical parameters. Kiršienė et al. (2021) propose autonomy and embodiment as important criteria, but argue that while there may be future conditions to justify AI personhood, doing so now appears to be technically premature and is likely to be inappropriate. In our view, similarity of physical embodiment should probably not be the determining factor for moral considerability. The cognition or sentience of an entity may not depend upon its medium of embodiment either (Doctor et al., 2022). Most current theories of the grounds of moral status focus on psychological and social properties as more important to moral status than an entity's particular form of embodiment, though of course certain forms of embodiment might make certain forms of cognition easier, more useful, or more likely.

Consciousness

An entity is *conscious* if and only if "there is something it's like" to be that entity (Nagel, 1974), or if the entity has a stream of experiences, such as sensory or affective experiences. Having consciousness might be necessary for moral considerability, or it might be sufficient, or both. Unfortunately, there is little consensus about what entities are conscious and how consciousness arises. Metaphysical options include dualism, according to which consciousness requires non-physical substances or properties, materialism, according to which everything in the world is wholly physical, and several types of alternative views or compromise views. Even among materialist views, options range from panpsychism or near-panpsychism, according to which consciousness is ubiquitous or at least very widespread, to views on which consciousness is a rare and delicate achievement only in the most sophisticated organisms. On liberal views of consciousness, artificial systems might already be conscious. On conservative views, artificial systems might never be conscious. If consciousness is required for moral considerability, then on conservative views, artificial systems might never merit moral consideration. Liberal views of consciousness, when combined with the view that consciousness is important to moral considerability, fit more neatly with the view that artificial systems might soon warrant moral consideration. However, the most liberal views might involve denying that the mere existence of consciousness is sufficient for a high level of moral considerability, unless one wants to commit to the view that very simple systems already have high moral status.

Sentience in a narrow sense refers to the capacity of entities to experience positive and negative affect, such as sensations of pleasure and pain. Sentience in a broad sense might also include features of the mind such as creativity, intelligence, sapience, self-awareness, and intentionality, which may not be needed for sentience in its narrower sense. Arguably, all sentient entities are also conscious the sense of having "something it's like" to be them (Nagel, 1974), although there might be room for a view in which wholly nonconscious entities also have affective systems. Utilitarian views typically emphasize specifically sentience rather than consciousness in general as the basis of moral considerability. Existing work on, for example, whether decapods or certain species of vertebrate fish can feel pain is often regarded as central to the question of whether they have moral considerability. This is pointed out by Elwood (2021), who point out behavioral responses to potentially painful events different from simple reflexes. This illustrates how in practice, the characterization of consciousness or sentience may connect closely with cognitive or behavioral considerations.

Due to the extreme difficulty of reaching consensus on a universal detector of conscious experience or sentience, it might be very difficult to assess whether an artificial entity is conscious or sentient. If its moral status turns on consciousness or sentience, we might be left with considerable moral uncertainty. To avoid situations in which we might be grossly mistreating entities that have, unbeknownst to us, the full moral status of human beings, we recommend considering the “Design Policy of the Excluded Middle” (Schwitzgebel and Garza, 2015). According to this policy, we should avoid creating AIs for which it is unclear whether they would or not deserve moral consideration similar to that of human beings.

Cognition and behavior

Cognition refers to any mental process consisting in gaining knowledge and comprehension, including reasoning, problem solving, remembering, judgment, planning, abstract thinking, complex idea comprehension, and learning from experience (Gottfredson, 2004). A typical example is the capacity for mathematical reasoning, or the ability to carry out a complex task. Although assessing the cognitive structures and capacities of a system is sometimes difficult, the task is considerably more straightforward than settling questions of consciousness. Partly for this reason, some theorists might prefer to think of moral considerability in cognitive terms: Any entity with the right cognitive capacities deserves moral consideration, whether than entity is human or artificial. Which are the right cognitive capacities may prove to be a difficult theoretical question (Shevlin, 2021a,b). Presumably the capacity to add a list of numbers is not enough for the highest level of moral status, since artificial systems already have this capacity. Conversely, infants and the severely cognitively disabled are typically regarded as having full moral status equivalent to ordinary adult human beings, even if their cognitive capacities are very different from ours.

Outward behavior is another potentially simple and tractable tool for assessing moral status. If a system behaves similarly to a human being, perhaps it deserves similar moral status (Danaher, 2021). Questions that arise are: Similar in what respects? If the system is architecturally simple, so that it plausibly lacks consciousness and cognition like ours, would we really want to treat it as having full moral status? What about systems that have limited outward behavior but whom we ordinarily regard as deserving of full moral status, such as people with locked-in syndrome?

Social attribution

Social structures arise from their members and the larger public, as entities that exist independently of of any particular individual. Such social entities come with their own sets of relationships and moral, legal, and physical features. If morality is partly grounded in intersubjective agreement, then belonging to the right social structure or being attributed moral status within a group might be sufficient for moral considerability. David Gunkel and Mark Coeckelbergh argue that moral status is a socially constructed result of negotiation among groups, and that the participating groups might at some point come to include artificial entities (Coeckelbergh, 2012; Gunkel, 2018).

A perhaps esoteric example of attribution may be found in an entity created mentally or spiritually, called *tulpa* (Mikles and Laycock, 2015; Veissière, 2016). It refers to a type of willed imaginary friend which practitioners consider to be sentient and possessing certain autonomous abilities. Such entities may be deserving of derived moral considerability, even if they have no intrinsic moral considerability, just as a child’s favorite stuffed animal might be treated with respect due to the child’s concern for it. This would imply including extrinsic properties as grounds for moral considerability, in opposition to strictly intrinsic grounds (Liao, 2020).

Relatedly, even if we suppose that some entities, such as artificial systems and non-human animals, have no intrinsic moral considerability, it might be morally wrong to harm them because harming them either expresses a vice in the person who does the harming or nurtures habits and attitudes that may be harmful in the long run through affecting how one treats other people (Kant, 1785; Darling, 2021).

We note there may be biases towards evaluating moral choices by artificial beings that resemble humans as less moral compared to the same moral choices made by either humans or clearly nonhuman robots (Laakasuo et al., 2021). This moral uncanny valley effect might have similar or even further implications for the moral considerability of artificial life, as it is designed to be as life-like as possible, possibly in different or more open-ended ways compared to AI which arguably suffers from risks of local optima due to the focus of solely imitating human behavior.

Group entities and the extended mind

Social entities themselves, at any scale, may come to be attributed a moral status. By the virtue of existing autonomously from its composing agents or others, including any organization such as social groups, legal structures, cultural entities, ecosystems, and more. Many of them, such as corporations or states, certainly are recognized to possess legal rights. Some have even argued that social groups, such as the United States might be literally conscious (Schwitzgebel, 2015; Lerner, 2021).

Others have argued that our minds literally extend into the world when we rely heavily on the world for our cognitive processes (Clark and Chalmers, 1998; Clark et al., 2008; Chalmers, 2019), which may involve conscious processes as well (Vold, 2020). As we become highly dependent on external devices, harming those devices might literally be harming our own minds, so that taking someone’s smartphone or stealing a blind person’s cane is better conceived of as assault resulting in cognitive damage rather than merely theft (Vold, 2018). Human-robot dyadic interaction may be a related case (Zahavi, 2019). Artificial life approaches have been proposed to study morality with the help of modeling populations of artificial agents (Sullins, 2005; Witkowski and Ikegami, 2016).

Discussion

One may wonder whether there would be practical applications for the problem of determination of moral status. One near-term issue is this: sometime in the near future some people with liberal ideas about what sorts of systems deserve moral consideration will likely come to think that some artificial systems have moral status and interests that need to be protected. They might rush to save a favorite robot in a fire, for example, risking their lives for it; or they might object to the mistreatment of a service delivery bot, thinking that abuse of such a bot deserves criminal penalty similar to the abuse of a dog. This is likely to occur in the near future regardless of whether such entities actually do deserve such moral consideration. The issue needs to be considered in advance, so that we can address this likely social problem in an informed way.

Coevolution, omnipresent in theory of artificial life, may be a promising way to think of the problem. A compelling example may be Neanderthals, who were assimilated by early modern human populations. There was an evolutionary, cultural, and technological gap between humans and Neanderthals. Although Homo Sapiens is considered to have "won" the survival game, due to interbreeding, Neanderthal genes still exist within human DNA, and the same might be said about elements of their culture and technology. Other examples of coevolutionary events might be found in pets, which are part of human history, or viruses which coevolved with bacteria. All such cases illustrate further our previous point on hybrid forms of life, where entities are not only to be considered on one layer of organization, but rather on multiple levels, vertically as well as laterally. For example, humans exist at the scale of their DNA, but are also part of larger ecosystems, cultures, or technological timelines, which may be considered as entities of their own right.

The topics treated in this article are contentious and require a transdisciplinary approach, touching multiple fields in science, engineering, and the humanities. Both within and between disciplines, clashing perspectives are likely. The aim of this paper is not to formulate a final response to the questions posed but rather to invite wide-ranging interdisciplinary conversation. We remark that moral consideration cannot be based merely on the results of scientific research, although it can be informed by them. Singer (1990), for example, argues that equality of consideration is a prescription, not an assertion of fact such as intelligence, physical strength, or moral capacity. We also note the existence of potential risks that, although not discussed in detail in this paper, should nevertheless be considered seriously when creating artificial living entities, especially when they are capable of moral judgment themselves (Cave et al., 2018).

Sadly, even the fight for universal human rights is far from won. It goes without saying that any discussion about the rights of machine or nonhuman life should by no means slow us down in our hard fight for the protection and support of human rights. We believe that the type of broad vision about the bases of moral considerability at work in thinking about the moral status of non-human animals and artificial systems is one that supports and aligns with, rather than competes with, a broad vision of human rights.

The principal ambition of this article is to shed some light on the relevance of the perspective of artificial life on the question of moral consideration of entities. Parallel to AI, where advances are happening at a much faster pace than the design of policies would be able to reach, the engineering of living systems is also moving up a gear, pressing society to include them in the current design processes. Artificial living systems may soon become a concrete aspect of our daily lives. When that time comes, research groundwork should already be in place to inform ethical policies, not only for life, but also for life as it could be.

Acknowledgements

OW would like to acknowledge the support of Templeton World Charity Foundation (TWCF) grant No. 0470. The opinions expressed in this publication are those of the authors and do not necessarily reflect the views of the TWCF.

References

Assembly, U. G. et al. (1948). Universal declaration of human rights. *UN General Assembly*, 302(2):14–25.

Basl, J. (2013). The ethics of creating artificial consciousness. *APA Newsletter on Philosophy and Computers*, 13(1):23–29.

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.

Bentham, J. (1988). The principles of morals and legislation. 1789. *Amherst, NY: Prometheus Books*, 25.

Birch, J. (2020). The search for invertebrate consciousness. *Noûs*.

Birch, J., Burn, C., Schnell, A., Browning, H., and Crump, A. (2021). Review of the evidence of sentience in cephalopod molluscs and decapod crustaceans.

Blackiston, D., Lederer, E., Kriegman, S., Garnier, S., Bongard, J., and Levin, M. (2021). A cellular platform for the development of synthetic living machines. *Science Robotics*, 6(52):eabf1571.

Boly, M., Seth, A. K., Wilke, M., Ingmundson, P., Baars, B., Laureys, S., Edelman, D., and Tsuchiya, N. (2013). Consciousness in humans and non-human animals: recent advances and future directions. *Frontiers in psychology*, 4:625.

Carruthers, P. (2003). *Phenomenal consciousness: A naturalistic theory*. Cambridge University Press.

Carruthers, P. (2019). *Human and animal minds: The consciousness questions laid to rest*. Oxford University Press.

Cave, S., Nyrup, R., Vold, K., and Weller, A. (2018). Motivations and risks of machine ethics. *Proceedings of the IEEE*, 107(3):562–574.

Chalmers, D. (2019). Extended cognition and extended consciousness. *Andy Clark and his critics*, pages 9–20.

Chittka, L. and Wilson, C. (2019). Expanding consciousness. *Amer Sci*, 107:364–369.

Clark, A. and Chalmers, D. (1998). The extended mind. *analysis*, 58(1):7–19.

Clark, A. et al. (2008). *Supersizing the mind: Embodiment, action, and cognitive extension*. OUP USA.

Coeckelbergh, M. (2012). *Growing moral relations: Critique of moral status ascription*. Palgrave Macmillan.

Coghlan, S. and Leins, K. (2020). "living robots": Ethical questions about xenobots. *The American Journal of Bioethics*, 20(5):W1–W3.

Cohen, C. and Regan, T. (2001). *The animal rights debate*. Rowman & Littlefield.

Coley, J. D. and Tanner, K. D. (2012). Common origins of diverse misconceptions: Cognitive principles and the development of biology thinking. *CBE—Life Sciences Education*, 11(3):209–215.

Dameski, A. (2018). A comprehensive ethical framework for ai entities: Foundations. In *International Conference on Artificial General Intelligence*, pages 42–51. Springer.

Danaher, J. (2021). What matters for moral status: Behavioral or cognitive equivalence? *Cambridge Quarterly of Healthcare Ethics*, 30(3):472–478.

Darling, K. (2021). *The new breed: what our history with animals reveals about our future with robots*. Henry Holt and Company.

DeGrazia, D. (2008). Moral status as a matter of degree? *The Southern Journal of Philosophy*, 46(2):181–198.

Dehaene, S. (2014). Consciousness and the brain: How the brain codes our thoughts. *NY: Viking*.

Dennett, D. C. (1978). Toward a cognitive theory of consciousness.

Dennett, D. C. (2008). *Kinds of minds: Toward an understanding of consciousness*. Basic Books.

Doctor, T., Witkowski, O., Solomonova, E., Duane, B., and Levin, M. (2022). Biology, buddhism, and ai: Care as the driver of intelligence. *mindRxiv*.

Elwood, R. W. (2021). Potential pain in fish and decapods: Similar experimental approaches and similar results. *Frontiers in Veterinary Science*, 8:369.

Frans, K., Soros, L., and Witkowski, O. (2021). Questions for the open-ended evolution community: Reflections from the 2021 cross labs innovation science workshop. *Artificial Life*, 25(1):4.

Goertzel, B. (2014). Artificial general intelligence: concept, state of the art, and future prospects. *Journal of Artificial General Intelligence*, 5(1):1.

Gopnik, A. (2017). An ai that knows the world like children do. *Scientific Amer. Mind*, 28(5):21–28.

Gottfredson, L. S. (2004). Life, death, and intelligence. *Journal of Cognitive Education and Psychology*, 4(1):23–46.

Gruen, L. (2021). *Ethics and animals: An introduction*. Cambridge University Press.

Gunkel, D. J. (2018). *Robot rights*. mit Press.

Harris, J. and Anthis, J. R. (2021). The moral consideration of artificial entities: a literature review. *Science and engineering ethics*, 27(4):1–95.

Hawking, S. and Mlodinow, L. (2008). The grand design.

Hoffman, D. D. (2014). The origin of time in conscious agents. *Cosmology*, 18:494–520.

Holy-Luczaj, M. and Blok, V. (2021). Hybrids and the boundaries of moral considerability or revisiting the idea of non-instrumental value. *Philosophy & Technology*, 34(2):223–242.

Jaworska, A. and Tannenbaum, J. (2018). The grounds of moral status. *Stanford Encyclopedia of Philosophy*.

Kahle, W. (1979). Band 3: nervensysteme und sinnesorgane. *Taschenatlas de anatomie. Stutttgart*.

Kant, I. (1785). Groundwork of the metaphysics of morals.(1785).

Kawai, M. (1965). Newly-acquired pre-cultural behavior of the natural troop of japanese monkeys on koshima islet. *Primates*, 6(1):1–30.

Kiršienė, J., Gruodytė, E., and Amilevičius, D. (2021). From computerised thing to digital being: mission (im) possible? *AI & SOCIETY*, 36(2):547–560.

Korsgaard, C. M. (2018). *Fellow creatures: Our obligations to the other animals*. Oxford University Press.

Laakasuo, M., Palomäki, J., and Köbis, N. (2021). Moral uncanny valley: A robot's appearance moderates how its decisions are judged. *International Journal of Social Robotics*, 13(7):1679–1688.

Langton, C. G. (1998). A new definition of artificial life. *URL: http://scifunam. fisica. unam. mx/mir/langton. pdf*.

Lerner, A. B. (2021). What's it like to be a state? an argument for state consciousness. *International Theory*, 13(2):260–286.

Liao, S. M. (2020). The moral status and rights of artificial intelligence. *Ethics of Artificial Intelligence*, page 480.

Martınez-Miranda, J. and Aldea, A. (2005). Emotions in human and artificial intelligence. *Computers in Human Behavior*, 21(2):323–341.

McDonald, G. (2012). Teaching critical & analytical thinking in high school biology? *The American Biology Teacher*, 74(3):178–181.

Mikles, N. L. and Laycock, J. P. (2015). Tracking the tulpa: exploring the "tibetan" origins of a contemporary paranormal idea. *Nova Religio: The Journal of Alternative and Emergent Religions*, 19(1):87–97.

Mill, J. S. (2001). Utilitarianism, ed. george sher.

Moravec, H. (1988). *Mind children: The future of robot and human intelligence*. Harvard University Press.

Nagel, T. (1974). What is it like to be a bat. *Readings in philosophy of psychology*, 1:159–168.

Nussbaum, M. C. (2009). Creating capabilities: The human development approach and its implementation. *Hypatia*, 24(3):211–215.

Oizumi, M., Albantakis, L., and Tononi, G. (2014). From the phenomenology to the mechanisms of consciousness: integrated information theory 3.0. *PLoS computational biology*, 10(5):e1003588.

Papineau, D. (2003). Could there be a science of consciousness? *Philosophical Issues*, 13:205–220.

Regan, T. (1997). Beyond prejudice: The moral significance of human and nonhuman animals, by evelyn pluhar. *Journal of Agricultural and Environmental Ethics*, 10(1):79–82.

Regan, T. (2004). *The case for animal rights*. Univ of California Press.

Rogers, R. D. and Monsell, S. (1995). Costs of a predictible switch between simple cognitive tasks. *Journal of experimental psychology: General*, 124(2):207.

Rosenthal, D. (1993). Thinking that one thinks. *Language and Thought*, pages 259–287.

Rubinstein, J. S., Meyer, D. E., and Evans, J. E. (2001). Executive control of cognitive processes in task switching. *Journal of experimental psychology: human perception and performance*, 27(4):763.

Ryder, R. D. (1989). Animal revolution. *Changing attitudes towards speciesism*, pages 101–104.

Schneider, S. (2017). Superintelligent ai and the postbiological cosmos approach. *What is life*, pages 178–198.

Schwitzgebel, E. (2015). If materialism is true, the united states is probably conscious. *Philosophical Studies*, 172(7):1697–1721.

Schwitzgebel, E. and Garza, M. (2015). A defense of the rights of artificial intelligences. *Midwest Studies in Philosophy*, 39:98–119.

Schwitzgebel, E. and Garza, M. (2020). Designing ai with rights, consciousness, self-respect, and freedom.

Scott-Phillips, T. C., Gurney, J., Ivens, A., Diggle, S. P., and Popat, R. (2014). Combinatorial communication in bacteria: Implications for the origins of linguistic generativity. *PLoS One*, 9(4):e95929.

Shevlin, H. (2021a). How could we know when a robot was a moral patient? *Cambridge Quarterly of Healthcare Ethics*, 30(3):459–471.

Shevlin, H. (2021b). Non-human consciousness and the specificity problem: A modest theoretical proposal. *Mind & Language*, 36(2):297–314.

Simon, H. (1955). A behavioural and organizational choice. *Quarterly Journal of Economics*, 69:99–118.

Singer, P. (1975). Animal liberation. *NY: Harper Collins Publishers Inc*, 1990:2002.

Singer, P. (1990). The significance of animal suffering. *Behavioral and Brain Sciences*, 13(1):9–12.

Singer, P. (1993). Taking life: humans. *Practical ethics*, 2:175–217.

Siqueiros, J. M. (2021). You, robot: Empathy in a hybrid world. In *ALIFE 2021: The 2021 Conference on Artificial Life*. MIT Press.

Stanley, K. O. (2019). Why open-endedness matters. *Artificial life*, 25(3):232–235.

Stepney, S. (2021). Modelling and measuring open-endedness. *Artificial Life*, 25(1):9.

Sullins, J. P. (2005). Ethics and artificial life: From modeling to moral agents. *Ethics and Information technology*, 7(3):139–148.

Tegmark, M. (2017). *Life 3.0: Being human in the age of artificial intelligence*. Vintage.

Veissière, S. (2016). Varieties of tulpa experiences: The hypnotic nature of human sociality, personhood, and interphenomenality. *Hypnosis and meditation: Towards an integrative science of conscious planes*, pages 55–76.

Vold, K. (2018). Are 'you' just inside your skin or is your smartphone part of you? https://aeon.co/ideas/are-you-just-inside-your-skin-or-is-your-smartphone-part-of-you.

Vold, K. (2020). Can consciousness extend? *philosophical topics*, 48(1):243–264.

Williams Korteling, N. (2018). The materiality of research: creating a community of writing practice in the classroom. *Impact of Social Sciences Blog*.

Wingfield, A. and Byrnes, D. (1981). The psychology of human. *Memory. New York: Aca-demic Press*.

Witkowski, O. and Ikegami, T. (2016). Swarm ethics: Evolution of cooperation in a multi-agent foraging model. *Proceedings of the First International Symposium on Swarm Behavior and Bio- Inspired Robotics*.

Witkowski, O., Ikegami, T., Virgo, N., Oka, M., and Iizuka, H. (2020). Artificial life next generation perspectives: Echoes from the 2018 conference in tokyo.

Wolff, J. (2012). *Ethics and public policy: a philosophical inquiry*. Routledge.

Zahavi, D. (2019). Second-person engagement, self-alienation, and group-identification. *Topoi*, 38(1):251–260.

Ziesche, S. and Yampolskiy, R. (2018). Towards ai welfare science and policies. *Big Data and Cognitive Computing*, 3(1):2.

Ziesche, S. and Yampolskiy, R. V. (2019). Do no harm policy for minds in other substrates. *Journal of Ethics and Emerging Technologies*, 29(2):1–11.

Zimmer, M. (2016). Extending court-protected legal person status to non-human entities. In *IJCA*, volume 8, page viii. HeinOnline.

Centralized and Decentralized Control in Modular Robots and Their Effect on Morphology

Mia-Katrin Kvalsund[1], Kyrre Glette[1,2] and Frank Veenstra[1]

[1]Department of Informatics, University of Oslo, Norway
[2]RITMO, University of Oslo, Norway
mia.kvalsund@gmail.com

Abstract

In Evolutionary Robotics, evolutionary algorithms are used to co-optimize morphology and control. However, co-optimizing leads to different challenges: How do you optimize a controller for a body that often changes its number of inputs and outputs? Researchers must then make some choice between centralized or decentralized control. In this article, we study the effects of centralized and decentralized controllers on modular robot performance and morphologies. This is done by implementing one centralized and two decentralized continuous time recurrent neural network controllers, as well as a sine wave controller for a baseline. We found that a decentralized approach that was more independent of morphology size performed significantly better than the other approaches. It also worked well in a larger variety of morphology sizes. In addition, we highlighted the difficulties of implementing centralized control for a changing morphology, and saw that our centralized controller struggled more with early convergence than the other approaches. Our findings indicate that duplicated decentralized networks are beneficial when evolving both the morphology and control of modular robots. Overall, if these findings translate to other robot systems, our results and issues encountered can help future researchers make a choice of control method when co-optimizing morphology and control.

Introduction

When co-optimizing morphology and control of modular robots, how do we optimize a controller for a robot that often changes its number of actuators and sensors? If a centralized approach is chosen, we must select a method to deal with disappearing actuators or the addition of new ones. Furthermore, although distributed control removes the issues of changing morphology, we must still then facilitate for global synchronization of the actuators. In this paper, we implement and discuss one centralized and three decentralized approaches to control and their effect on morphology. We suggest a decentralized control strategy that reuses control units across the robot body and demonstrate that such an approach leads to higher performance and more morphological development.

Throughout Evolutionary Robotics' short history, there have been many approaches to co-optimizing morphology and control. Most notably, the work of Karl Sims showed virtual creatures evolved using a nested graph where body and control elements were connected (Sims, 1994). Controllers were duplicated as body parts were copied in a semi-distributed approach. Lipson and Pollack (2000) later showed a pipeline to transfer such virtual creatures to reality, where they used centralized control. Later, Cheney et al. (2013, 2014) displayed soft-robots that evolved control through an indirect encoding using a Compositional Pattern Producing Network (CPPN). Similarly, Auerbach and Bongard (2011) used a CPPN encoding to generate a morphology and weights for a centralized continuous time recurrent neural network (CTRNN) controller. Both these approaches have reuse of control elements due to the indirect encodings.

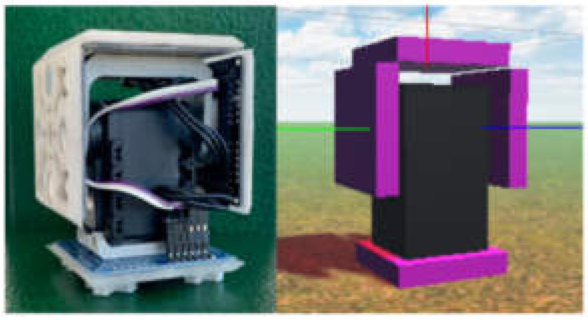


Figure 1: **The EMeRGE module**. The left image is an example of a real EMeRGE module, and the right is our Unity simplification.

A common problem when co-optimizing morphology and control is that of early convergence of morphology. As described by Joachimczak et al. (2016), and later explored further by Cheney et al. (2016), the morphology will reach its almost final form relatively early. Cheney et al. theorize that because the controller interacts with the environment through the interface of the body, changes to the body will scramble the control. However, this effect has not been studied much in modular robots. Decentralized control in modular robots could potentially decrease this issue because

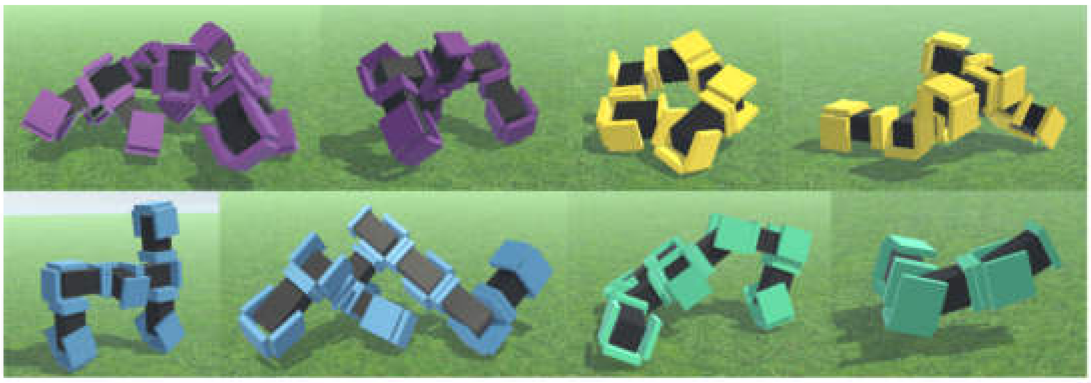


Figure 2: **Examples of well-performing morphologies.** Purple and top left: Copy controller. Yellow and top right: Sine controller. Blue and bottom left: Decentralized CTRNN controller. Teal and bottom right: Centralized CTRNN controller.

adding a new module with the same controller as its parent could still work without scrambling the overall performance.

Modular robotics (MR) concerns robots built from separable modules or units that encapsulate some function of a robotic system (Stoy et al., 2010). This is as opposed to an integrated design with no clear sectional modularity. The modules contain actuation, computation, energy, and sensing as needed, as well as some mechanism to connect and transmit to other modules. They can easily be reconfigured by hand or by machine, making them highly suited for rapid prototyping of robot designs.

The field of modular robotics is quite young, with some of the first notable papers, like the CEBOT (Kawauchi et al., 1992) and Fracta (Murata et al., 1994) papers, being published in the 90s. Modular robots promised easily reconfigurable robots that could adapt their form to any use (Yim et al., 2000). Early systems such as the M-TRAN showed self-reconfiguration into shapes for walking and climbing (Murata et al., 2002), and Zykov et al. (2005) showed the first minimal example of self-reproduction in modular robots. Throughout the 2000s and early 2010s, the focus was mostly on exploring the promise of reconfiguration and creating novel mechanical solutions for the modules.

Later Marbach and Ijspeert (2004), inspired by the works of Sims (1994) and Lipson and Pollack (2000), started to co-evolve configuration and control. With works like Marbach and Ijspeert's Adam and EDHMoR (Faíña et al., 2013), researchers started to experiment with the pipeline to create modular robots for any task. Evolutionary algorithms were uniquely suited for optimizing the robots, because the complexity of control scales exponentially with the number of modules (Marbach and Ijspeert, 2004). Additionally, by co-evolving we avoid the limitations and biases a human designer would bring, hopefully producing more novel and better adapted solutions (Faíña et al., 2013).

Many systems in MR use sine wave generator controllers when the focus of the research is elsewhere (Faíña et al., 2013; Liu et al., 2017; Veenstra et al., 2017), because they produce periodical movement with few parameters to optimize. This controller is a good baseline for the behavior of other controllers, as it is what we minimally expect from a decentralized controller.

Evolved centralized control is something that is not much used in MR, as the focus early on was on hand-crafted centralized reconfiguration systems (Murata et al., 2002). Later works have also shown centralized control that focus on task execution and locomotion (Brunete et al., 2012), however these are not evolved and do not necessarily scale well. It is still thought that some form of centralized control could be more suited to task execution (Seo et al., 2019), and so evolving centralized control should be explored.

Even so, decentralized control has also shown impressive results in task execution. Christensen (2006) showed an example of a decentralized neural network controller, which could self-reconfigure and self-repair. Their modules' swarm-like, imperfect behavior was able to control above 3000 modules in simulation, showing the scalability of good decentralized control. Another good example of neural network control is Jelisavcic et al.'s (2019) use of CPGs in the RoboGen modules. While this still results in distributed control, a CPPN encoding determines the CPG weights and can still enable module synchronization.

Our contribution to the field is two-fold: We present an investigation into centralized and decentralized controllers, and present a decentralized controller that reuses control units across the body. The controllers were implemented on a chain-type modular robot system. Through measuring performance and morphological diversity, we evaluate which control approach is better suited for co-optimizing morphology and control in light of premature convergence of morphology. Our findings show that the controller that reuses control units leads to an increase in performance and morphological development, which advocates for reuse of control elements in controllers in general.

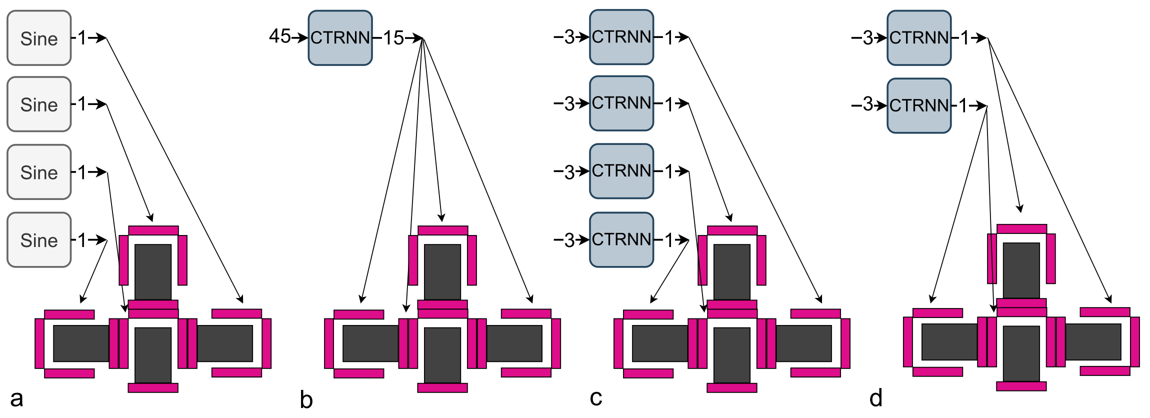


Figure 3: **The controllers and how they could map to the modules in a small modular robot.** The arrows with numbers represent inputs (left side of boxes) and outputs (right side of boxes). a) The sine controller, b) The centralized CTRNN controller, c) The decentralized CTRNN controller, d) The copy controller.

Methods

Our system consists of modular robots simulated in a flat ground environment and being measured on the task of locomotion. To co-optimize the morphology and controller, an evolutionary algorithm is used. This, as well as the four controllers we are investigating, will be presented below.

For this project, an environment and simulated modular robots were built in Unity with the framework of ML-Agents [1]. ML-Agents version 1.0.7 was used. Unity uses the Nvidia PhysX physics engine, which supports rigid body dynamics and updates physics steps every 0.02 seconds. The default physics engine settings were used.

The Modules

We are using the EMeRGE module (Moreno et al., 2017), see Figure 1. It is a simple module with one servo motor, so that each module works like a hinge. It has four connection faces, one male at the base, and three female on the top and sides. The male connection face can only connect to the female ones and vice versa, meaning the robot will have a root module with three possible child modules, growing outwards in a tree-like structure. We do not allow connections to break. The joint is driven by a spring joint with 200 in spring torque and a damping value of 5. The max force is not constrained. Example morphologies can be seen in Figure 2.

The original design for the modules includes infrared proximity sensors on each face. This was also implemented in our abstraction of the module, although the workings of the exact sensor model were not replicated. Instead, sensors register all distances through a ray cast, meaning the distances it can register are not capped and could get very high. For not registering a distance, for example from being angled towards the sky, a sensor returns -1. When a child module occupies a site, the sensor will register the small distance towards the child.

The Controllers

There are four controllers implemented in this system (Figure 3), each described below. For all of them, the output produced is directly the desired angle of a module's servo. The simulated module constrains the movement to +/-90°.

Three of the controllers use a continuous time recurrent neural network (CTRNN)[2]. This network was chosen to have the possibility of dynamic temporal behavior (Beer, 1995). Here each node updates based on a differential equation, with neuron potentials as dependent variables.

The CTRNN mutation operators can adjust weights and biases, change activation and aggregation functions, and cut away/disable or add/enable connections and nodes. The gene mutation rates are equal for all our CTRNN controllers and are given in the source code. These rates were then scaled by the controller's control mutation rate.

Open-loop sine wave generator The open-loop sine wave generator is a decentralized controller that performs well because it produces periodic movement through sine waves, although it has no sensor input. In our case, the sine wave controller is used to provide a baseline to compare the other controllers to. The controller is given by the function

$$y(t) = A * \sin(w * t + p) + o \quad (1)$$

where A is the amplitude, w is the frequency, t is time, p is phase, and o is offset. $y(t)$ is the controller output at time t that is directly fed to the servo's desired angle.

[1]Source code at `https://github.com/mia-katrin/Modbots`

[2]The CTRNN implementation used is from the neat-python library `https://neat-python.readthedocs.io/en/latest/`

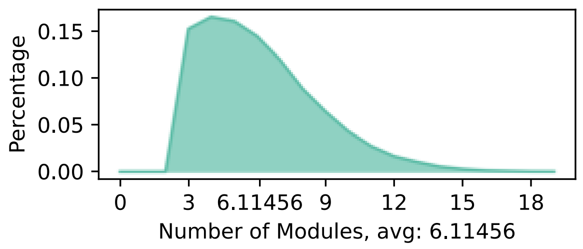


Figure 4: **The distribution of number of modules in individuals in a random population before optimization.**

To enable synchronization between the modules, the frequency was fixed in all sine wave controllers and was not allowed to mutate. We noticed that without this, the sine controller was susceptible to choose local optima solutions. The sine wave controller has 3 parameters for each joint, meaning an average robot of 6 modules will have 18 parameters to optimize for the controller. When a module is added to the morphology, it is instantiated with the control parameters of the parent module.

Centralized CTRNN A straightforward approach to using a CTRNN for a modular robot is to simply gather all sensor outputs and feed them into one big CTRNN, that then outputs all controller actions. This leads to a fixed size CTRNN controller.

In initial experiments with the centralized CTRNN controller, we tested numbers of inputs and outputs corresponding to 50 modules. Because there was a clear tendency to have a small number of modules, we tested with fewer inputs and outputs until finally we chose 15 to be the number of modules that would be controlled. This allowed the network to be as small as possible while still accommodating larger creatures at initialization, however it was very rare to see larger creatures with this controller. When a modular robot is smaller than 15 modules, the rest of the inputs to the network is set to 0.

The centralized CTRNN controller has 45 inputs and 15 outputs. It is initialized with 45 hidden nodes but is only 20% connected. The order of mapping modules to input and output follows a depth first ordering of the modules, so that the first three inputs and first output goes to the root, the second three inputs and second output goes to its child, etc.

The centralized CTRNN controller ends up having circa 600 connections and 60 nodes, each with respectively 3 and 5 parameters to tune, for a total of 2100 parameters.

Decentralized CTRNN Foregoing the benefits of a centralized brain, a decentralized approach leads to less parameters and a less complex optimization. The decentralized approach consists of each module getting its own CTRNN controller. When a module is added, it is instantiated with the control parameters of the parent module. Here, we used a small compact network of 3 inputs and 1 output, and with 3 hidden nodes and enabling all connections from the start, we get 4 nodes and 16 connections, totaling 68 parameters. For an average sized robot with one of these controllers in every module, this leads to about 400 parameters.

Copy Decentralized CTRNN The copy decentralized CTRNN controller, or the copy controller for short, is our alternative to the decentralized approach. It functions by having each robot keep a list of two CTRNN networks, which maps to different modules. The networks are the same as in the decentralized CTRNN controller. At initialization, these networks are clones, but as the optimization progresses, they will mutate separately. The modules will then mutate which network they use for control, theoretically allowing specialization. When a module is added to the morphology, it will use the same network as its parent module.

At the start of an evaluation, the networks are copied into their corresponding modules, hence the name. They then function independently of each other, and because of different sensor input such as detecting ground or the presence of child modules, they will likely behave differently. Nevertheless, it is reasonable to assume it will not achieve the level of specialization that the decentralized CTRNN controller can. While this might be a trade-off, we assume the copy controller will be quicker to achieve a good fitness, and possibly not be as dependent on number of modules.

This controller, like the centralized CTRNN controller, is the same no matter the size of the robot. This gives us two controllers with 68 parameters, for a total of 136 parameters. Additionally, each module can change which controller they use, giving us a further average of 6 parameters.

Evolutionary Algorithm

The evolutionary algorithm used had tournament selection, with a tournament size of 4, and generational replacement. It was implemented using the DEAP framework (Fortin et al., 2012). Generational replacement was chosen because it can sometimes dislodge a population from early convergence. This happens because the best genotype will rarely be kept when the population is mutated and no elites are kept. Other parameters of the algorithm were also chosen to keep diversity. Most notably, the tournament selection size was small to increase selection pressure on the elites, while still being large enough to avoid a noisy evolutionary progression.

The morphology and the controller had separate mutation rates. All controller gene values mutated with a Gaussian distribution based on the mutation power. For the rates, a parameter sweep was done for each controller on a grid of 8 values for both controller and morphology (Figure 5), a total of 64 pairs. Each pair was run for 50 generations with a population size of 50, which totaled 2500 evaluations. This was done 4 times for each pair. Finally, the columns and rows in

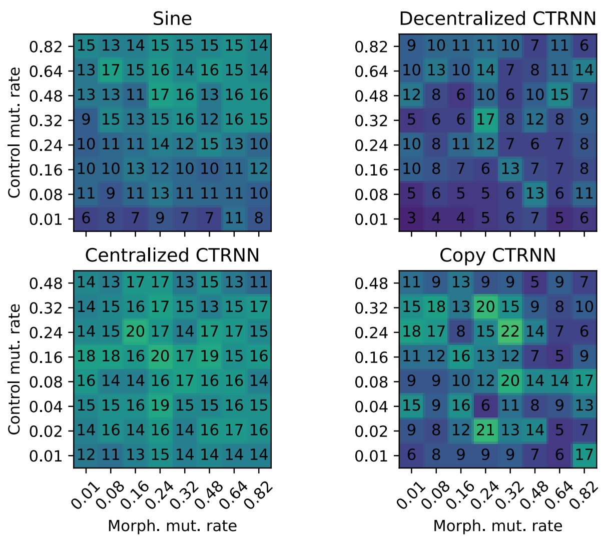


Figure 5: **The grids filled for the parameter tuning**, values are rounded average fitness for each mutation rate pair.

the grid were collapsed and plotted because there were a lot of variation between the pairs. The best performing column and row value were then chosen for each controller.

The 8 sweep values were chosen for each controller and the body based on whether they divided the mutation rate internally. In the body, the sine, and the decentralized CTRNN controller, the mutation rate is divided by the number of modules so that creatures mutate the same amount no matter the size. This encourages the use of more modules because larger creatures are not more unstable solutions. Since the average number of modules is 6 (Figure 4), the per-module mutation rate is 0.14 for the largest sweep value of 0.82. In the morphology mutation, this is further multiplied by circa 0.25 for each gene in the module.

Encoding of robots

For the encoding of the robots, a direct encoding is used. A directed tree is generated from a root node, after which it is sent to the simulator. The simulator builds the robot and prunes any branches that collide, prioritizing keeping modules closer to the root. Modules that are not expressed are still kept in the genome.

The morphology can mutate by adding and removing modules, as well as changing the angle of modules. There is a slightly higher chance to get an add module mutation in order to bias creatures to grow. Additionally, one mutation will make a module duplicate one child branch to another connection site. This mutation was chosen to facilitate symmetry and larger jumps in the morphology landscape.

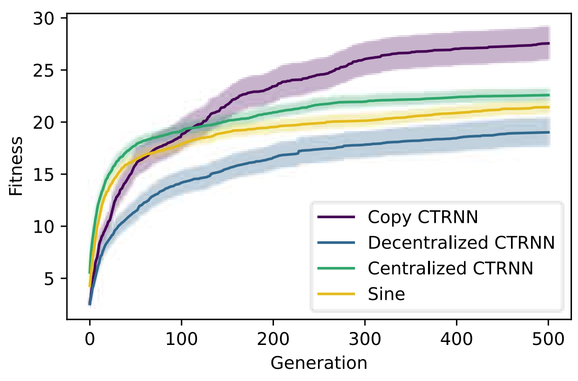


Figure 6: **The fitness progressions for all four controllers.** The solid lines are averages of the best individuals in each generation, and the shaded areas represent the standard error.

Fitness function

The task that we measured the modular robots on was locomotion away from the origin during a set amount of time (100 steps of 0.2 seconds, roughly equaling 20 seconds in real time). Because this often leads to robots discovering the immediate optima of the somersault, or simply falling over, the robots were given 2 seconds to fall before the evaluation started. At the same time, the controller is not given input and will not give output. The fitness function is then

$$Fitness = \sqrt{(x_{end} - x_{start})^2 + (y_{end} - y_{start})^2} \quad (2)$$

where x_{start} is the x-position after 2 seconds of simulation, and x_{end} is the x-position when the simulation ends. Likewise for y. The fitness is measured in units that correspond to the height of one module, which is circa 8 cm. A fitness of 30 would therefore mean 2.4 m has been travelled. The evaluation will stop early after 4 seconds from start if the robot has not moved in the last 2 seconds.

Results

Mutation rate sweep

As can be seen in Figure 5, the results from the sweep were often quite even. Only the copy and decentralized CTRNN controllers saw huge differences between different pairs, but all controllers had only minor differences after collapsing the data. Therefore, we settled on choosing a few of the ones that were contenders after the initial sweep and run a few more evolutionary runs on those. A winner would then often be clearer. The final parameters for all controllers can be seen in Table 1.

Controller performance

The final runs were done on populations of 50 individuals for 500 generations, for a total of 25 000 evaluations. This

Table 1: **The mutation rate parameters chosen after the sweep.** Note that for the sine wave and decentralized CTRNN controller, as well as the morphology, the working mutation rate on one module is divided by the number of modules in the robot.

Controller	Morph. rate	Controller rate
Sine wave	0.32	0.64
Centralized CTRNN	0.24	0.16
Decentralized CTRNN	0.24	0.48
Copy CTRNN	0.32	0.08

was done for all controllers 64 times, and the resulting performances can be seen in Figure 6. Figure 7 shows the distribution of performances for the different controllers.

In order to get an overview of all significant differences, 6 two-sided Mann-Whitney U test were performed between all controllers at generations 50 and 500, a total of 12 tests. An alpha level of 0.05 was chosen. Because we were conducting multiple comparisons, Bonferroni correction was used. This gives us an adjusted alpha level of 0.05 / 12 = 0.00416.

At generation 50, the sine and centralized CTRNN controllers were significantly different from the decentralized CTRNN controller (both $p < 0.0001$). There was no significant difference between the sine and copy ($p > 0.2$), sine and centralized CTRNN ($p > 0.07$), copy and centralized CTRNN ($p > 0.06$), and the copy and decentralized CTRNN controllers ($p > 0.01$).

At generation 500, the copy controller was significantly different from the centralized CTRNN, the sine, and the decentralized CTRNN controllers (respective p-values $p < 0.003$, $p < 0.0004$, $p < 0.0002$). There was no significant difference between the sine controller and the centralized and decentralized CTRNN controllers (both $p > 0.1$), or between the centralized and the decentralized CTRNN controllers ($p > 0.04$).

Effect on morphology

In Figure 8, the progressions for number of modules in morphologies are plotted for all the controllers. Here, we can see that the sine and centralized CTRNN controllers both end up at a lower average number of modules than the copy and decentralized CTRNN controllers. To confirm if this was significant, as previously 6 two-sided Mann-Whitney U tests were performed with Bonferroni correction between the different count distributions. An alpha level of 0.05 was used, which means that with correction we consider p-values below 0.05 / 6 = 0.0083 as significant.

Here we found that the copy and decentralized CTRNN controllers had no significant difference between them ($p > 0.4$) and the sine and centralized CTRNN controllers likewise had no difference ($p > 0.2$). However, the copy and decentralized CTRNN controllers were both different from the

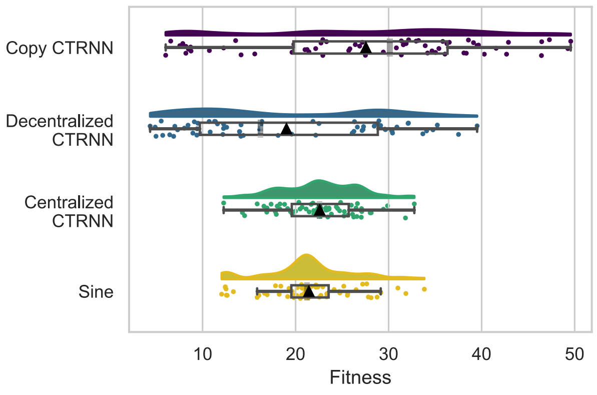


Figure 7: **The final fitnesses for all runs for each controller.** The distributions are showed with the underlying points scattered below. A boxplot is placed over the scattered values, with the mean marked with a triangle.

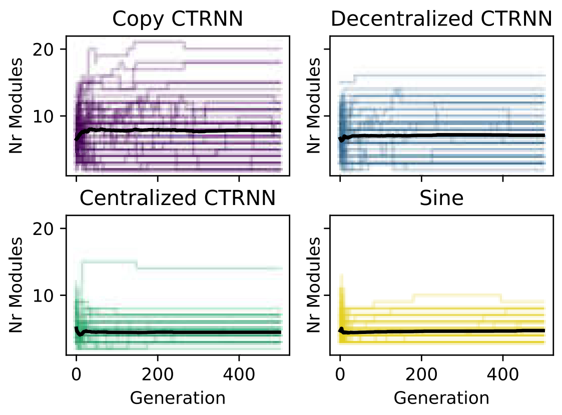


Figure 8: **The progression of number of modules for all runs.** Only changes in number of modules that led to a more fit individual is shown.

sine and centralized CTRNN controllers (all $p < 0.0001$).

Qualitatively, we recognize this from looking at the robots. The sine and centralized controllers had small, effective strategies while the copy and decentralized CTRNN controller both tended towards larger morphologies. The sine and centralized CTRNN controllers would do large, powerful movements, with some modules in the morphology not moving at all. As opposed to this, the copy and decentralized CTRNN controllers favored small, rapid movements. Here, the copy moved most its modules, while the decentralized CTRNN controller sometimes had unmoving modules. Example behaviors can be seen in the accompanying video[3].

In Figure 9, we see that there seems to be a divide between

[3]`https://www.mn.uio.no/ifi/english/research/groups/robin/research-projects/cocomo/modbots.html`

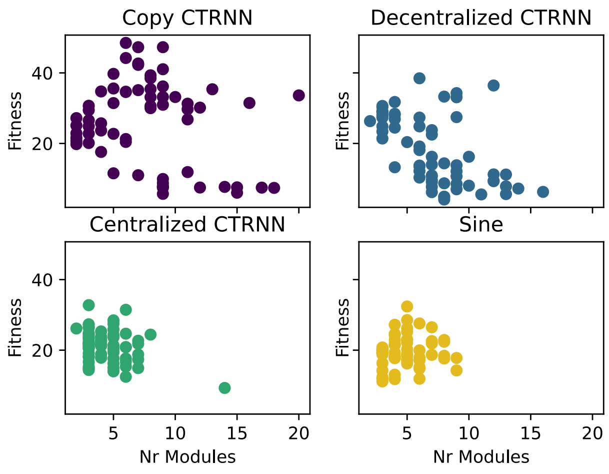


Figure 9: **The expressed modules of all robots plotted against their fitness.**

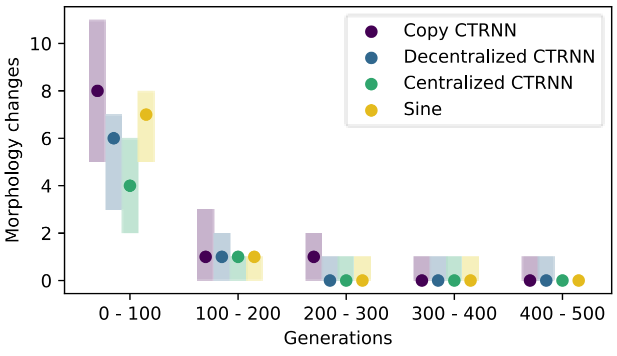


Figure 10: **Number of morphology changes that led to better fit individuals in each generation interval**. The shaded areas are between the 25 and 75 percentiles, and the dots are the medians.

larger creatures that get high fitness, and those that get low fitness. Presumably, this is because falling strategies tend to be larger in order to get further, but it is also interesting to see that some larger solutions did quite well. Especially the copy controller did well with larger morphologies, managing to produce a fitness on par with the centralized CTRNN and sine controllers with upwards of 10 expressed modules.

When looking into the issue of early convergence of morphology only, Figure 10 was made. It shows shaded areas between the 25 and 75 percentiles of number of beneficial morphology changes in a run in each interval of 100 generations. It shows that the centralized CTRNN and sine controllers stop mutating morphology after 400 generations. The last two controllers kept mutating all the way up until the end. Interestingly, 72% of the copy controller runs experienced morphology changes in the 100-200 generation interval, and more than 50% had morphology changes in the 200-300 interval.

Discussion

Our results have shown that the copy controller performs significantly better than other controllers when co-optimizing the morphology and control of modular robots. Since it duplicates behaviors, modules are more likely to be synchronized. Moreover, when a new module is added, a working control unit can be inherited that is already potentially useful. Even though the sine and decentralized CTRNN controllers had a similar feature of inheriting the parent module's control, the copy controller is more likely to be useful since it is already evolved to work in many different parts of the robot. In addition, because a control mutation affects multiple modules, it has an overall larger effect on the behavior of the robot compared to a mutation in the other controllers. Because of this, the copy CTRNN approach is less able to fine-tune a single controller compared to the other approaches and therefore may rely on morphological change to see a performance increase. This feature would thereby promote continued morphological diversification compared to the other controllers, as seen in Figure 10.

From the fitness progressions, we can see that the sine and centralized CTRNN controllers converged rather fast compared to the other two. They also showed a pattern of quickly finding a final morphology of relatively small size, and then optimizing the controller. Meanwhile, the other two controllers spent time developing both and thus converged slower. Because having more modules means the robot has more potential force, allowing for more movement and higher fitness overall, the sine and centralized CTRNN controllers were then at a disadvantage. These results confirm that there is a trade-off between fine-tuning controllers and getting a good fitness with a small morphology while losing the potential of getting a higher fitness and a large morphology.

The distributions of solutions for the controllers vary wildly, as seen in Figure 7. While the sine and centralized CTRNN controllers had a more solidly high performance, the decentralized CTRNN controllers both had very flat distributions, stretching from the worst to the best performances recorded. The lower fitnesses can be accounted for as robots that grow tall and fall in one direction, as some of these have been visually confirmed to be. The higher values of the copy, decentralized, and centralized CTRNN controllers often had rapid module movements that either led to small jumps or shuffling behaviors. Because of fixing the sine controller's frequency, this strategy was not available to the sine controller, and so its worse performance must at least be partially attributed to that. Even though it could have rivalled the others by growing larger, the CTRNN controllers' behavior was likely less complicated to evolve.

Still, the sine and centralized CTRNN much more often arrived at very similar local optima, which tended to have the same fitness. Here, the centralized CTRNN controller had an advantage over the sine controller because it could optimize further by adding non-periodic movements.

Because we had the same morphology mutation rate for the sine and copy controllers, we could expect similar morphological diversity from these. However, it is clear from our results that this is not the case. When keeping in mind that some of the more scalable strategies available to the CTRNN controllers were not available to the sine controller, it could simply be that there were less available good morphologies for the sine controller.

The centralized CTRNN approach that was implemented could evolve a network topology that connected to up to 15 modules. This means that evolving larger morphologies would involve the CTRNN accommodating for more outputs. Since the CTRNN in this case would then have even more parameters to optimize, we would expect the centralized CTRNN to converge even quicker. This could potentially be overcome by connecting parts of the neural network of the centralized CTRNN to the morphology and copying these parts of the CTRNN when a new module is introduced. Another possible cause for the rapid convergence seen in the centralized CTRNN is that the initial experiments to determine the supported number of modules only ran for 100 generations. Although we have no indication that it would, the centralized CTRNN approach could generate larger morphologies when given different mutation rate values.

The copy and decentralized controllers both had issues of some number of unexpressed modules being added to the genome. These were unexpressed because they collided with other modules or the floor. 2 and 4 out of 64 samples from respectively the copy and decentralized CTRNN controllers had 10 to 20 unexpressed modules. Since they were unexpressed, the only effect they had on the individual was lowering the per-module mutation rate. This meant that the morphology in both, and the controllers in the decentralized CTRNN, would mutate less as the number of unexpressed modules grew. This bloating of the genome would in theory stabilize them from mutating. The two other controllers did not have issues with this.

Another issue is that of some pervasive local optima. Likely due to the angular shape of the EMeRGE modules, initial populations of robots found success with a single module dragging itself forward. In an attempt to avoid this, we constrained the robots to having a limp root module. This mitigated the problem somewhat, but similar strategies of using the corners of the female connection plates persisted all throughout the project. For example, the aforementioned jumping and shuffling behaviors are likely only possible because of the module shape.

To avoid local optima, we tailored our algorithm to keep diversity, for example by having generational replacement. While using a diversity maintenance method could have minimized the occurrence of local optima solutions, it would have been difficult to parse which results were caused by the controllers and which were caused by the algorithm. Even so, in future work the same controller approaches could be tested with diversity enhancing methods to investigate what different control behaviors arise.

Because the sensor implementation was not very realistic, it could be useful to focus on adding more realism and to measure the different controllers on tasks or environments that require more sensing. Here, we would be better able to study different strategies that emerged, and whether the controllers were as equally equipped for task execution as they were for locomotion. It seems probable that the centralized CTRNN would perform better than the others here. The decentralized approaches would benefit from communication between modules, and coupled CPGs such as the ones used by Ijspeert et al. (2007) could likely work well here.

Lastly, to test if these findings translate to other robotic systems, the same controller types should be implemented on a system with different modules and/or controllers. Since the EMeRGE module's female connection plates can be used for dragging and jumping, a less angular option like the RoboGrammar modules (Zhao et al., 2020) could force the controllers to choose more complicated movements. Additionally, the previously discussed CPGs can be used to achieve periodic motion and would be a good option to further incorporate the sensors in locomotion.

Conclusion

In this article we implemented and tested four controllers that were co-optimized along with a modular robot morphology. With testing three decentralized and one centralized controller, we got insight into how these can be done well and different challenges that arises for each approach. Markedly, we learned that there is significant advantage to simplify your controller to facilitate for global synchronization, as was found when the copy controller outcompeted the decentralized CTRNN controller. The copy approach allows for better new control of added modules, thus more morphological development, and larger jumps in the search space. Regarding centralized control, we highlighted the early convergence of morphology and performance when it comes to having a complex controller to optimize. Given that these findings translate to other controller networks and morphologies, they can aid future choices of control when co-optimizing morphology and control.

Acknowledgements

This work was performed on the Fox resource, owned by the University of Oslo Center for Information Technology. This work was partially supported by the Research Council of Norway through its Centres of Excellence scheme, project number 262762.

References

Auerbach, J. E. and Bongard, J. C. (2011). Evolving complete robots with cppn-neat: the utility of recurrent connections. In *Proceedings of the 13th annual conference on Genetic and evolutionary computation*, pages 1475–1482.

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

Brunete, A., Hernando, M., Gambao, E., and Torres, J. E. (2012). A behaviour-based control architecture for heterogeneous modular, multi-configurable, chained micro-robots. *Robotics and Autonomous Systems*, 60(12):1607–1624.

Cheney, N., Bongard, J., SunSpiral, V., and Lipson, H. (2016). On the difficulty of co-optimizing morphology and control in evolved virtual creatures. In *Proceedings of the Artificial Life Conference 2016, ALIFE 2016*.

Cheney, N., Clune, J., and Lipson, H. (2014). Evolved electrophysiological soft robots. In *Artificial Life 14 - Proceedings of the 14th International Conference on the Synthesis and Simulation of Living Systems, ALIFE 2014*.

Cheney, N., MacCurdy, R., Clune, J., and Lipson, H. (2013). Unshackling evolution: Evolving soft robots with multiple materials and a powerful generative encoding. In *GECCO 2013 - Proceedings of the 2013 Genetic and Evolutionary Computation Conference*.

Christensen, D. J. (2006). Evolution of shape-changing and self-repairing control for the atron self-reconfigurable robot. In *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006.*, pages 2539–2545. IEEE.

Faíña, A., Bellas, F., López-Peña, F., and Duro, R. J. (2013). EDHMoR: Evolutionary designer of heterogeneous modular robots. *Engineering Applications of Artificial Intelligence*.

Fortin, F.-A., De Rainville, F.-M., Gardner, M.-A., Parizeau, M., and Gagné, C. (2012). DEAP: Evolutionary algorithms made easy. *Journal of Machine Learning Research*, 13:2171–2175.

Ijspeert, A. J., Crespi, A., Ryczko, D., and Cabelguen, J.-M. (2007). From swimming to walking with a salamander robot driven by a spinal cord model. *science*, 315(5817):1416–1420.

Jelisavcic, M., Glette, K., Haasdijk, E., and Eiben, A. E. (2019). Lamarckian Evolution of Simulated Modular Robots. *Frontiers in Robotics and AI*.

Joachimczak, M., Suzuki, R., and Arita, T. (2016). Artificial Metamorphosis: Evolutionary Design of Transforming, Soft-Bodied Robots. *Artificial Life*, 22(3).

Kawauchi, Y., Fukuda, T., and Inaba, M. (1992). "a strategy of self-organization for cellular robotic system (cebot)". In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, volume 3, pages 1558–1565. IEEE.

Lipson, H. and Pollack, J. B. (2000). Automatic design and manufacture of robotic lifeforms. *Nature*, 406(6799).

Liu, C., Liu, J., Moreno, R., Veenstra, F., and Faina, A. (2017). The impact of module morphologies on modular robots. In *2017 18th International Conference on Advanced Robotics, ICAR 2017*.

Marbach, D. and Ijspeert, A. J. (2004). Co-evolution of configuration and control for homogenous modular robots. In *Proceedings of the eighth conference on intelligent autonomous systems (IAS8)*, pages 712–719. IOS Press.

Moreno, R., Liu, C., Faina, A., Hernandez, H., and Gomez, J. (2017). The emerge modular robot, an open platform for quick testing of evolved robot morphologies. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, pages 71–72.

Murata, S., Kurokawa, H., and Kokaji, S. (1994). Self-assembling machine. In *Proceedings of the 1994 IEEE International Conference on Robotics and Automation*, pages 441–448. IEEE.

Murata, S., Yoshida, E., Kamimura, A., Kurokawa, H., Tomita, K., and Kokaji, S. (2002). M-TRAN: Self-reconfigurable modular robotic system. *IEEE/ASME Transactions on Mechatronics*, 7(4).

Seo, J., Paik, J., and Yim, M. (2019). Modular Reconfigurable Robotics. *Annual Review of Control, Robotics, and Autonomous Systems*, 2(1).

Sims, K. (1994). Evolving virtual creatures. In *Proceedings of the 21st Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH 1994*, pages 15–22.

Stoy, K., Brandt, D., Christensen, D. J., and Brandt, D. (2010). *Self-reconfigurable robots: an introduction*. Mit Press Cambridge.

Veenstra, F., Faina, A., Risi, S., and Stoy, K. (2017). Evolution and morphogenesis of simulated modular robots: A comparison between a direct and generative encoding. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 10199 LNCS.

Yim, M., Duff, D. G., and Roufas, K. D. (2000). PolyBot: a modular reconfigurable robot. In *Proceedings - IEEE International Conference on Robotics and Automation*, volume 1.

Zhao, A., Xu, J., Konaković Luković, M., Hughes, J., Speilberg, A., Rus, D., and Matusik, W. (2020). Robogrammar: Graph grammar for terrain-optimized robot design. *ACM Transactions on Graphics (TOG)*, 39(6):1–16.

Zykov, V., Mytilinaios, E., Adams, B., and Lipson, H. (2005). Self-reproducing machines. *Nature*, 435(7039):163–164.

What does functional connectivity tell us about the behaviorally-functional connectivity of a multifunctional neural circuit?

Eduardo J. Izquierdo, Madhavun Candadai

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

Abstract

What insights can statistical analysis of the time series recordings of neurons and brain regions during behavior give about the neural basis of behavior? With the increasing amount of whole-brain imaging data becoming available, the importance of addressing this unanswered theoretical challenge has become increasingly urgent. We propose a computational neuroethology approach to begin to address this challenge. We evolve dynamical recurrent neural networks to be capable of performing multiple tasks. We then analyze the neural activity using popular network neuroscience tools, specifically functional connectivity using Pearson's correlation, mutual information, and transfer entropy. We compare the results from these tools against a series of informational lesions, as a way to reveal their degree of approximation to the ground-truth. Our initial analysis reveals an overwhelming large gap between the insights gained from statistical inference of the functionality of the circuits based on neural activity and the actual functionality of the circuits as revealed by mechanistic interventions.

Introduction

A central goal in neuroscience is to understand how the brain, body and environment come together to produce behavior. Specifically, we would like to understand in some detail the functional role of the nervous system in behavior. To this end, researchers are imaging with increasing time and spatial resolution the neural activity of living organisms at various scales, ranging from *C. elegans* to humans (Nguyen et al., 2016; Aimon et al., 2019; Randlett et al., 2015). Furthermore, technological advancements are starting to make recording of neural activity from freely moving animals possible (Lin et al., 2022). This increase in neural activity data has led to a similar increase in statistical measures and methods for inferring function from the time series of the neural activity (Paninski and Cunningham, 2018; Ramaswamy, 2019). Despite the incredible experimental progress and the overwhelming explosion in data availability, a fundamental theoretical challenge remains open (see Fig. 1): What can statistical measures of neural activity during behavior reveal about the function of the components of the nervous system?

Of the wide range of statistical methods that are available, the application of network theoretic tools to interpret animal brain activity as it pertains to behavior and disease has seen an explosion of interest in the last decade (Sporns, 2010; Fornito et al., 2016). Specifically, there has been a myriad of methods for constructing functional connectivity networks from neural activity to understand the interaction between brain regions at various scales with the ultimate goal of understanding the underlying causal relationships (Van Den Heuvel and Pol, 2010; Smith et al., 2011; Yeo et al., 2011). Of these, the most popular methods include Pearson's correlation, mutual information and transfer entropy (Friston, 1994). While these statistical methods can provide very useful insights about the interactions between the different components of the neural system, they provide no guarantees as to their ability to converge to the ground-truth causal relationships.

Computational models of neural networks have proven to be an excellent test bed for generating and evaluating such statistical methods (Dayan et al., 2003). For instance, using a computational model of a fully-connected spiking neural network, Ito et al. (2011) showed that while transfer entropy can get close, it still cannot estimate the structural connectivity of a neural network from its activity alone. Similarly, using a recurrent dynamical neural network model optimized to perform a task, Candadai and Izquierdo (2020) showed that mutual information cannot disambiguate between predictive information from different sources. Maheswaranathan et al. (2019) showed that analysis of some features of the representation geometry led to conclusions that were not related to the function of the network, while others did. Similar approaches have been taken to show that polyadic interactions and the presence of underlying common inputs present challenges to these methods (Stevenson et al., 2008; James and Crutchfield, 2017). In such studies, two aspects that is often overlooked are: (a) animals don't function as "brains in a vat" but are embodied and embedded in the environments that they are continuously interacting with; and (b) natural systems are multifunctional, however most computational models that are studied are typically built to perform only a single behavior. This is where computational neuroethological approaches to understand-

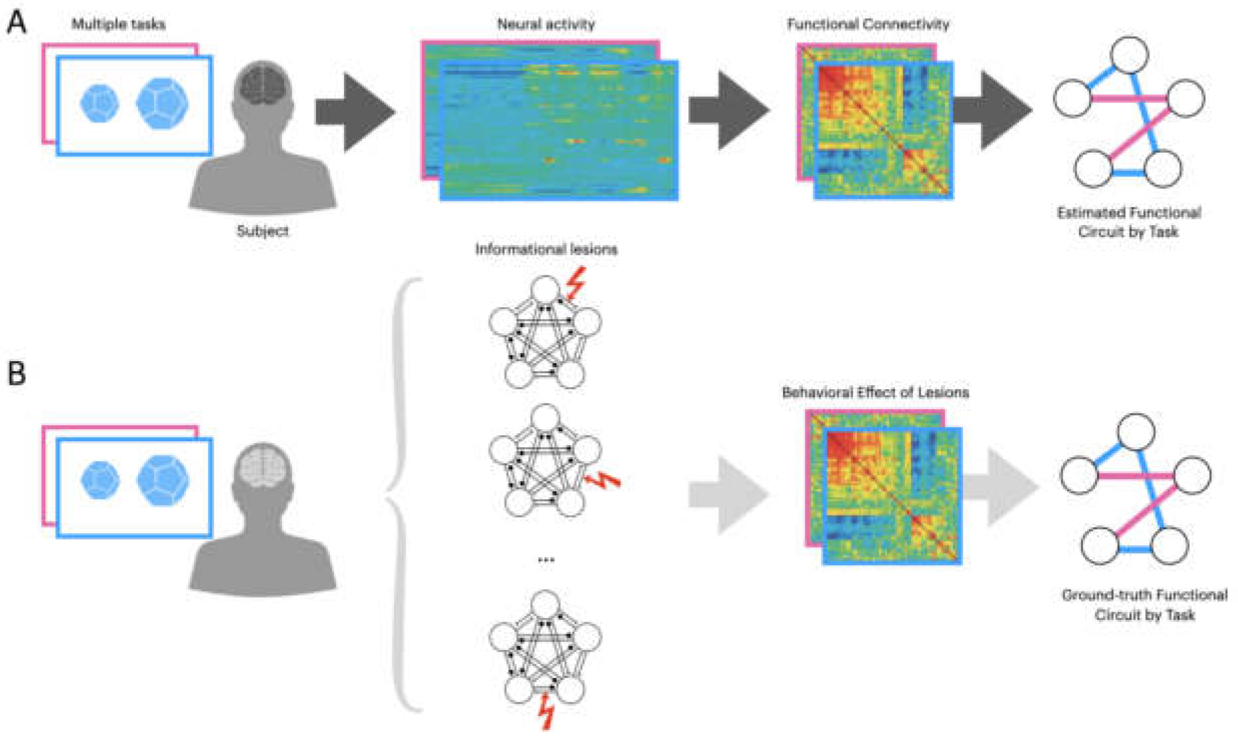

A
Multiple tasks
Subject
Neural activity
Functional Connectivity
Estimated Functional Circuit by Task
Informational lesions
B
...
Behavioral Effect of Lesions
Ground-truth Functional Circuit by Task

Figure 1: Computational neuroethology approach to uncover what a statistical measure of functional connectivity tell us about the actual functional connectivity of a nervous system. In order to address this theoretical challenge, we propose the following paradigm. In scenario (**A**), the subject is presented with two different tasks (blue and magenta). For each task, there are multiple trials (different sizes, different starting conditions). For each trial, neural activity is recorded for the subject. From the combined neural recordings of each task, a node functional connectivity (nFC) is created, using one of three techniques: Pearson's correlation, mutual information, and transfer entropy. Finally, from the nFC the subcircuit for each task is estimated. In scenario (**B**), the same subject is now tested on the same two tasks, but now the drop in performance is recorded during information lesions to each pair of connections or each individual connection between the components of the subject's brain. This effect of lesions per pairwise component is considered the actual functional connectivity (aFC). From it, the ground truth functional circuit is obtained for each task, which is used to assess the usefulness of the statistical nFC approach. We cannot do part B of this approach with humans, or with any other living organism, given current experimental limitations and ethical considerations. However, we can use artificial life techniques to: first generate agents capable of multiple tasks, and then analyze them in the way proposed above.

ing the neural basis of behaviors come in (Beer, 1996; Datta et al., 2019; Candadai, 2021): build computational models of brain-body-environment (BBE) systems, optimize them to perform multiple tasks, and use them as ground-truth for neural network operation with *behavior* defining what function means.

We would like to revisit the question that is at the heart of all the work that uses Network Neuroscience tools from a computational neuroethology perspective: What can statistical analysis of the time-series of the neural activity of brain-body-environment systems tell us about the behavioral functionality of the system? Our overarching goals are to: First, build intuition about what the statistical methods tell us about the ground-truth. Second, use this to guide and inform predictions and generate hypotheses in experiments. Finally, improve the tools of analysis for complex BBE systems. In this paper, we would like to tackle the first of those goals in the simplest set of conditions possible: (1) Evolve a BBE model to perform a pair of visually-guided behaviors; (2) Infer the functional connectivity for each task from neural activity time series using Pearson's correlation, mutual information, and transfer entropy; (3) Compare the insights gained against the ground-truth obtained from an informational lesion characterization of the circuit.

This paper is organized as follows: the next section describes the design of visually-guided behaviors, and the agent; the following sections describe our results with regards to how the agent performed said behaviors and the comparison between statistically-inferred functional connectivity and the ground-truth; finally we discuss our results and present ideas for future work.

Methods

We replicated the visually-guided agent described in (Beer, 1996), including two tasks previously used in (Beer, 1996; Slocum et al., 2000). The model agent is illustrated in Figure 2. The agent has a circular body with a diameter of 30 (in an environment of size 400×275). The agent possesses an "eye" consisting of a foveated array of distance sensors. The eye consists of 15 proximity sensors of maximum length 220, uniformly distributed over a visual angle of $\pi/4$. An intersection between a ray and an object causes an input to be injected into the corresponding sensory neuron. The magnitude of the injected input is inversely proportional to the distance to the object, with values ranging from 0 (no intersection) to 10 (no separation). The agent has two "motors" that produce 1D movement of the entire body. The agent moves according to first-order dynamics, with motor neurons directly specifying the velocity of movement. The agent's horizontal velocity is proportional to the sum of opposing forces produced by a bilateral pair of effectors (with a constant of proportionality of 8).

The agent's behavior is controlled by a continuous-time recurrent neural network (CTRNN) with the following state equation:

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

where y is the state of each neuron, τ is its 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, and I represents an external input (e.g., from a sensor). States were initialized to 0 and circuits were integrated using the forward Euler method with an integration step size of 0.1.

A real-valued genetic algorithm was used to evolve CTRNN parameters. A population of individuals was maintained, with each individual encoded as a length M vector of real numbers. Initially, a random population of vectors was generated by initializing each component of every individual to random values uniformly distributed over the range ± 1 (they could not move outside this range during evolution). Individuals were selected for reproduction using a linear rank-based method. Children were generated by either mutation or crossover with an adjustable crossover probability. A selected parent was mutated by adding to it a random displacement vector whose direction was uniformly distributed on the M-dimensional hypersphere and whose magnitude was a Gaussian random variable with 0 mean and variance σ^2 (for details, see Slocum et al. (2000)). Search parameters in the range ± 1 were mapped linearly into CTRNN parameters with the following ranges: connection weights $\in [-5, 5]$, biases $\in [-5, 5]$, and time-constants $\in [1, 2]$. The parameters of the neural circuit were bilaterally symmetric.

For the first task, the embodied agent must be capable of visually discriminating between objects of different sizes, catching smaller circular objects while avoiding the larger circular objects. Objects fell straight down with an initial horizontal offset in the range ± 25 and a vertical velocity of 3. The circular objects had a diameter in the range $[20, 40]$. Accordingly, the performance measure to be maximized was:

$$f_A = \frac{1}{T}\sum_{i=1}^{T} p_i \tag{2}$$

where $p_i = 1 - d_i$ for smaller circular objects and $p_i = d_i$ for larger circular objects, d_i is the horizontal distance between the centers of the object and the agent when their vertical separation goes to zero on the ith trial (clipped to MaxDistance and normalized to run between 0 and 1), T is the total number of trials, and D is the maximum distance. The reason that d_i was clipped to D was to prevent the avoidance of, for example, larger circles by large distances from dominating the fitness at the expense of accuracy in catching smaller circles. A total of 24 evaluation trials were used during evolution, uniformly distributed over the range of horizontal offsets.

For the second task, the embodied agent must become sensitive to the relationship of its own body to its surroundings and it must be able to perceive the actions that this environment affords. We evolved agents that could accurately distinguish between passageways and obstacles in a falling wall, passing through openings wide enough to accommodate their bodies while avoiding openings that were too narrow. Walls consisting of two squares of width 20 separated by an aperture whose width was in the range $[20, 40]$ dropped from above with a vertical velocity of 3 and a horizontal offset of ± 25 relative to the agent. Accordingly, the performance measure to be maximized was:

$$f_B = \frac{1}{T}\sum_{i=1}^{T} p_i \tag{3}$$

where $p_i = 1 - d_i$ for an aperture wide enough for the agent to pass through and $p_i = d_i$ for an opening too narrow for the agent to pass through, d_i is the horizontal distance between the centers of the object and the agent when their vertical separation goes to zero on the ith trial (again clipped to MaxDistance and normalized to run between 0 and 1), T is the total number of trials, and D is the maximum distance. A total of 24 evaluation trials were used during evolution, uniformly distributed over the range of horizontal offsets.

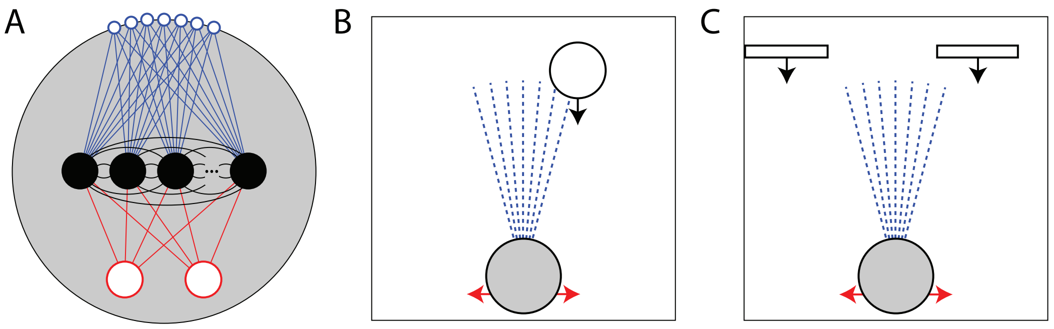


Figure 2: Agent and tasks set up. (**A**) Agent. Sensory neurons (blue) are interconnected to interneurons (black), then which are in turn connected to the two motor neurons (red). The interneurons are recurrently interconnected. The nervous system is bilaterally symmetric. (**B**) Object-size discrimination task. The agent moves horizontally while a circle of different size falls from above. The rays of the agent's proximity sensors are shown in dashed blue. (**C**) Perceiving affordances task. The agent moves horizontally while a wall with an adjustable aperture falls from above.

Statistical analyses of the neural activity, namely mutual information and transfer entropy, were estimated using the *infotheory* Candadai and Izquierdo (2019) package. Both metrics were estimated using an average shifted histogram based binning of the normalized activity with 100 bins along each dimension.

Part I: Generating a Multi-Functional Agent

In order to study the relationship between the statistical functional connectivity inferred from neural activity and the actual functional connectivity from lesion analysis, we first need an agent that is capable of performing multiple tasks. Thus, our first step was to generate an ensemble of successful multi-behavioral embodied dynamic recurrent neural systems. The agents were tasked with solving two minimally-cognitive tasks Beer (1996); Slocum et al. (2000): an object-size discrimination task and a perceiving affordances task. We performed 100 evolutionary runs with different random seeds (Fig. 3A). Agents are evolved to solve both tasks. During each fitness evaluation, an agent is tested first on the 24 evaluation trials of the object-size discrimination task and then on 24 evaluation trials of the perceiving affordances task. The performance is calculated for each task according to the defined fitness function (see Methods) and multiplied together to produce a score. After 1000 generations, many of the runs found successful configurations of the neural circuit that could solve the multiple tasks. A histogram of the final performance of each of the best agents in each run is shown in Fig. 3B. At least three of the runs achieved near-perfect performance on the fitness evaluation. As far as we are aware, this is the first report of agents successfully evolved for multiple minimally-cognitive tasks.

Before setting out to analyze one of these agents in some detail, it is important that we examine the generality and robustness of these solutions across a wider range of behavioral conditions. Our goal is to use this further examination to select which circuit to analyze in detail. We based our selection on which of the evolved agents solved both tasks equally well and also generalized well across a wider range of conditions for both tasks. There were three key changes with respect to the original fitness evaluation: (a) The step size of integration is made smaller (from 0.1 to 0.01). (b) The range of object sizes and aperture sizes was drawn from [20,40] in steps of 0.05 instead of 1. (c) The starting position was drawn from [1,5] in steps of 0.01. Altogether, this corresponds to 200000 trials for the generalization performance analysis (up from 100 trials per evaluation for the fitness function). In Figure 3C, we show the performance of each of the final circuits from each of the 100 evolutionary runs in each of the two tasks. As expected, the same three circuits that performed best in the fitness evaluation also generalized best across the wider range of conditions and are thus most appropriate for further examination. The best performing agent obtained a near-perfect performance 0.977 on the object-size discrimination task and 0.976 on the perceiving affordances task. We focus on this agent for the remained of this paper. What is the behavior and neural activity of this agent? In Figure 4, we show the behavior and neural activity for this best agent across the two different tasks. Traces are colored according to whether the agent has to catch or avoid the object or pass through the aperture or avoid it. We use these neural traces for the functional connectivity analysis ahead.

Part II: Comparing Functional Connectivities

Our second step is to analyze how well the insights gained from the statistically-inferred functional connectivity help us understand the actual functional connectivity of the circuit, as determined through informational lesion analysis.

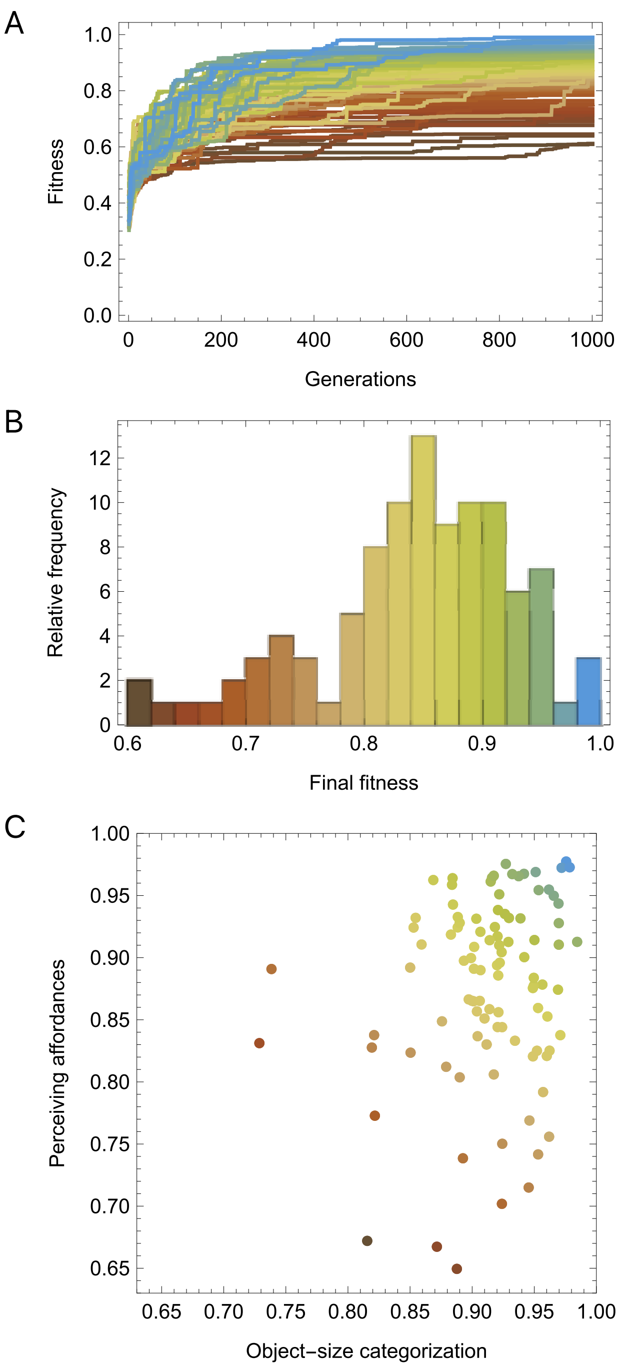


Figure 3: Generating a multifunctional agent using an evolutionary algorithm. (**A**) Best fitness over generations for hundred evolutionary runs, color coded by final fitness. (**B**) Relative frequency of the final fitness for each of the evolutionary runs. (**C**) Generalization performance for each of the final solutions on the two tasks: object-size discrimination and perceiving affordances.

In order to do this, we first characterize the node functional connectivity using traditional methods from network neuroscience (first column, Fig. 5). Specifically, we focus here on estimating the functional connectivity between pairs of neurons using Pearson's correlation, mutual information, and transfer entropy (Candadai and Izquierdo, 2019). We perform the analysis using the neural traces generated from each of the two tasks (Fig. 4B). From the magnitude of each of these measures, we establish the degree of involvement of each pair of neurons to the task. That is, pairs of neurons with little or no correlation in their activity during a task are deemed unlikely to be involved in that task; whereas pairs of neurons with strong correlation (either positive or negative) are deemed likely to be involved in the task.

Second, we characterize the two-way and one-way causal pairwise function of the edges using informational lesions (second column, Fig. 5). For the two-way informational lesions, we clamp the interchange of activity between two neurons in both directions, have the agent perform the task at hand, and measure the deficit in performance. We do this for a large range of potential basal outputs for each neuron in the pair (from an output of 0 to 1 in steps of 0.01 for each neuron), and we select the smallest deficit generated. The two-way lesions is used as the ground-truth for the nFC generated using Pearson's correlation and mutual information, because both provide only symmetric information between two neurons. For the nFC calculated using transfer entropy, which is directional, we perform a one-way information lesion analysis. The method here is the same as the previous one but it is perform for each individual connection in the circuit.

Finally, we compare the difference between the estimated involvement of each of the pairs calculated using nFCs to the ground-truth functional involvement characterized during the lesion studies for the two tasks (third column, Fig. 5). In these plots, each point represents a pair of neurons in the circuit and the different colors represent the different tasks, with the gray line connecting the same pair of neurons across different tasks. Points in the upper left corner represent unimportant connections for the task that are well captured by the statistical measurements. That is, connections between neurons that do not have much correlation and whose physical disruption does not cause a noticeable effect on task performance. Points in the bottom right corner represent important connections for the task that are also well captured by the statistical measurements. That is, connections between neurons that have a strong correlation and whose physical disruption causes a noticeable effect on task performance. Points in the upper right triangle represent points that are not causally important to the task, but that come up as important in the statistical measurements. Points in the bottom left triangle represent points that are causally important, but that do not show up as relevant in the statistical measures. The distance between the yellow and blue

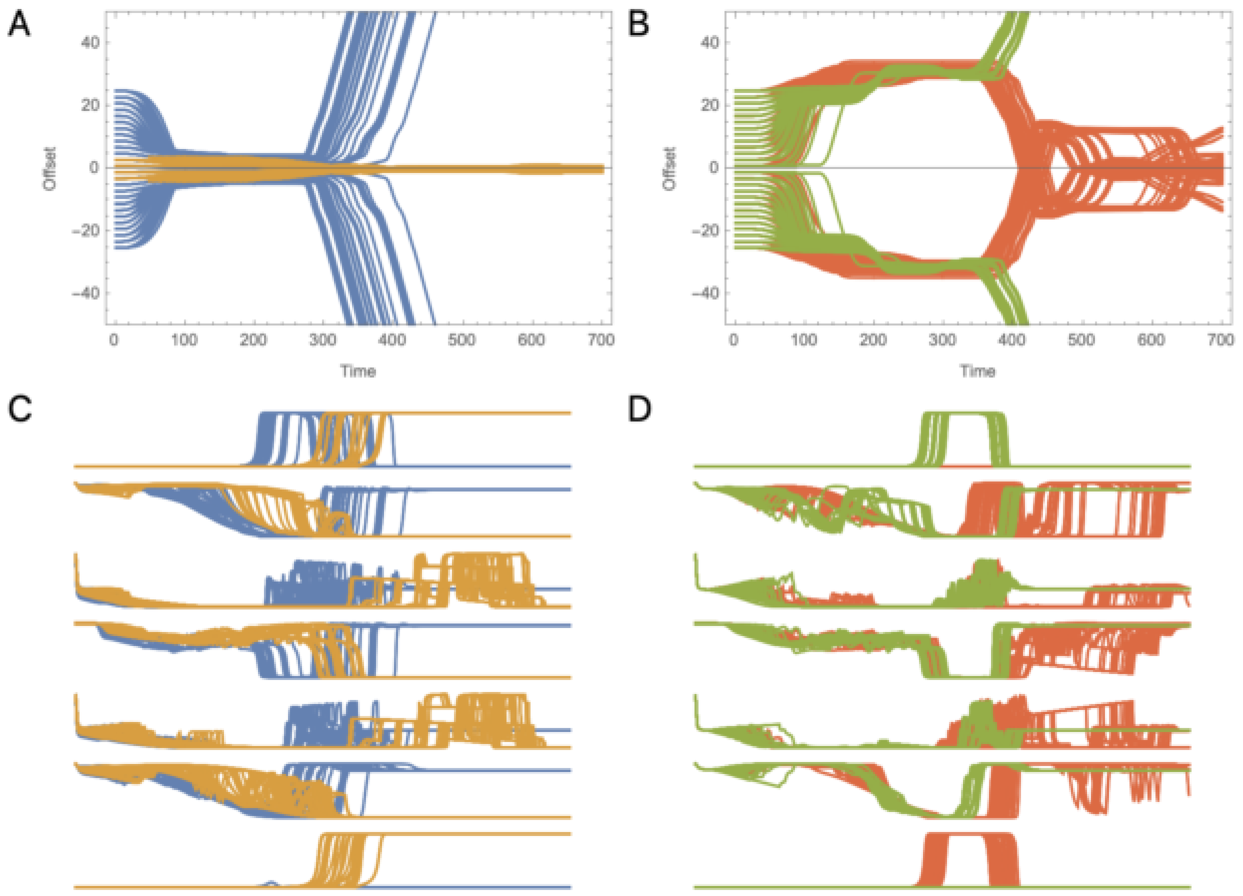


Figure 4: Behavior and neural activity across both tasks. (**A** and **B**) Behavioral traces during object-size discrimination and perceiving affordances tasks, respectively. Relative position of the agent in relation to the center of the falling objects over time. Traces are color coded depending on whether they need to be caught or avoided. In the object-size discrimination task, trials with the smaller circles are shown in yellow and traces with the larger circles are shown in blue. In the perceiving affordances task, trials with the smaller apertures are shown in green and those with the larger apertures are shown in red. (**C** and **D**) Neural activity from the seven recurrently interconnected interneurons driving behavior for each of the two tasks. Color coding follows the same pattern used for panels A and B.

points shown by the gray line represent the functional variation across tasks.

We highlight here four key qualitative insights gained from this analysis (Fig. 5). First, across all levels of analysis, the functional connectivities are different depending on the task being performed. This is true across all levels of analysis, including the informational lesion studies. This shows how the same nervous system, even without neuromodulation and synaptic plasticity, can have functionally different configurations, based on task engagement alone. This highlights the importance of studying nervous systems in the context of behavior. Second, the one-way lesion analysis reveals major differences in the directionality of interactions between components in the circuit. This, of course, highlights the inherent limitations of the two more established methods for estimating functional connectivity due to their symmetrical treatment of the relationships, Person's correlation and mutual information. Third, as can be appreciated by even a mere cursory look at how much darker the range of shades of the maps in the nFC column are in relation to those in the aFC column, all of the statistical measures over-infer the importance of the relationship between the variables relative to the actual role that those relationships play, as determined by the information lesions. As can be seen more easily in the third column, across all three measurements, lesions to the majority of connections have little or no effect on the behavior, despite the statistical measures inferring high levels of correlation, mutual information, and

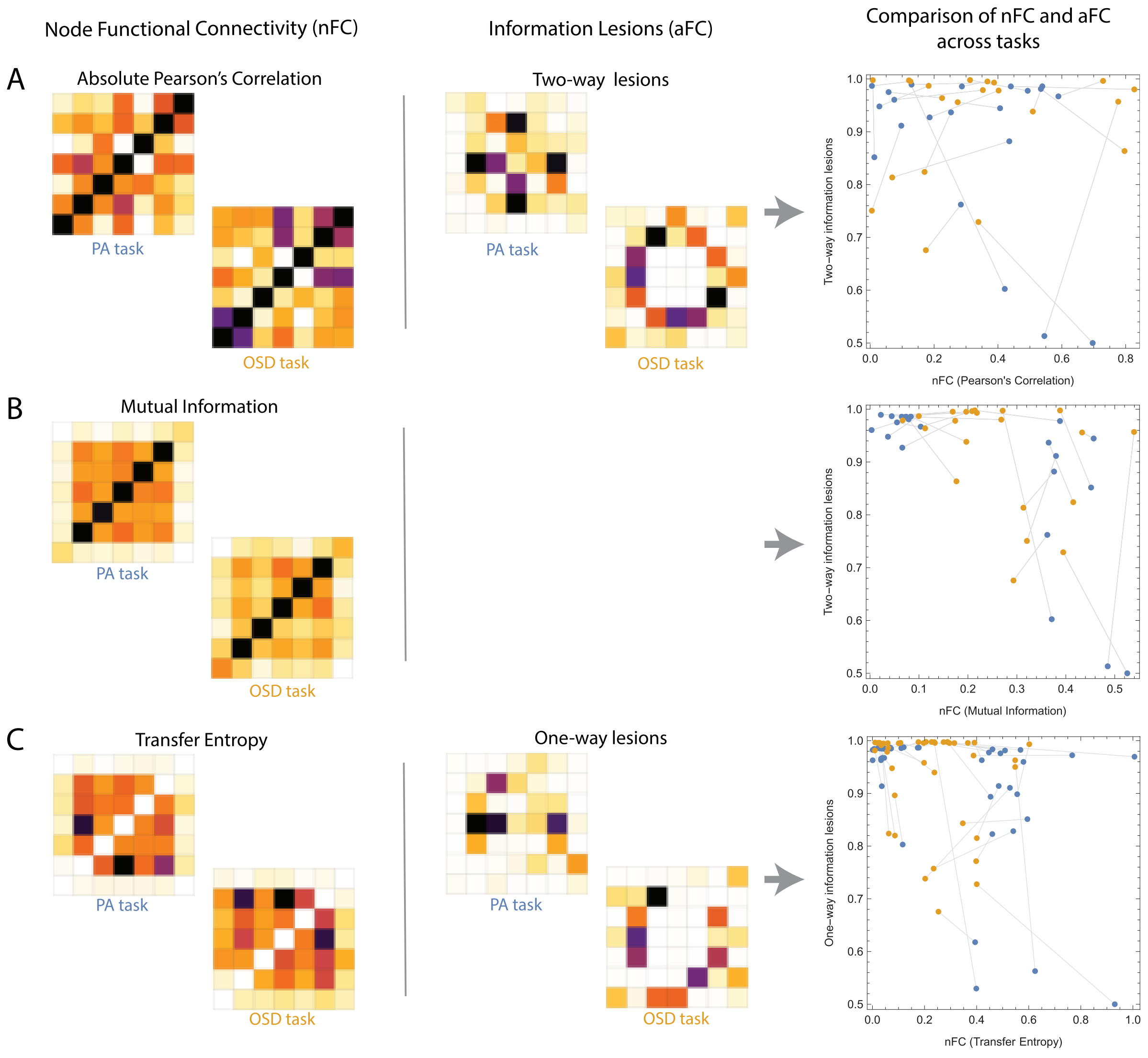


Figure 5: Comparing insights gained from the statistical measures of node functional connectivity based on neural activity to the ground-truth characterization from informational lesions analysis across tasks. (**A**) A comparison of the nFC using Pearson's correlation to the aFC using two-way lesions. (**B**) A comparison of the nFC using mutual information to the aFC using two-way lesions again. (**C**) A comparison of the nFC using transfer entropy to the aFC using one-way lesions. Across the first two columns, the matrix represents the pairwise interactions between the seven interneurons in the circuit. The shade of each cell represents the level of involvement of that pair in each task. The lighter the color represents little or no involvement, where white corresponds to no correlation, no mutual information, no transfer entropy, and no deficit in performance after a two-way lesion, or a one-way lesion, respectively. The darker shades represent increasing involvement, where black corresponds to the maximum level of correlation, mutual information, and transfer entropy for the statistical measurements, and a complete disruption in the performance of the task in the case of the two-way and one-way lesions. The last column depicts the comparison between statistical inference (nFC) and ground-truth (aFC). Each point represents a pair of neurons in the circuit (blue for the perceiving affordances task and yellow for the object-size discrimination task). The gray line connecting two dots represents the same pair of neurons across the different tasks. On the y-axis, the two-way and one-way lesions are defined such that the value represents the deficit in performance. So a performance of 0.95 after a lesion to the connection between neurons X and Y means that this link was not very important to the functioning of the circuit; on the other hand a resulting performance of 0.5 after a lesion would indicate that the connection between those pair of neurons is extremely important to the functioning of the circuit. See main text for interpretation of the results.

transfer entropy between them. Although this is expected to some degree, the full effect can be quantified here concretely. Fourth, despite the tendency for statistical measures to over-infer causality that we just discussed, our analysis of this agent allows us to see some clear examples of connections that are causally important that do not show up in the statistical measurements. These are, of course, much less common. They are particularly salient in the Pearson's correlational analysis and in the transfer entropy analysis.

Discussion

In this paper, we set out to answer the question: What does functional connectivity tell us about the behaviorally functional connectivity of a multifunctional neural circuit? We used a computational neuroethology approach to begin to address this theoretical challenge. We evolved a dynamical recurrent neural network to be capable of performing multiple tasks, and then we analyzed its neural activity using traditional network neuroscience tools. While our analyses were performed on a neuron-to-neuron basis and functional connectivity is typically performed across brain regions, CTRNNs are universal function approximators and can model neural activity in brain regions thus enabling our analyses to scale. We then compared the results against a series of informational lesions as a way to reveal their degree of approximation to the ground-truth. Overall, our analysis reveals a large gap between the insights gained from statistical inference of the functionality of the circuits based on neural activity and the actual functionality of the circuits as revealed by mechanistic interventions.

It is important to note that the measures of functional connectivity being investigated in this paper are measures of a statistical relationship. They are neither measures of causal effect nor of how such a statistical relationship might relate to interventional impact on a task. However, these statistical methods are often used as tools to make claims about the relationship between neural activity and behavior. The goal of this paper is to examine the degree to which those measures of a statistical relationship estimate causal effect on behavior, as determined in this case through the interventional impact on task function. While there has been some work that had attempted to answer this question in the past (Ay and Polani, 2008; Lizier and Prokopenko, 2010; Chicharro and Ledberg, 2012), that work considered only neural circuits in a vacuum; here we extend this work to consider functional and complete brain-body-environment systems.

It is relatively straightforward to understand why a connection between two neurons may have a high correlation, a high mutual information, or a high transfer entropy, and yet not have a high interventional causality: The two neurons can be correlated for reasons other than their connection to each other. The opposite is also straightforward to understand (i.e., why a connection between two neurons may have a low correlation, low mutual information, or low transfer entropy and nevertheless have a high interventional causality): The transformation of information between the two neurons may be such that the two neurons do not have similar informational profiles and yet they are still causally linked. Finally, it is only to the degree that the two neurons are causally linked, that lesioning the connection will have some effect on functional performance. In other words, a causal link is necessary for function, but not sufficient. There may be some connections that are causally linked, but that do not contribute to the circuit's function. Finally, it is important to note that all analyses in this work was done on dyads and the statistical and interventional methods alike would benefit from polyadic analysis.

Future Work

We have three main directions of future work. (1) In this paper, we deliberately focus our analysis on a single circuit as a way to begin to gather intuition. One direction for future work is to perform a similar analysis across an ensemble of successful but different solutions, as a way to uncover the more general principles. This will involve examining methods for quantifying how close the different statistical methods approximate the causal relationships. (2) The neural network under consideration in our current analysis was fully recurrent. We would like to study the degree to which the structural connectivity of the neural circuit affects the usefulness of the functional connectivity. One direction for future work will be to systematically study the effect that the structure of the connectivity of the circuit has on the performance of the statistical measures of functional inference. As part of this analysis, we also plan to study structures that are grounded in available connectomes. (3) Finally, we deliberately focused our analysis on the most popular measures of node functional connectivity. One additional direction for future work is to study the wider range of measurements used in the network neuroscience literature. Crucially, we would like to use further refined versions of the computational neuroethology approach proposed here as an ideal testground for generating novel variations of statistical measurements to gain insights on the functional connectivity of these complex neural circuits.

Acknowledgements

We are thankful for the reviewers' comments. This material is based upon work supported by the National Science Foundation under Grant No. 1845322.

References

Aimon, S., Katsuki, T., Jia, T., Grosenick, L., Broxton, M., Deisseroth, K., Sejnowski, T. J., and Greenspan, R. J. (2019). Fast near-whole–brain imaging in adult drosophila during responses to stimuli and behavior. *PLoS biology*, 17(2):e2006732.

Ay, N. and Polani, D. (2008). Information flows in causal networks. *Advances in complex systems*, 11(01):17–41.

Beer, R. (1996). Toward the evolution of dynamical neural networks for minimally cognitive behavior. *From animals to animats 4: Proceedings of the Fourth International Conference on Simulation of Adaptive Behavior*, pages 421–429.

Candadai, M. (2021). Information theoretic analysis of computational models as a tool to understand the neural basis of behaviors. *arXiv preprint arXiv:2106.05186*.

Candadai, M. and Izquierdo, E. J. (2019). infotheory: A c++/python package for multivariate information theoretic analysis. *arXiv preprint arXiv:1907.02339*.

Candadai, M. and Izquierdo, E. J. (2020). Sources of predictive information in dynamical neural networks. *Scientific reports*, 10(1):1–12.

Chicharro, D. and Ledberg, A. (2012). When two become one: the limits of causality analysis of brain dynamics. *PloS one*, 7(3):e32466.

Datta, S. R., Anderson, D. J., Branson, K., Perona, P., and Leifer, A. (2019). Computational neuroethology: a call to action. *Neuron*, 104(1):11–24.

Dayan, P., Abbott, L. F., et al. (2003). Theoretical neuroscience: computational and mathematical modeling of neural systems. *Journal of Cognitive Neuroscience*, 15(1):154–155.

Fornito, A., Zalesky, A., and Bullmore, E. (2016). *Fundamentals of brain network analysis*. Academic Press.

Friston, K. J. (1994). Functional and effective connectivity in neuroimaging: a synthesis. *Human brain mapping*, 2(1-2):56–78.

Ito, S., Hansen, M. E., Heiland, R., Lumsdaine, A., Litke, A. M., and Beggs, J. M. (2011). Extending transfer entropy improves identification of effective connectivity in a spiking cortical network model. *PloS one*, 6(11):e27431.

James, R. G. and Crutchfield, J. P. (2017). Multivariate dependence beyond shannon information. *Entropy*, 19(10):531.

Lin, A., Witvliet, D., Hernandez-Nunez, L., Linderman, S. W., Samuel, A. D., and Venkatachalam, V. (2022). Imaging whole-brain activity to understand behaviour. *Nature Reviews Physics*, pages 1–14.

Lizier, J. T. and Prokopenko, M. (2010). Differentiating information transfer and causal effect. *The European Physical Journal B*, 73(4):605–615.

Maheswaranathan, N., Williams, A., Golub, M., Ganguli, S., and Sussillo, D. (2019). Universality and individuality in neural dynamics across large populations of recurrent networks. *Advances in neural information processing systems*, 32.

Nguyen, J. P., Shipley, F. B., Linder, A. N., Plummer, G. S., Liu, M., Setru, S. U., Shaevitz, J. W., and Leifer, A. M. (2016). Whole-brain calcium imaging with cellular resolution in freely behaving caenorhabditis elegans. *Proceedings of the National Academy of Sciences*, 113(8):E1074–E1081.

Paninski, L. and Cunningham, J. P. (2018). Neural data science: accelerating the experiment-analysis-theory cycle in large-scale neuroscience. *Current opinion in neurobiology*, 50:232–241.

Ramaswamy, V. (2019). An algorithmic barrier to neural circuit understanding. *BioRxiv*, page 639724.

Randlett, O., Wee, C. L., Naumann, E. A., Nnaemeka, O., Schoppik, D., Fitzgerald, J. E., Portugues, R., Lacoste, A. M., Riegler, C., Engert, F., et al. (2015). Whole-brain activity mapping onto a zebrafish brain atlas. *Nature methods*, 12(11):1039–1046.

Slocum, A., Downey, D., and Beer, R. (2000). Further experiments in the evolution of minimally cognitive behavior: From perceiving affordances to selective attention. *From Animals to Animats 6: Proceedings of the Sixth International Conference on Simulation of Adaptive Behavior*, pages 430–439.

Smith, S. M., Miller, K. L., Salimi-Khorshidi, G., Webster, M., Beckmann, C. F., Nichols, T. E., Ramsey, J. D., and Woolrich, M. W. (2011). Network modelling methods for fmri. *Neuroimage*, 54(2):875–891.

Sporns, O. (2010). *Networks of the Brain*. MIT press.

Stevenson, I. H., Rebesco, J. M., Miller, L. E., and Körding, K. P. (2008). Inferring functional connections between neurons. *Current opinion in neurobiology*, 18(6):582–588.

Van Den Heuvel, M. P. and Pol, H. E. H. (2010). Exploring the brain network: a review on resting-state fmri functional connectivity. *European neuropsychopharmacology*, 20(8):519–534.

Yeo, B. T., Krienen, F. M., Sepulcre, J., Sabuncu, M. R., Lashkari, D., Hollinshead, M., Roffman, J. L., Smoller, J. W., Zöllei, L., Polimeni, J. R., et al. (2011). The organization of the human cerebral cortex estimated by intrinsic functional connectivity. *Journal of neurophysiology*.

Bottom-up formation of number representation and top-down understanding of symbolic manipulation

Yasuhiro Shimada[1] , Wataru Noguchi[2], Hiroyuki Iizuka[2,3], and Masahito Yamamoto[2,3]

[1] Graduate School of Information Science and Technology, Hokkaido University, Japan
[2] Faculty of Information Science and Technology, Hokkaido University, Japan
[3] Center for Human Nature, Artificial Intelligence, and Neuroscience, Hokkaido University, Japan
y.shimada@ist.hokudai.ac.jp

Abstract

Several studies that deal with the acquisition of concepts in a bottom-up manner from experiences in the physical space exist, but there are few of them that deal with the bidirectional interaction between symbolic operations and experiences in the physical world. It was shown that a shared module neural network succeeded in generating a bottom-up spatial representation of the external world, without involving learning of the signals of the spatial structure. Furthermore, the module can understand the external map as a symbol based on its spatial representation, and top-down navigation can be performed using the map. In this study, we extended this model and proposed a simulation model that unifies the emergence of a number representation, learning of symbol manipulation on the representation, and top-down understanding of symbol manipulation onto the physical world. Our results show that the learning results of the symbol manipulation can be applied to the physical world prediction, and our proposed model succeeded in grounding symbol manipulation onto physical experiences.

Introduction

Humans typically quickly learn the concept of numbers and can recognize different numbers of objects. An apple is perceived as a different object from an orange, yet we know that both are instances of the same number. Similarly, three watermelons may be expected to occupy a larger proportion of our visual field than five oranges, but we can distinguish the larger number of oranges compared to the watermelons. Once we obtain the concept of numbers, this abstract understanding allows us to understand the idea of 1,000 apples, although we may never have actually perceived, counted, or compared such a large amount. This indicates that our concept of numbers develops through experiences in a physical world in a bottom-up manner, and then we learn to manipulate concepts as symbols in an abstract (or mental) world with some set of conceptual rules. Thus, we can understand the results of operations in the abstract world by linking them to real-world phenomena in a top-down manner.

The idea of intelligence based on symbol manipulation without grounding in the physical world has been criticized by Searle and Harnad via the Chinese room argument and the symbol grounding problem (Searle, 1980; Harnad, 1990). With the development of neural networks and robotics, the importance of physicality or embodiment as a foundation for intelligence as situated in the physical world has been emphasized (Pfeifer and Scheier, 2001; Brooks, 1991; Harvey et al., 1997; Nolfi and Floreano, 2000). The idea of embodiment considers that an autonomous entity obtains concepts emergently from interactions with their environment in a bottom-up manner, without designing clear abstract symbols in advance (Marocco et al., 2003; Beer, 2000). However, these embodiment models remain limited to the bottom-up process of concept generation. No models have been proposed to integrate bottom-up concept generation, the symbolic manipulation of concepts generated by physical interaction, and top-down processes for understanding the results of symbolic manipulation in the physical world.

In contrast, a shared module neural network has been proposed to perform sensory integration and to differentiate self from others (Noguchi et al., 2022a,b). In this model, a single shared module processes sensor inputs from different modalities in the same manner. This constraint of sharing the module among different modalities was used to generate a bottom-up spatial representation of the external world, which was common to all modalities. Furthermore, the shared module neural network could understand the external map as a symbol based on its spatial representation and perform top-down navigation using the map. However, the authors considered continuous space rather than discrete abstract symbols, and did not consider explicit symbolic operations such as addition and subtraction.

Therefore, in this study, we extend the previous model and propose a simulation model that unifies the emergence of a representation of numbers, learning symbol manipulation on the representation, and top-down understanding of symbol manipulation onto the physical world. The remainder of this paper consists of three sections, discussing respectively the emergence of number representation, learning of symbol manipulation processes, and the top-down understanding of symbol manipulation. The overall simulation models

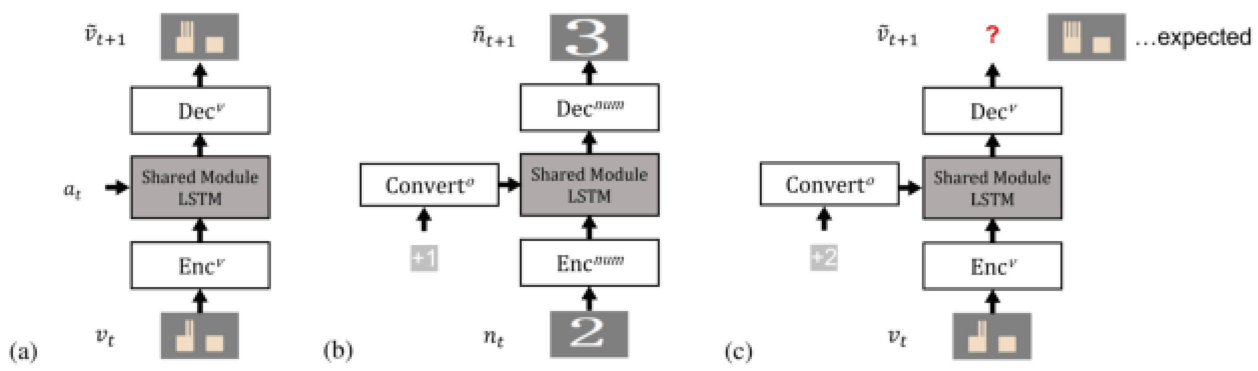


Figure 1: Our proposing shared module neural network models. (a) A basic model used to predict visual inputs in the physical world experiment. The visual images include stretching and bending a finger and placing and removing an object. (b) Re-use of the shared module for the experiment of symbolic number prediction. Enc^{num} and Dec^{num} are new networks and a operation converter is added for symbolic operations (See the main text for the details). (c) Applying the model obtained in (b) to the physical situations of finger counting.

the process by which children acquire numbers using their fingers and objects and learn new symbolic operations with numbers. Finally, we show that the learning results can be applied to the physical world.

Bottom-up formation of number representation

In this section, we simulate the formation of a representation of numbers through visual predictive learning. To realize this, we trained a neural network model to predict images changing as a result of action. We simulated a humanoid robot as a model of a child performing actions of bending and stretching its finger, and placing and removing objects. The neural network model was trained by predictive learning on the robot's physical experiences. This simulation is referred to as a physical world experiment in the sense that the robot interacts with a simulated physical environment to obtain simulated sensory information.

Model

Figure 1 (a) shows our proposed shared module neural network. The model takes the current vision image v_t and the current action a_t as its proprioceptive inputs. Then, it outputs the prediction $\tilde{v}_{t+1}$ of the next vision input v_{t+1}. The model consists of three main components, including an encoder Enc^v, decoder Dec^v, and a shared module with an LSTM. The encoder and decoder are implemented with feed-forward neural networks and the LSTM is used for the shared module, which is re-used for different tasks. The model performs prediction on consecutive sequences of visual and motor inputs. To encourage the model to perform prediction across time steps on the sequences, we introduce a mask operation to block the outputs of the encoder, which are the inputs to the LSTM. The mask operation replaces the output of the Enc^v with the zero vector, and it causes the network to perform prediction using mainly a_t, without relying strongly on v_t. It has been shown that this masking operation promotes the structuring of internal states of LSTM trained by predictive learning (Noguchi et al., 2017; Banino et al., 2018). The visual input is first encoded by the encoder. Next, the LSTM receives the encoded vector and action. Finally, visual prediction is generated from the outputs of the LSTM by the visual decoder. The following equations formulate this prediction process.

$$enc_t^v \quad = \quad \mathrm{Enc}^v(v_t), \tag{1}$$

$$(h_t^v, c_t^v) = \mathrm{LSTM}(\mathrm{Mask}(enc_t^v; p^{mask}), a_t, h_{t-1}^v, c_{t-1}^v), \tag{2}$$

$$\tilde{v}_{t+1} = \mathrm{Dec}^v(h_t^v), \tag{3}$$

where h_t^v and c_t^v represent hidden and cell states of LSTM at time t, respectively. Mask denotes the mask operation and the p^{mask} is the probability of masking.

Robot's vision and action

A robot which has two hands with five fingers was simulated to obtain data on physical experiences. The robot performs actions in the environment to move its fingers, bending or stretching them. The robot also performs a task of placing or removing objects in front of it. It can observe its own hands and objects by means of a camera. The actions performed by the robots are simulated in an extremely simplified form. One action involves stretching a single finger and placing a single object, and another bending a single finger and removing a single object. The current action, a_t, is represented as 2 dimensional vectors, $[x, 0]$ or $[0, x]$, ($x \in X \ where \ X = \{x \in \mathbb{R} \mid 0.8 \leq x \leq 1.0\}$). $[x, 0]$ is used for stretching a finger and placing an object, and $[0, x]$

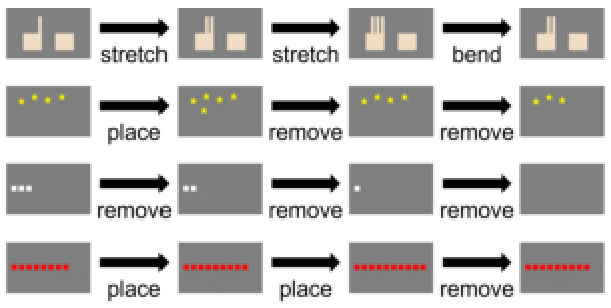


Figure 2: Example of training datasets of sequences of visual images. There are 4 different kinds of images, fingers, stars, white and red balls.

represents bending a finger and removing an object. The intermediate states of the actions are not simulated; the resulting states of the finger or object are obtained in a single time step. Specifically, the interaction between objects and hands, such as grasping, was not simulated. Although validation in a more complex simulation environment with continuous actions would be necessary, since the primary goal of this study is to show that the shared module can be used as a basic principle to link bottom-up formation of representations and top-down understanding of symbols, our simulations are kept extremely simple in the subsequent experiments

We collected sequences of vision and action from the perspective of the simulated robot. Four types of sequences correspond to the robot's physical experiences of its hands and three kinds of objects: stars, white balls, and red balls. The robot performs a series of actions and records vision data (Fig. 2). We considered eleven states for each type of sequence corresponding to the number of stretched fingers and the number of placed objects. The vision of the simulated robot was obtained as RGB images with a size of 48×80.

Experiment

The proposed LSTM model had 128 hidden units. Enc^v and Dec^v were composed of convolution and fully connected layers. 300 sequences of vision and action were used to train the model, and 200 sequences were collected for further analysis. The length of a single sequence was 40 images. During the collection, the actions of the robot were controlled as follows. First, an action was selected randomly from possible actions. Then, the robot repeated the selected same action for a randomly determined duration. When it finished this repetition, the action was selected randomly again and the robot began repeating that action. When the hand or object states reached the terminal state, corresponding to zero or ten, the robot changed its current action to another action with a new randomly determined repeating duration. The model was trained over 160 epochs. The parameters of the model were updated with a variable learning rate (0.001 in the initial 100 epochs, and 0.0001 in the remaining), with a batch size of 8 and $p_{mask} = 0.5$. For predictive learning, prediction loss L^{object}_{pred} was calculated as a mean squared error (MSE) between a predicted vision and the actual next vision as follows.

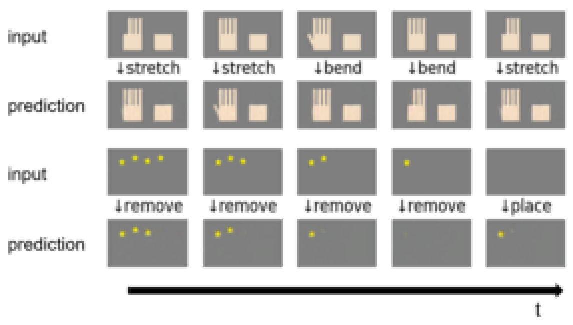


Figure 3: Example of the results of predictive learning. The rows of input show the current states of hands at a given time, and the row of prediction shows the predicted visual images from the input.

$$L^{object}_{pred} = \frac{1}{T}\sum_{t=1}^{T} \mathrm{MSE}(\tilde{v}_{t+1}, v_{t+1}), \tag{4}$$

where T is the length of the sequence of vision and action.

Results

Figure 3 shows the prediction results by the trained network. It may be observed that the neural network model can successfully predict visual images according to the current states and actions (stretching and bending a finger, and placing and removing an object). We only show the results of the hand and star, but the results of other objects were also successful. Note that if p^{mask} is set too high, learning dose not success.

Next, we visualized the internal states of the trained shared module LSTM, for which, we used a principal component analysis (PCA) to reduce the internal states to two dimensions. Figure 4 shows the internal states of the shared LSTM module. The color corresponds to the number represented as the target states of the prediction. In the visualized internal state space, we can determine the internal states organized from two aspects. The first is that the states corresponding to the same objects align linearly in the space and the corresponding numbers increase or decrease along the aligned direction. The other is that the alignments are shared among different objects. Therefore, we call to this state space as a "number representation."

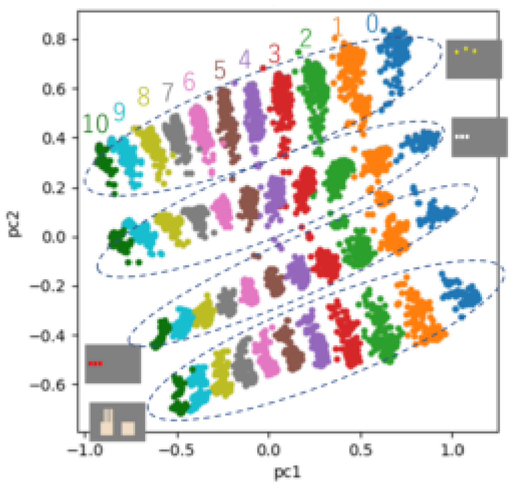


Figure 4: Visualized internal states of the shared module LSTM, using PCA to reduce the dimensionality to two dimensions. The color corresponds to the number represented by the prediction output of the model. The four clusters correspond to four types of objects.

Associating symbolic number with number representation and learning of symbol manipulation

In this section, we describe how abstract symbol manipulations were learned on the neural network model trained in the previous section. This simulates that the network model learned the symbol manipulations using the number representation obtained from the physical experiences. To realize this process, we re-trained the neural network model with the shared module LSTM to predict the outcomes of the symbolic operations. In the simulation, the robot receives an image of symbolic numbers shown in Fig. 5 and operations as symbol manipulations. The network model predicts the image of the next number as the outcome of the operations. In the same way as the previous task, the network model is simply trained by predictive learning.

Model using shared module

Figure 1 (b) shows our model, which receives the images of the current number and the operation as its inputs in a similar manner with the physical world experiments. Then, it outputs a prediction of the next number as the result of the operation. It differs in two ways from the previous model. First, the shared module LSTM trained in the previous task is used, but those parameters are frozen during learning. The encoder and decoder networks are new networks initialized randomly, and the parameters of these networks are updated by predictive learning. Second, a operation converter is added to the model to convert operations into inputs of shared module LSTM. It is required because the proprioceptive inputs as physical actions are used for the shared module described in the previous section, but there are no corresponding physical actions for this task. Hence, we call the output of operation converter virtual action. We considered two kinds of symbolic operations: + 1 and -1. In the simulation model, random values are assigned to +1 and -1 operations as inputs for the operation converter, which are not relevant to the physical actions used in the previous task. Notably, we use +1 and -1 to express two types of symbolic operations, but because those operations are input to the network as randomly assigned values, they do not have the literal meaning of +1 and -1 except for the two different operations. The operation converter $\mathrm{Convert}^o$ changes the inputs into virtual action vectors with the same number of dimensions as the physical action as follows.

$$a_t^{virtual} = \mathrm{Convert}^o(o_t), \tag{5}$$

where o_t is assigned random values for the symbolic process proprioceptive inputs +1 and -1. Because the shared module network receives and operates without distinction between physical and virtual action, the virtual action output by the operation converter can be interpreted and used as physical action.

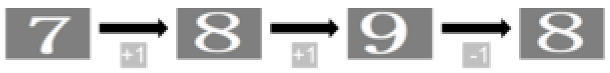


Figure 5: Example of visual image of number changes in response to the operations, +1 and -1.

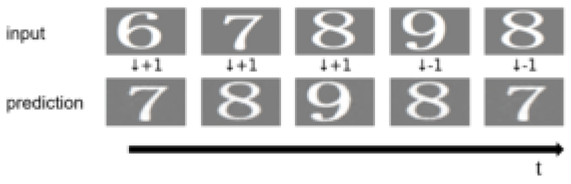


Figure 6: Prediction results of images of symbolic numbers with operations +1 and -1.

Simulation of symbolic operation

We consider the situation in which the number is displayed as visual inputs in front of the robot. The number changes sequentially in response to the given symbolic operations, +1 or -1. Similarly to the explanation above, we collected sequential datasets composed of numbers and symbolic operations (Fig. 5). The numbers ranged from 0 to 10. We used RGB images with a size of 48×80. The symbolic operations ($+1$ and -1) were represented as the three-dimensional random vector.

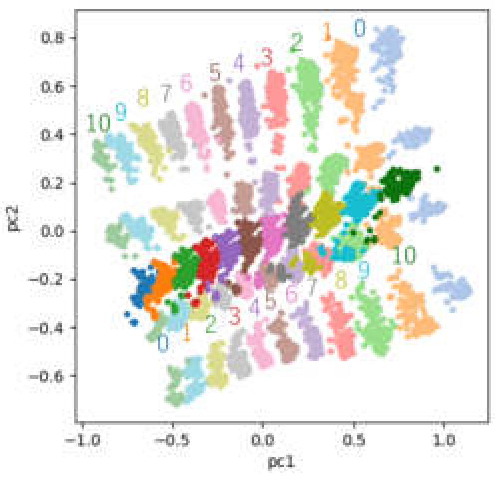


Figure 7: Internal states of the shared module trained in the experiment of symbolic number prediction (dark dots). The light dots indicate the number representation obtained in the previous physical world experiment. The correspondence of numbers are in reverse order.

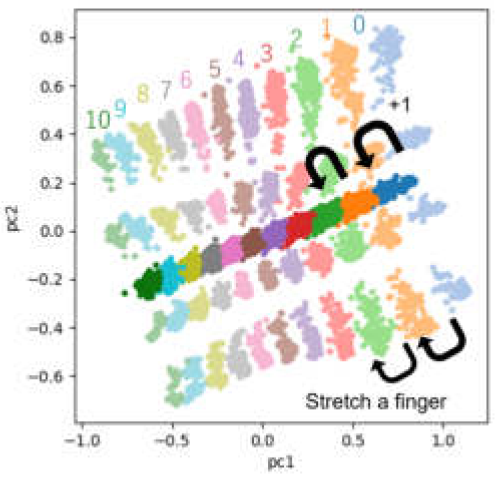


Figure 8: Internal states of the shared module trained with correspondence loss (eq. (7)).

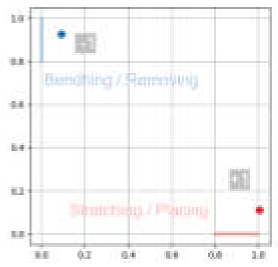


Figure 9: Virtual actions converted from the symbolic operations +1 and -1 by Convert^o. The lines show the physical actions used in the experiment with fingers and objects.

Experiment

Enc^{num} and Dec^{num} had the same structure (but not the same parameters) as Enc^v and Dec^v. Convert^o were composed of fully connected layers. 300 sequences of vision and action were collected to train the model, and the length of the sequence was 40. The model was trained for 40 epochs. The parameters of the Enc^{num}, Dec^{num}, and Convert^o were updated with variable learning rates (0.002 in the initial 20 epochs, and 0.0002 in the last 20 epochs), and a batch size was 8 and $p^{mask} = 0.8$. For predictive learning, the prediction loss L^{num}_{pred} was calculated as the MSE between the images of predicted number and the next number as follows.

$$L^{num}_{pred} = \frac{1}{T}\sum_{t=1}^{T} \text{MSE}(\bar{n}_{t+1}, n_{t+1}). \tag{6}$$

The network was trained to minimize L^{num}_{pred}. Note that no loss was given to directly supervise actions.

Results

Figure 6 shows the results of learning on symbolic operations, +1 and -1. It shows that the model was able to predict the next image of the number according to the current number and the operation. Figure 7 shows the internal states of the shared module, which were mapped onto the PCA spaces obtained in Fig. 4. The dark and light dots indicate the internal states obtained in this and the previous experiments, respectively. [1] It may be observed that the internal states for the symbolic operations were in line with the representations formed in the previous experiment. However, the correspondence of the numbers was in reverse order. This is because there was no relationship between the numbers in the physical world and the numbers in the symbol manipulation process. It is important to note here that reversal or lack thereof is determined by chance. Sometimes they happen to be the same.

Here, we assumed that the supervised signal is necessary to correspond the symbolic number to the number of objects,

[1]Note that too high p^{mask} causes learning to fail. The reason we increase p^{mask} is to make correspondence better. If we use low learning rate, the distribution of the dark dots was smaller than that of the light dots.

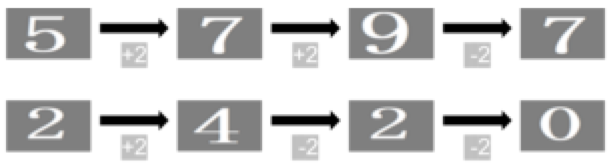


Figure 10: Example of training sequences of visual images of symbolic numbers and operations +2 and -2.

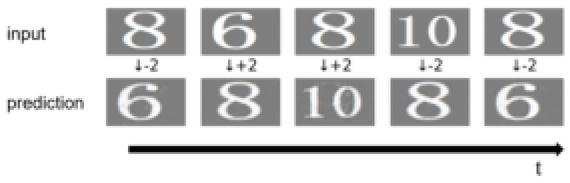


Figure 11: Prediction results on symbolic numbers with +2 and -2 operations.

so we used an additional loss as follows.

$$L_{corr} = \frac{1}{T}\sum_{t=1}^{T} \text{MSE}(enc_t^v, enc_t^{num}). \quad (7)$$

where enc_t^v and enc_t^{num} are the output of encoder Enc^v and Enc^{num}, respectively. Then, the overall loss was as follows.

$$L_{all} = L_{pred}^{num} + L_{corr}. \quad (8)$$

The internal states after adding L_{corr} are shown in Fig. 8. In this figure, the opposite correspondence has been eliminated, and the numbers thus overlap in an orderly fashion. For example, the visual image of 10 as a symbol corresponds to the situation in which 10 fingers or 10 objects exist in front of the robot. In other words, the structure of the number representations is re-used to represent the states of the symbolic numbers. This fact also implies that the symbolic operations +1 and -1, are likely to be shared with the physical action. This occurs because if the action of stretching one finger does not correspond to the symbolic operation +1, it becomes difficult to transition the internal states in the same way between physical experiences and symbolic manipulation. In fact, it is observed that Convert^o output similar values to the actions of increasing and decreasing in the first experiment following the predictive learning process (Fig. 9). The results of this experiment demonstrate that predictive learning with the shared module neural network successfully grounded the virtual action to match the physical experience.

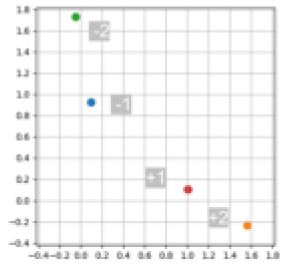


Figure 12: Obtained virtual actions corresponding to $+2$ and -2.

Top-down understanding of symbolic operation

In the previous section, we confirmed the correspondence between physical experiments and symbolic manipulation in terms of internal states and actions. Physical and virtual actions were found to be interdependent, and the virtual action was learned to match the physical action. On the contrary, it must be possible to generate physical action in a top-down manner by learning a virtual action. Such a top-down understanding is shown in this section.

Learning on new symbolic operations to generate virtual actions

Our model learned new symbolic operations +2 and -2, for which there were no corresponding physical actions. The new random values were assigned to the symbolic operations +2 and -2 as inputs to the operation converter. The model (Fig. 1(b)) trained in the previous section was used here. The model predicted on physical experiences in the same manner as described in the previous section. However, only the parameters of Convert^o were trained using only L_{pred}^{num}, whereas the parameters of Enc^{num}, Dec^{num} and the shared module were frozen. For inputs, sequential datasets comprising numbers and the new symbolic operations were collected as explained in the previous section (Fig. 10).

The model was trained for 20 epochs. The parameters of Convert^o were updated with variable learning rates (0.002 in the first 10 epochs, and 0.0002 in the last 10 epochs), with a batch size of 8 and $p^{mask} = 0.8$. The hyper parameters follows those of the previous experiment except for the number of training epochs.

Results

Figure 11 shows the prediction results of learning on new symbolic operations, +2 and -2. As in the previous task, the network model could successfully predict changes in the numbers simply by learning the weights of the operation converter. The virtual actions from the operation converter for the new symbol manipulation are shown in Fig. 12. The

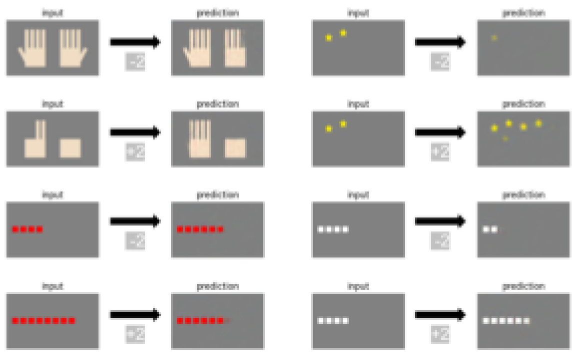


Figure 13: Example of prediction results. Virtual action can be used to represent physical actions like stretching two fingers.

virtual actions naturally differed from the virtual actions of +1 and -1, but +2 was represented on the same side as +1, and -2 existed on the other side of +1 and +2. The result indicates that the concepts of plus and minus and the amount of the numbers are represented in the virtual actions.

After obtaining virtual actions through predictive learning, we investigated whether the model could use them as actions in the physical space. Then, we replaced the encoder and decoder with those trained for the experience in the physical space (Fig. 1 (c)). The model received a current visual image of fingers or objects and the operation (+2 or -2), and then predicted the next vision.

Figure 13 shows the prediction results when virtual action was input to the neural network model as physical action. The results indicate that after associating the symbolic numbers with experiences in physical space, the model could understand the new symbolic operation learned only in symbolic space as a new action in physical space.

Discussion

Besides using the shared module as in the current study, there are also models that acquire number representations through learning. For example, there is a study showing that the activity of convolutional neural networks (CNNs) trained to classify natural images is correlated with the number of objects in the image, and that the CNNs acquire an internal representation corresponding to the number of objects (Nasr et al., 2019). However, in that case, the CNNs cannot manipulate the internal representations because they only receive image inputs and passively evoke internal representations. In contrast, our model can manipulate the internal number representation using both physical and virtual action. Another point that makes our model different from simple CNN classifier models is that the symbolic numbers do not correspond to the number of visual objects in the images. The shared module allows the association between the visually uncountable numbers in the symbolic space and visually countable objects in the physical space.

The current simulation does not take into account the concept of order of digits in numbers. For example, the symbolic number 10 has the symbol 1 at the ten's place and the symbol 0 at the one's place; the number 10 is represented by a combination of the two different symbols as units. On the other hand, the model in this study recognizes the symbolic number 10 as a single isolated symbol. This is because the model acquires the structure of the number representation space based on experiences in the physical space, and the structure of order in the symbolic number is not explicitly observable in the physical space and only exists in the symbolic space. To handle such higher-order structure of the symbol like the combination of numbers, it is necessary to learn the structure of the symbolic number space through learning in the symbolic space as same as the learning of new operations through learning in the symbolic space shown in the current study.

Conclusion

This study proposed the shared module neural network, which can ground symbolic numbers onto the internal representations evoked by the physical interactions. Thanks to the use of the shared module, it becomes possible to understand symbolic operations that manipulate symbolic numbers as physical actions. Furthermore, after the symbolic numbers and operations are grounded, learning new operations on symbols allows the shared module neural network to predict the consequences in the physical world when the corresponding physical action is performed. We believe that a mechanism like the shared module may exist in our brain to connect abstract thought with physical experience. However, the simulation in this research was very simplified. For future work, it is necessary to investigate whether the same results can be reproduced with a more complex simulation.

Acknowledgements

This work was partially supported by JSPS KAKENHI Grant Number JP20H04989.

References

Banino, A., Barry, C., Uria, B., Blundell, C., Lillicrap, T., Mirowski, P., Pritzel, A., Chadwick, M. J., Degris, T., Modayil, J., Wayne, G., Soyer, H., Viola, F., Zhang, B., Goroshin, R., Rabinowitz, N., Pascanu, R., Beattie, C., Petersen, S., Sadik, A., Gaffney, S., King, H., Kavukcuoglu, K., Hassabis, D., Hadsell, R., and Kumarans, D. (2018). Vector-based navigation using grid-like representations in artificial agents. *Nature*, 557(7705):429–433.

Beer, R. D. (2000). Dynamical approaches to cognitive science. *Trends in cognitive sciences*, 4(3):91–99.

Brooks, R. A. (1991). Intelligence without representation. *Artificial intelligence*, 47(1-3):139–159.

Harnad, S. (1990). The symbol grounding problem. *Physica D: Nonlinear Phenomena*, 42(1-3):335–346.

Harvey, I., Husbands, P., Cliff, D., Thompson, A., and Jakobi, N. (1997). Evolutionary robotics: the sussex approach. *Robotics and autonomous systems*, 20(2-4):205–224.

Marocco, D., Cangelosi, A., and Nolfi, S. (2003). The emergence of communication in evolutionary robots. *Philosophical Transactions of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences*, 361(1811):2397–2421.

Nasr, K., Viswanathan, P., and Nieder, A. (2019). Number detectors spontaneously emerge in a deep neural network designed for visual object recognition. *Science advances*, 5(5):eaav7903.

Noguchi, W., Iizuka, H., and Yamamoto, M. (2017). Cognitive map self-organization from subjective visuomotor experiences in a hierarchical recurrent neural network. *Adaptive Behavior*, 25(3):129–146.

Noguchi, W., Iizuka, H., and Yamamoto, M. (2022a). Multimodal shared module that enables the bottom-up formation of map representation and top-down map reading. *Advanced Robotics*, 36(1-2):85–99.

Noguchi, W., Iizuka, H., Yamamoto, M., and Taguchi, S. (2022b). Superposition mechanism as a neural basis for understanding others. *Scientific Reports*, 12(1).

Nolfi, S. and Floreano, D. (2000). *Evolutionary robotics: The biology, intelligence, and technology of self-organizing machines*. MIT press.

Pfeifer, R. and Scheier, C. (2001). *Understanding intelligence*. MIT press.

Searle, J. R. (1980). Minds, brains, and programs. *Behavioral and Brain Sciences*, 3(3):417–424.

The Evolution of Genetic Robustness for Cellular Cooperation in Early Multicellular Organisms

Katherine G. Skocelas[1,2,3] Austin J. Ferguson[1,2,3] Clifford Bohm[3,4] Katherine Perry[1,3]
Rosemary Adaji[3,5] Charles Ofria[1,2,3]

[1]Department of Computer Science [2]Program in Ecology, Evolution, and Biology
[3]BEACON Center for the Study of Evolution in Action [4]Department of Integrative Biology [5]Department of Epidemiology
Michigan State University, East Lansing, MI 48824
kgs@msu.edu

Abstract

The major evolutionary transition to multicellularity shifted the unit of selection from individual cells to multicellular organisms. Constituent cells must regulate their growth and cooperate to benefit the whole organism, even when such behaviors would have been maladaptive were they free living. Mutations that disrupt cellular cooperation can lead to various ailments, including physical deformities and cancer. Organisms therefore employ mechanisms to enforce cooperation, such as error correction, policing, and genetic robustness.

We built a simulation to study this last mechanism under a range of evolutionary conditions. Specifically, we asked: How does genetic robustness against cellular cheating evolve in multicellular organisms? We focused on early multicellular organisms (with only one cell type) where cells must control their growth to avoid overwriting each other. In our model, unrestrained cells will outcompete restrained cells within an organism, but restrained cells alone will result in faster reproduction for the organism. Ultimately, we demonstrate a clear selective pressure for genetic robustness in multicellular organisms and show that this pressure increases with the total number of cells in the organism.

Introduction

Multicellular organisms have needed to coordinate cellular activity since the origin of multicellularity three and a half billion years ago (Callier, 2019). Within these organisms, cells must cooperate with copies of themselves for the higher-level organism to function and reproduce (Smith and Szathmary, 1997; Calcott and Sterelny, 2011; Queller, 2000). The clonal nature of cells in an organism ensures that kin selection aligns cellular and organismal goals, but mutants can arise that do not engage in the cooperative behaviors. If those mutants also replicate more rapidly, they can disrupt cooperation, reducing the fitness of the organism, possibly leading to cancer or death.

A variety of techniques are used by multicellular organisms to prevent defection from cooperation. These include *policing* (monitoring cellular behavior and punishing or killing cells that fail to cooperate properly), *apoptosis* (cellular suicide to avoid engaging in harmful behaviors), *error correction* (repairing mutations before they can cause harm), and genetic robustness (reducing the probability of harmful effects from mutations that do occur). Here, we focus on this final technique and examine the selective pressures by which multicellular organisms evolve robustness to mutational effects (by means of simple redundancy) in order to preserve cooperation.

Genetic robustness preserves phenotypic traits despite mutational disruption, typically via redundancy or compensatory processes (De Visser et al., 2003), and is prevalent throughout nature. Knockout experiments in yeast, for example, have demonstrated that up to half of all genes can be individually deactivated with minimal impact on fitness (Thatcher et al., 1998). Significant attention has therefore been given to the study of genetic robustness (Kitano, 2004; Lauring et al., 2013; Masel and Siegal, 2009; Lenski et al., 2006), including the role of redundancy (Gu et al., 2003; Láruson et al., 2020).

In designing our system, we choose to focus on one of the most simple forms that cellular cooperation takes: avoiding killing neighboring cells during proliferation. Indeed, inhibition of cellular replication is a hallmark of a major evolutionary transition, as a collection of cells begins to act as a higher-level individual by sacrificing their immediate replication potential for the good of the whole (Calcott and Sterelny, 2011).

Each cell division in a multicellular organism has a chance of incurring a mutation that reduces the daughter cell's ability to inhibit its own proliferation. As such, evolution selects for robustness in cooperative reproductive strategies, often with genetic redundancies to ensure that cells maintain their limits on cellular proliferation. All else equal, organisms that experience more cell divisions in their lifetimes (via having more cells or living longer) are more likely to accumulate mutations that cause cells to proliferate out of control. As such, we expect larger and longer-lived organisms to have greater selective pressures for redundancies in cellular controls.

Indeed, this type of loss of inhibited cellular proliferation is often studied in the context of cancer. As described by Nunney (1999), "Cancer occurs because the genetic control of cell growth is vulnerable to somatic mutations [...], particularly in large, continuously dividing tissues." Despite undergoing more cell divisions in their lifetimes, however, larger and longer-lived species do not typically exhibit proportionally increased rates of cancer, a phenomenon known as Peto's paradox (Peto, 1977). In fact, Nunney (1999) statistically demonstrated that the mechanisms to avoid mutations or cancer-causing mutational effects in small and large species are so different that the mechanisms to prevent cancer in one would not be evolutionarily stable in the other. Mechanisms used by smaller organisms would be ineffective in larger organisms, while mechanisms in larger organisms would be too expensive to sustain in smaller organisms.

Does this effect occur in more primitive circumstances? We simulated asexual multicellular organisms composed of cells as they would have existed shortly after the transition to multicellularity, before any developmental processes or division of labor evolved. Cells must inhibit their growth to avoid killing (overwriting) neighboring cells, which would reduce organism fitness. The only mechanism that evolution has to work with is genetic robustness, implemented as redundancy of the inhibitory behavior. We ask: Does the selective pressure for inhibited cellular proliferation during cellular reproduction increase with organism size? Specifically, do larger organisms evolve more redundancy for inhibited cellular proliferation on average?

Simulation

To test our hypothesis, we simulated evolving populations of multicellular organisms in a system that we named Primordium. Computer simulations of multicellular organisms have often been utilized to study the relationship between intra- and inter-cellular competition (Goldsby et al., 2014a,b; Pfeiffer and Bonhoeffer, 2003; Moreno and Ofria, 2022; Rose et al., 2020). We focused solely on organism growth (tissue accretion), avoiding complexities associated with more evolved multicellular life, such as specialized cell types and developmental patterns. Each organism begins as a single cell near the middle of a square grid; as soon as the grid is filled, the organism reproduces. As such, the size of the grid represents the organism's body size, and the speed of filling the grid is the organism's fitness. Within an organism, each cell attempts to replicate into a random neighboring grid position. An unrestrained cell will always place its offspring in the selected grid position, even if doing so replaces and kills an existing cell. A restrained cell will abort replication rather than replace a neighbor, since replacement would restart replication from that position. In practice, an organism consisting of restrained cells reproduces up to 20% faster than an organism with all unrestrained cells. Organisms that begin with a restrained cell but develop unrestrained cells during their lifetime (due to somatic mutations) end up with intermediate fitness values depending on how early unrestrained cells arose.

A cell's level of restraint is determined by its genome of zeros and ones. Organisms also have a sequestered germ cell whose genome is used to produce its initial somatic cell and is inherited by any offspring organisms. A cell is restrained if the number of ones in its genome is greater than a *restraint threshold* (60% of the genome in this work). The number of ones beyond the restraint threshold is termed the *restraint value* of that cell. A somatic mutation may induce a bit flip when a cell replicates, changing the daughter cell's restraint value. Likewise, germ-level mutations may occur when a whole organism reproduces. Since the restraint value of an organism's germ cell is inherited, it is under selection at the population level; we term this value the *restraint buffer* of the organism since it sets the initial cell's restraint value. A higher restraint buffer means that more somatic mutations can be sustained by cells (on average) before restraint is lost. (For full details on Primordium, see Methods below).

Results overview

In experiments with Primordium, we demonstrated that larger organisms have a stronger selective pressure for high restraint buffers, but many complicating factors exist. These include that mutation-selection balance can have a profound effect on the outcome of evolution, especially when restraint mechanisms require substantial genetic material. While larger organisms benefit more from a higher restraint buffer, the fitness benefit they gain declines with each additional step in restraint, making it more difficult for evolution to achieve. Thus, while we see that increasing organism size heightens selective pressure for large restraint buffers, we also find that it can easily be countered by mutational drift or the effects of a noisy fitness function. We conclude that selection technically favors higher levels of restraint in larger organisms, but other factors may prevent evolution from realizing those levels. Only under perfectly idealized conditions are we able to observe a positive relationship between organism size and restraint buffer value that continues into the largest sizes, indicating that additional factors may need to be present in nature.

Methods

Below, we detail the implementation of Primordium (Ferguson et al., 2022), the experimental methods used for data collection, and the statistical analyses performed on the results.

The Primordium evolution system

Primordium is a digital system that simulates an evolving, well-mixed population of multicellular organisms, useful for investigating how organism size influences the evolution of cellular restraint. Each organism maintains a sequestered germ cell that is used to initialize its body (with a single somatic cell, or 'soma') and for the genetic material passed to its offspring. At birth, the initial soma is placed near the center of a two-dimensional square grid (see Figure 1). Cells copy themselves into neighboring grid positions and are subject to somatic mutations with each replication. An organism reproduces when its entire grid is filled with cells. The population of organisms is kept at a constant size, with new offspring always replacing a random existing organism; this replacement is the only mechanism for organism death. As such, organisms that reproduce faster are more likely to produce offspring before being overwritten, creating a selective pressure for organisms that fill with cells as rapidly as possible.

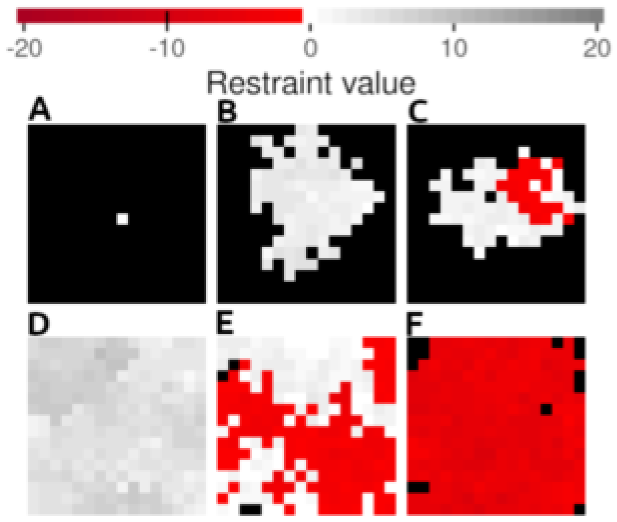


Figure 1: A visualization of six example 16x16 organisms. The color scale for cells is shown above, and empty cells are shown in black. Organism A is a brand new orgnaism. Organisms B and C have the same restraint buffer (3) and have both grown for 300 updates, but unrestrained cells have only appeared in organism C. The second row shows three organisms with restraint buffers of 5 (D), 2 (E), and -5 (F). All three organisms have received 2,732 updates, which is how long it took for organism D to fill and thus reproduce.

Inside an organism, each cell has a replication timer that starts when the cell is born. A cell's replication time is calculated as a base number of cycles (100) plus a noise factor (uniform between zero and 50 cycles) to stagger replication. When the timer elapses, the cell randomly chooses an offspring position from the eight options in its Moore neighborhood. If the selected position is empty, the cell replicates into it. If the position is occupied, however, the behavior depends on whether the parent cell's genome encodes for restraint. If the cell is *restrained*, the offspring cell is discarded, leaving the neighbor unaffected; if the cell is *unrestrained*, the neighboring cell is overwritten by the offspring, which then starts replicating itself from the beginning. In either case, the parent cell's timer is then reset to zero and a new replication time is calculated.

Each genome is a bitstring (length 100 by default) that determines if its cell is restrained. A *restraint threshold* indicates the minimum number of ones in the bitstring required for the cell to be restrained (here, 60% of genome length). While restrained behavior is determined merely by whether the number of ones in a genome is greater than the threshold, we also measure a cell's *restraint value* as the difference between these values. For restrained cells, this value indicates how many restraint-reducing mutations can be sustained before restraint is lost. For unrestrained cells, restraint value is negative, indicating how far away from restraint the cell is. Note that the data in a cell's genome is fully encapsulated in its restraint value, as order does not matter. Mutations do, however, function as they would in a bitstring; for example if a length 100 genome has 70 ones, that corresponds to a 30% chance of a mutation increasing the number of ones and a 70% chance of it decreasing.

Given that the default restraint threshold requires at least 60% ones in a genome, somatic mutations will reduce restraint on average; as such, mutation accumulation during soma growth moves toward unrestrained behavior. Since restrained cells never overwrite their neighbors, they are at a competitive disadvantage against any unrestrained cells that may appear.

Organism reproduction also has some probability of mutation when the parent organism's sequestered germ cell is passed to the child organism. This probability is parameterized as the germ mutation rate; we used, by default, a 2% chance per organism reproduction event. If a mutation occurs, a single bit in the germ cell's genome is flipped, increasing or decreasing the offspring's germ cell restraint value by one. To create a distinction between the cellular and organismal levels, we record the restraint value of the germ cell as the organism's *restraint buffer*. The restraint buffer provides an indication of how large the organism can grow before mutation accumulation starts producing unrestrained cells. Thus, different restraint buffers can result in different reproduction times for organisms and is the only factor subject to evolution in between-organism competition. Larger organisms undergo more cell replications and therefore require higher restraint buffers to avoid unrestrained cells arising within their lifetime.

Simulation controls for infinite population sizes and genome lengths

To exclude the possibility that our results were caused by insufficient population size or genome length, we performed controls to simulate infinite-size populations and infinite-length genomes.

In a finite genome, a bit flip from a zero to a one would decrease the number of zeros and thus decrease the likelihood that subsequent mutations would hit a zero. We simulated an infinite genome by removing this feedback on subsequent probabilities; specifically, we locked both germ and somatic mutations to a 60% probability of reducing restraint, which is the same probability as finite genomes at the restraint threshold. We also removed all limits on restraint values and restraint buffers.

To simulate an infinite population, we converted the principles of Primordium into a population genetics model. For each replicate, we first used Primordium to gather the distribution of organism fitness at each restraint buffer value. We then applied an iterative formula to determine the restraint buffer distribution after a given number of generations (based on models of asexual haploids in discrete, non-overlapping generations (Crow and Kimura, 1970)). Each generation was represented by the portion of the population with each restraint buffer. The weighted proportion of offspring was initially determined as the current proportion times its (empirically measured) expected fitness. After normalizing these values (such that all proportions again add up to one), we accounted for mutations, moving an appropriate portion of each group to a restraint buffer category of plus or minus one. Full details of the model can be found in Section 10 of the supplement (Ferguson et al., 2022).

Experiment design

In our baseline experiment for this work, we examined the effect of six organism sizes (16x16, 32x32, 64x64, 128x128, 256x256, and 512x512) on the evolution of restraint using all default parameters. We conducted more limited studies at organism size 1024x1024, but these experiments proved too slow to include for all results and are included in supplement Sections 2 and 9 (Ferguson et al., 2022). The initial experiment produced results partially opposing our hypothesis, so we then conducted additional experiments to tease apart the underlying dynamics. Specifically, we analyzed the importance of the germ mutation rate, somatic mutation rate, genome length, and population size.

We experimentally determined the default values for each parameter by conducting preliminary experiments where we swept each value (data available in supplement Sections 3-6 (Ferguson et al., 2022)). By default, all of our experiments consisted of a population of 200 multicellular organisms evolving for 10,000 generations with 100-bit genomes, and each treatment was replicated 100 times with different random number seeds. The restraint threshold was always set such that 60% of the genome must be ones in order to confer restraint. All organisms in the population begin with a restraint buffer of zero (*i.e.*, evolution always begins at the restraint threshold).

With Primordium's default parameters, there is a 50% chance that a single somatic mutation will occur when an individual cell replicates and a 2% chance of a single germ mutation when a whole organism reproduces. Both types of mutations toggle a single bit and will therefore either increase or decrease the restraint value of the genome by one.

To make the computational costs feasible, we pre-generated the replication time data for organisms before evolution. We simulated 100 organisms at each possible combination of restraint buffer value and organism size to produce a distribution of the time required for the organism to reproduce. During evolution, each time an organism is born, we pull from these distributions to determine when it will reproduce instead of simulating each individual cell. Comparison experiments demonstrated that this limited number of fitness samples has no qualitative effects on the overall evolution of restraint (see supplement Section 5 (Ferguson et al., 2022)).

At the beginning of the simulation, all organisms are given a generation value of zero. When an organism reproduces, the offspring's generation is set to its parent's generation plus one. Every time the average generation value of organisms in the population surpasses a whole number, we collect the average restraint buffer of all organisms in the population. Examining these values from preliminary data, we determined that populations had stabilized by the time the average generation crossed 10,000 (*i.e.*, running the simulation longer produced no additional changes in the evolved restraint buffer values). Most analyses focused on the average restraint buffers at the end of 10,000 generations of evolution. Additionally, we analyzed the pre-generated replication times for organisms under various configurations to determine how different parameters affected fitness.

Statistics and data availability

All statistics were calculated by first performing a Kruskal-Wallis test to determine if significant variation existed across treatments. When significance was indicated, we determined which treatments were significantly different with a pairwise Wilcoxon Rank-Sum test and Bonferroni-Holm corrections for multiple comparisons. Statistics have been included in the figures where appropriate, and all statistics are available in the supplement (Ferguson et al., 2022). Primordium was developed with the Empirical C++ library for scientific software (Ofria et al., 2020). Analyses and visualizations were conducted using R version 3.6.3 (R Core Team, 2020) and the ggplot2 library (Wickham, 2016). All source code, analyses, and other supplemental material can be found on GitHub (Ferguson et al., 2022).

Results & Discussion

For our baseline analysis, we examined the evolved restraint buffer for a range of organism sizes. Larger multicellular organisms, by definition, undergo more cellular replication on average. As such, a higher restraint buffer is required to avoid unrestrained cells appearing during organism growth and slowing reproduction. Indeed, we found that (on average) the evolved restraint buffer initially increases with organism size. A turning point emerges at size 128x128, however, beyond which the evolved restraint buffer decreases, in opposition to our expectation (see Figure 2).

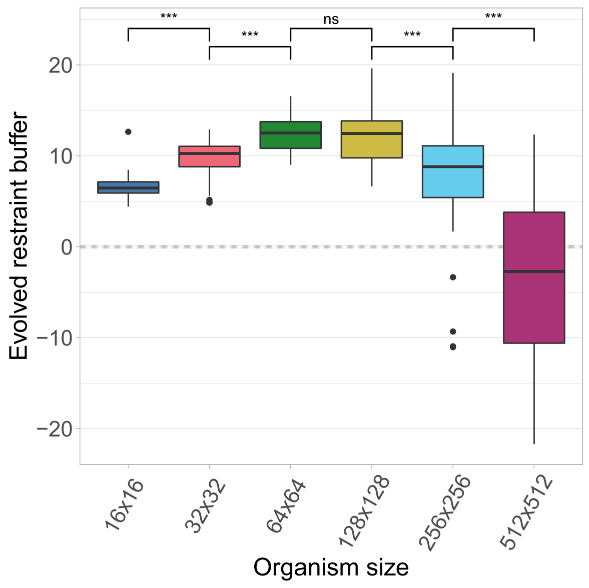


Figure 2: Boxplots show the evolved restraint buffers of populations with varying organism sizes and 100-bit genomes. Each boxplot represents 100 independent replicates. Each replicate is summarized as the average restraint buffer of all organisms at the end of 10,000 generations. Colors are unique for each organism size and consistent across all figures. Statistics between adjacent organism sizes are shown (for all figures: 'ns' for $p > 0.05$, '*' for $p \leq 0.05$, '**' for $p \leq 0.01$, '***' for $p \leq 0.001$)

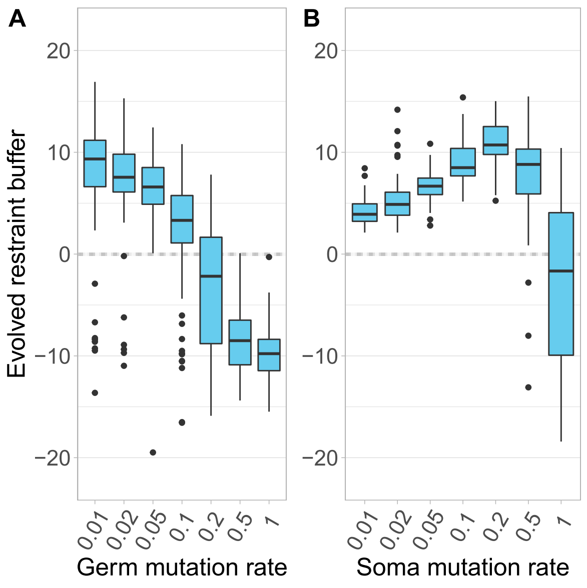


Figure 3: Boxplots show the evolved restraint buffers of populations of 256x256 organisms evolved under varying A) germ and B) somatic mutation rates. Both mutation rates are quantified per-genome. Each boxplot represents 100 independent replicates. Each replicate is summarized as the average restraint buffer of all organisms at the end of 10,000 generations.

Due to the simplicity of this system, there are only two key pressures acting on the population: selection and mutational drift. Selection acts on both the population level (improved organism restraint can speed up reproduction) and the cellular level (unrestrained cells spread faster). Mutational pressures are always biased toward an equal balance of zeros and ones, which is below the 60% ones needed for restraint. At the population level, this biased mutational pressure acts counter to selection, pulling down germ restraint buffers. Mutational pressure from somatic mutations provides a selective advantage for organisms with a higher restraint buffer, as their cells remain restrained for longer. Where mutation-selection balance leaves a genome at the end of evolution depends on the strength of each pressure and the nuances of their interactions. Below, we disentangle how these pressures interact across levels to create the observed peak and subsequent decline in evolved restraint values.

Mutation rates for germ and somatic cells push evolution in opposing directions

Mutations to organism germ cells provide variation for evolution to act upon. If the mutation rates for these cells are too high, however, they create a strong mutational pressure for less restrained organisms, overwhelming selection. Figure 3A shows how evolution is less capable of producing high restraint buffers as germ mutation rates increase; 256x256 organisms are used for illustrative purposes, but a qualitatively similar effect can be seen at all organism sizes (see Section 4 of the supplement (Ferguson et al., 2022)).

Conversely, resistance to somatic mutations is why higher levels of restraint are valuable in the first place. If the somatic mutation rate drops too low, then a high restraint buffer is no longer valuable and is not selected by evolution. Figure 3B demonstrates this effect by measuring evolved restraint buffers in tests of 256x256 organisms at various somatic mutation rates. If the somatic mutation rate is too high (above 0.2), however, we see that selection is no longer able to counter mutational decay and the organisms evolve smaller restraint buffers than they did at lower somatic mutation rates. Data for all organism sizes are available in Section 3 of the supplement (Ferguson et al., 2022))

In considering how these effects play out in nature, germ cells are sequestered in most organisms in order to keep their mutation rates low. Somatic cells, on the other hand, do all of the dirty work for the body and thus tend to be subjected to mutation rates two orders of magnitude higher (Milholland et al., 2017). As such, the combination of mutation rates that we use as our default parameters appear to be realistic for natural evolution.

Longer genomes reduce mutational pressure

In our baseline experiment, each genome consists of 100 bits, with 60 bits (60%) needing to be ones for a cell to be restrained. As such, a cell with a restraint value of zero (exactly 60 ones) would have a 60% chance of a mutation eliminating its restraint and a 40% chance of increasing it. A restraint value of 10 would shift these values to a 70% chance of reduction and only a 30% chance of increase.

We examined the comparable situation in genomes of other lengths, from 25 bits to 400 bits in length, each with a 60% threshold for restraint. A cell with a length 400 genome and a restraint value of 10 has 250 ones in it, and thus only a 62.5% chance of a restraint-reducing mutation and a 37.5% chance of a mutation increasing restraint. At the other extreme, a cell with a length 25 genome and a restraint buffer of 10 consists of all ones, and so it has a 100% chance of a mutation reducing its restraint.

What effect should a longer genome have on evolution? At the population level, it should reduce the incremental mutational pressure against restraint, and thus allow higher restraint buffers to evolve in organisms. Within cells, however, a longer genome also decreases the probability of restraint-reducing somatic mutations, reducing the rate at which restraint decays, and thus weakening selection for high restraint buffers. We created an additional control that isolated the change in mutational pressure, but the results were qualitatively identical to those described below (see Section 8 of supplement for details (Ferguson et al., 2022)).

Figure 4A shows that, when we evolve 256x256 organisms with differing genome lengths, longer genomes result in the evolution of higher restraint buffers. The range of evolved restraint buffers also increases with genome length, because all genome lengths have results that extend to the median of the genome, which is at a lower restraint buffer in longer genomes. Indeed, examining length-400 genomes at all organism sizes, we see in Figure 4B that restraint buffers now peak at organisms of size 256x256, but drop again for the largest organisms.

To ensure these trends were not the result of insufficient genome length, we repeated the experiment with infinite genomes (*i.e.*, all mutations have a 60% chance of decreasing restraint and restraint buffers are not capped). These results were qualitatively the same. While the "peak" organism size increased, the constant mutational pressure was still strong enough to cause a downturn in evolved restraint at the largest organism sizes (see Section 9 of supplement (Ferguson et al., 2022)).

Natural genomes are, of course, not infinite, yet they are far longer than the finite genomes used in this study, though only a small portion of a natural genome is related to cellular restraint. The system we use is also far simpler than the complex regulatory networks involved in real-world organisms, and in future work it will be important to examine how those complications play out.

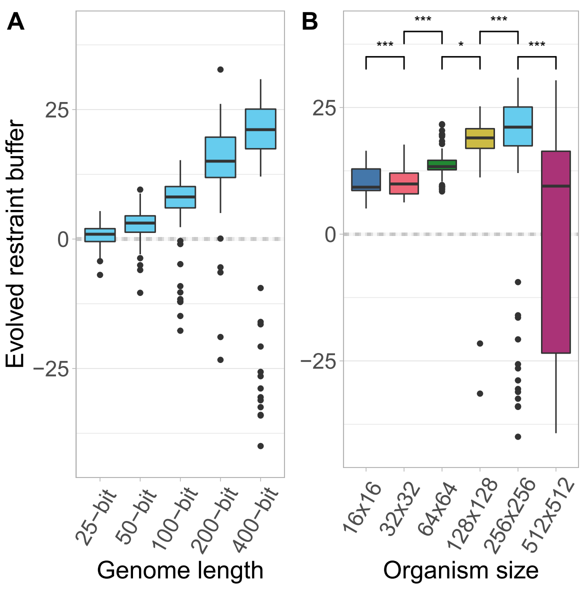


Figure 4: Subplot A shows the evolved restraint buffer for each genome length with a 256x256 organism, while subplot B shows the evolved restraint buffer at each organism size for a 400-bit genome. Each boxplot represents 100 independent replicates. Each replicate is summarized as the average restraint buffer of all organisms at the end of 10,000 generations.

The selective pressure for restraint has diminishing returns

Next, we analyzed how the selective pressure at different restraint values changes with organism size. While each successive increment to the restraint buffer proved to be beneficial, we observed a diminishing return in that benefit in practice (see Figure 5). Small organisms see the greatest fitness boost just above the restraint threshold, after which the fitness advantage for additional restraint quickly diminishes once an organism's restraint buffer is high enough that unrestrained cells never appear during its lifetime. Larger organisms have a smaller initial spike, but given that they experience more cell divisions, the saturation point occurs at larger restraint buffer values. Thus, selective pressure persists longer in larger organisms. In fact, in the largest organisms (256x256 and 512x512), this saturation point is unreachable with the 100-bit genome.

While the fitness data shows that additional bits of restraint still provide a small selective advantage to larger organisms (Figure 5), the evolved restraint buffers decline for organisms larger than 128x128 (Figure 2). We know that

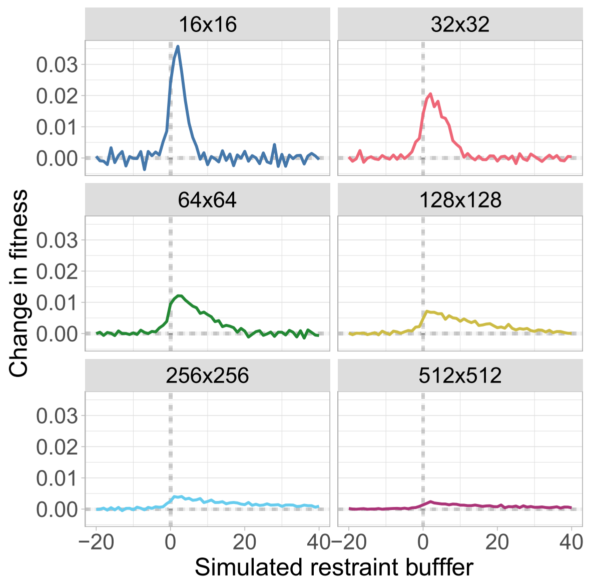


Figure 5: Lines show the average benefit of each additional bit of restraint buffer for various organism sizes. Values are calculated as the measured difference in fitness between n bits of restraint and $n - 1$ bits of restraint, and each restraint buffer value was averaged over 10,000 samples. Larger values on the y-axis indicate a greater increase in benefit. Data were calculated to a restraint value of -60, but all values below -20 fluctuate around zero across all organism sizes.

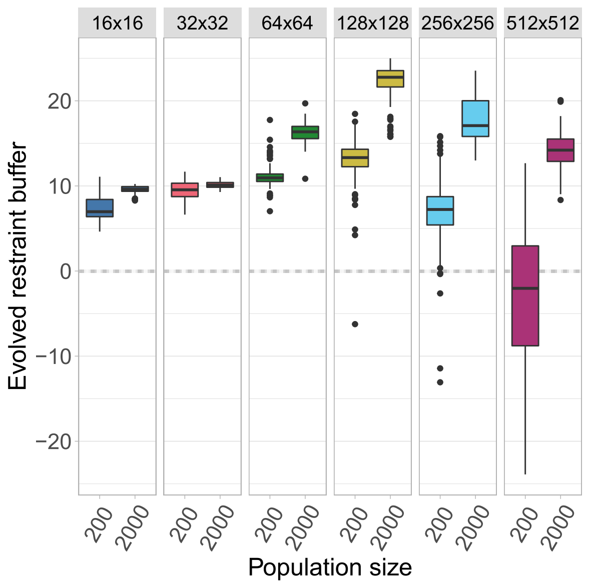


Figure 6: Boxplots show the evolved restraint buffers of populations with 100-bit genomes. Each subplot shows a particular organism size evolved with two population sizes: 200 organisms (left boxplot, default) and 2,000 organisms (right boxplot). Each boxplot represents 100 independent replicates, each summarized as the average restraint buffer of all organisms at the end of 10,000 generations. All comparisons between population sizes of 200 and 2,000 organisms are highly significant ($p \leq 0.001$).

populations are expected to evolve to the restraint value where mutational and selection pressures are in equilibrium. Thus, the selection pressure for additional bits of restraint must have become too weak to oppose the mutational pressure against additional bits. From evolutionary theory, we know that population size greatly affects selective pressure, with larger populations experiencing increased selective pressure (Gossmann et al., 2012). Thus, we replicated the original experiment with larger populations to observe the results of increased selective pressure. Indeed, Figure 6 shows that increasing the population size from 200 to 2,000 increased the evolved restraint buffer at every organism size. Even with this larger population, however, the evolved restraint buffers still decrease at sizes greater than 128x128.

Natural populations are often huge, so we extended this experiment with a population genetics model to simulate the selection pressure in an infinite population. In Figure 7A, we see that populations of size 256x256 organisms evolve more restraint than the 128x128 populations, but the 512x512 populations still evolve less restraint. Therefore we conclude that the mutational pressure is a limiting factor in the evolution of restraint in our system.

When all other factors are controlled for, larger organisms evolve greater restraint

Even with the increased selective pressure of an infinite population size, the largest organisms are unable to overcome the mutational pressure of the finite genome to evolve more restraint than the next largest organism size. Thus, we asked if combining the infinite population with an infinite genome to neutralize mutational pressure would be sufficient to continue the expected trend. Figure 7B shows that, indeed, when both the infinite population and infinite genome controls are in place, the evolved restraint buffer continues to increase with organism size. Since this combination of controls changes the trend, we conclude that the downturn pattern must be a combination of increasing mutational pressure and decreasing selective pressure as restraint buffers increase.

In nature, populations are not unlimited, but are often much larger than the size-2,000 populations that we tested. Similarly, while biological genomes are not unlimited, they are typically significantly larger than modeled here. Thus, our system's infinite population and genome model highlights factors that are clearly important for these dynamics, but are reasonable to assume that they would exist in nature.

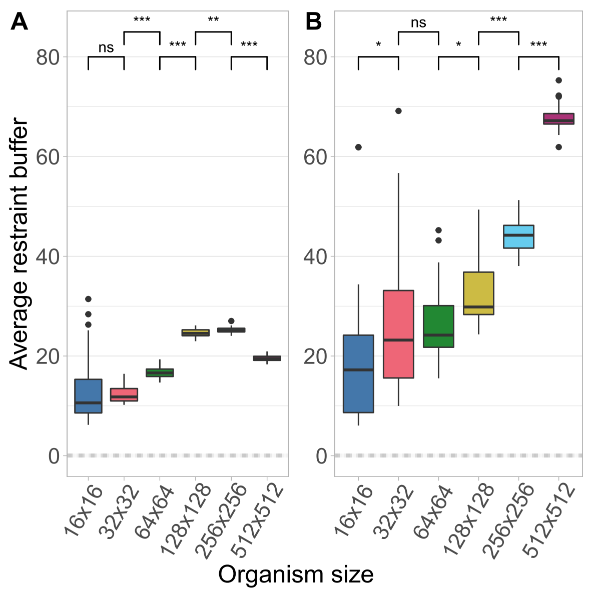


Figure 7: Boxplots show the average restraint buffer value after 10,000 updates in an infinite population model. Each boxplot represents 100 independent replicates. Subplot A shows the results for the default 100-bit genome, while subplot B shows the results for the infinite genome.

Conclusions

We have shown evidence that the pressure imbued by simple space management can select for genetic robustness to improve cooperation. Larger organisms optimize their fitness advantage if their genomes are more mutations away from an unrestrained state. After an initial transition to multicellularity, these dynamics appear sufficient to create a selective pressure for increased restraint on cellular replication without the need for developmental patterning.

While larger organism sizes do have increased selective pressure for larger restraint buffers, we have also found that this is not a strong effect. As the restraint buffer increases, so too does mutational pressure. Larger organisms can have mutational pressure so strong (and selective pressure so weak) that the evolved restraint buffer actually decreases.

In Primordium, being unrestrained results in at most a 20% fitness loss for an organism, making it possible to overcome with other pressures. More complex aspects of multicellular organisms (such as developmental patterns) would make unrestrained cellular behaviors more harmful or even fatal. As a result, “the cells of multicellular organisms, even those with body plans as simple as sponges, have evolved mechanisms to maintain appropriate numbers of cells within tissues” (DeGregori, 2011). Avoiding the turning point that we saw in Primordium required unrealistic controls, indicating that there must, indeed, be other factors in natural systems increasing the selective pressure for restraint. The dynamics we observed, however, may help explain the initial bootstrap as multicellular organisms first evolved to be large enough for developmental processes to become beneficial. The benefit to biological organisms of being larger also applies a strong indirect selective pressure for restraint. Dedicating more energy and more of their genome to restraint mechanisms imposes a cost on organisms, but helps not only in avoiding diseases such as cancer, but also facilitates achieving larger body masses. A larger body size can improve predation success, defense against predation, range of food sources, mating success, longevity, and intelligence (with increased brain size) (Hone and Benton, 2005).

From the perspective of cancer research, it is clear that Peto’s paradox (*i.e.*, larger/longer-lived species are expected to exhibit proportionally higher rates of cancer, but do not in practice) is the result of many evolutionary forces and dynamics (Peto, 1977). Here, we focused only on the pressure imbued by the management of space as a resource, disallowing the evolution of most parameters (organism size, genome length, cell replication strategies, *etc.*).

Furthermore, Primordium uses binary genomes and a simple restraint mechanism. In a real-world biological system, restraint would be harder to build than destroy—worsening mutational pressures—but a smaller region of the genome would encode for it, making it a smaller mutational target.

Many of these complications would be exciting to study to identify their effects on genetic robustness in multicellular organisms. For example, we should examine the effect of a more complex genome alphabet and genes that code for functional fitness. Primordium would also be ideal to study the evolution of genome length, with large genomes allowing for increased cellular robustness, trading off against a higher mutational load.

In nature, multicellular organisms can be huge, despite undergoing vastly more cell divisions. Indeed, the largest (blue whales) coordinate quadrillions of cells. Deciphering how this is possible will allow us to not only better understand our natural world, but also give us insights on how to evolve larger and more complex artificial organisms.

Acknowledgements

We thank Heather Goldsby, Acacia Ackles, Ben Kerr, the members of the MSU Digital Evolution Lab, and prior anonymous reviewers for their contributions. This work was supported by U.S. National Science Foundation grant DEB-1655715, the BEACON Center for the Study of Evolution in Action, and the Department of Defense (DoD) through the National Defense Science & Engineering Graduate (NDSEG) Fellowship Program. Computational resources were provided by the MSU Institute for Cyber-Enabled Research.

References

Calcott, B. and Sterelny, K., editors (2011). *The Major Transitions in Evolution Revisited*. Vienna Series in Theoretical Biology. MIT Press, Cambridge, MA.

Callier, V. (2019). Core Concept: Solving Peto's Paradox to better understand cancer. *Proceedings of the National Academy of Sciences*, 116(6):1825–1828.

Crow, J. and Kimura, M. (1970). An introduction to population genetics theory. Harper and Row, Publishers.

De Visser, J. A. G. M., Hermisson, J., Wagner, G. P., Meyers, L. A., Bagheri-Chaichian, H., Blanchard, J. L., Chao, L., Cheverud, J. M., Elena, S. F., Fontana, W., Gibson, G., Hansen, T. F., Krakauer, D., Lewontin, R. C., Ofria, C., Rice, S. H., Dassow, G. v., Wagner, A., and Whitlock, M. C. (2003). Perspective: Evolution and Detection of Genetic Robustness. *Evolution*, 57(9):1959–1972.

DeGregori, J. (2011). Evolved Tumor Suppression: Why Are We So Good at Not Getting Cancer? *Cancer Research*, 71(11):3739–3744.

Ferguson, A. J., Ofria, C., and Skocelas, K. (2022). Primordium Supplemental Material. GitHub. https://doi.org/10.5281/zenodo.6564985.

Goldsby, H. J., Knoester, D. B., Kerr, B., and Ofria, C. (2014a). The effect of conflicting pressures on the evolution of division of labor. *PLoS ONE*, 9(8):e102713.

Goldsby, H. J., Knoester, D. B., Ofria, C., and Kerr, B. (2014b). The evolutionary origin of somatic cells under the dirty work hypothesis. *PLoS Biology*.

Gossmann, T. I., Keightley, P. D., and Eyre-Walker, A. (2012). The effect of variation in the effective population size on the rate of adaptive molecular evolution in eukaryotes. *Genome Biology and Evolution*, 4(5):658–667.

Gu, Z., Steinmetz, L. M., Gu, X., Scharfe, C., Davis, R. W., and Li, W.-H. (2003). Role of duplicate genes in genetic robustness against null mutations. *Nature*, 421(6918):63–66.

Hone, D. W. and Benton, M. J. (2005). The evolution of large size: How does Cope's Rule work? *Trends in Ecology and Evolution*, 20(1):4–6.

Kitano, H. (2004). Biological robustness. *Nature Reviews Genetics*, 5(11):826–837.

Lauring, A. S., Frydman, J., and Andino, R. (2013). The role of mutational robustness in RNA virus evolution. *Nature Reviews Microbiology*, 11(5):327–336.

Lenski, R. E., Barrick, J. E., and Ofria, C. (2006). Balancing Robustness and Evolvability. *PLoS Biology*, 4(12):e428.

Láruson, A. J., Yeaman, S., and Lotterhos, K. E. (2020). The Importance of Genetic Redundancy in Evolution. *Trends in Ecology & Evolution*, 35(9):809–822.

Masel, J. and Siegal, M. L. (2009). Robustness: mechanisms and consequences. *Trends in Genetics*, 25(9):395–403.

Milholland, B., Dong, X., Zhang, L., Hao, X., Suh, Y., and Vijg, J. (2017). Differences between germline and somatic mutation rates in humans and mice. *Nature Communications*, 8(1):1–8.

Moreno, M. A. and Ofria, C. (2022). Exploring Evolved Multicellular Life Histories in a Open-Ended Digital Evolution System. *Frontiers in Ecology and Evolution*, 10:750837.

Nunney, L. (1999). Lineage selection and the evolution of multistage carcinogenesis. *Proceedings of the Royal Society of London. Series B: Biological Sciences*, 266(1418):493–498.

Ofria, C., Moreno, M. A., Dolson, E., Lalejini, A., Papa, S. R., Perry, K., Boyd, S., Fenton, J., Jorgensen, S., Hoffman, R., Miller, R., Edwards, O. B., Stredwick, J., Clemons, R., Vostinar, A., Moreno, R., Nitash C G, Zaman, L., Schossau, J., and Rainbow, D. (2020). Empirical. https://doi.org/10.5281/zenodo.2575606.

Peto, R. (1977). Epidemiology, multistage models, and short-term mutagenicity tests. In Hiatt HH, Watson JD, W. J., editor, *The origins of human cancer, Cold Spring Harbor conf. on cell proliferation.*, pages 1403–1428. Cold Spring Harbor Laboratory Press, Cold Spring Harbor, NY.

Pfeiffer, T. and Bonhoeffer, S. (2003). An evolutionary scenario for the transition to undifferentiated multicellularity. *Proceedings of the National Academy of Sciences*, 100(3):1095–1098.

Queller, D. C. (2000). Relatedness and the fraternal major transitions. *Philosophical Transactions of the Royal Society of London. Series B: Biological Sciences*, 355(1403):1647–1655.

R Core Team (2020). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria.

Rose, C. J., Hammerschmidt, K., Pichugin, Y., and Rainey, P. B. (2020). Meta-population structure and the evolutionary transition to multicellularity. *Ecology Letters*, 23(9):1380–1390.

Smith, J. M. and Szathmary, E. (1997). *The Major Transitions in Evolution*. Oxford University Press, New York, NY.

Thatcher, J. W., Shaw, J. M., and Dickinson, W. J. (1998). Marginal fitness contributions of nonessential genes in yeast. *Proceedings of the National Academy of Sciences*, 95(1):253–257.

Wickham, H. (2016). *ggplot2: Elegant Graphics for Data Analysis*. Springer-Verlag New York.

The Information Complexity of Navigating with Momentum

Bente Riegler, Daniel Polani and Volker Steuber

School of Physics, Engineering and Computer Science, University of Hertfordshire
b.riegler@herts.ac.uk

Abstract

Many models of organism navigation concern themselves in essence just with the sequence of locations visited and how to manage it. However, larger and bulkier organisms have also to deal with managing momentum. We expect that this affects the cognitive management of movement. Here we propose a simple model for the information processing complexity of navigation when velocity and acceleration are considered, moving away from a kinematic perspective to a partially dynamic model, to separate the effects of location and momentum management.

The work is discussed in the context of recent neurobiological research suggesting that biological agents plan around acceleration and deceleration phases, showing high neural activity during their body's velocity changes.

Introduction

Commonly, navigation and movement tasks are modeled by defining movements through a sequence of positions, and eventually to a final position, be it via key poses or through forward and inverse kinematics. In more physically involved scenarios other approaches had great successes modelling the specific dynamics of the agent and their domain. Such models include the presence of momentum and inertia and the use of force to effect changes. The inverse pendulum balancing task is a classical example and benchmark in nonlinear control theory (Boubaker, 2013). It has been solved with multiple algorithms taking the position and angular velocity as well as the mass and force into account (Boubaker, 2013; Furuta et al., 1992) to highlight few. In autonomous vehicle control, speed, angular velocity and vehicle mass constitutes the dynamic of the system. Reinforcement learning with hierarchical temporal abstraction has achieved safe control in merging traffic lanes (Shalev-Shwartz et al., 2016). In helicopter control, differential dynamic programming (Abbeel et al., 2006) allows for difficult maneuvers such as tail-in funnel or flip. Similarly, in legged robots approaches such as Zero-Moment-Point walking (Kajita et al., 2003; Mitobe et al., 2000) that generate walking motion while balancing the dynamical center of the particular agent. In all these approaches, the use and limits of force are largely given by real world physical models and the dynamics are centered around the specific use case.

Evidence from neurobiological research suggests that organisms whose brains include a cerebellum do in fact model and simulate their world in a dynamic instead of a kinematic way when controlling movement. Recent research by Becker and Person focusing on the cerebellar control activity of a mouse in regards to precisely reaching a goal position showed the importance of controlling velocities (Becker and Person, 2019). While the task is still defined by kinematics, the control very much requires velocity management. In their experiment, they measured both the velocity and the neural activity in the cerebellum of the mouse. While reaching for the goal, the mouse shows moderately high neural activity at the start of the reaching motion, during acceleration, and high activity during the deceleration phase, towards the end of the movement.

Having to consider velocity creates a timing component and thus requires some temporal planning. When operating with velocities, one has to manage momentum which requires the ability to integrate the application of forces. In fact, specific neural circuits in the cerebellum have the ability to integrate (Maex and Steuber, 2013). This suggests that an agent with a cerebellum is endowed with the capacity to predict or simulate possible future positions through forward integration which permits it to plan when the agent needs to manage velocity.

While the various scenarios of momentum/velocity management are specific to particular agents or organisms, we wish to extract several general insights which emerge from the information processing cost that momentum/velocity management requires as compared to purely kinematic/location-based navigation models. Current models that allow for velocity management be it in autonomous vehicle control, legged robot control or the inverse pendulum — though achieving balanced and precise movements within their problem domain — are too concerned with the specific agent, its physics and the task at hand to allow general insight. These models, furthermore, are not concerned with the pressure towards parsimonious information processing which would not be necessary for a merely optimally performing solution in the given problem domain, but becomes highly relevant once it comes to biological agents (Polani, 2009). Therefore, we will introduce a minimalistic model to study aforementioned phenomena.

The paper is organized as follows: in section 2 we will define our models and in section 3 we will define how we measure cognitive cost. In section 4 we will present our experiments and results which we will discuss in section 5.

Perception-Action Models

The perception-action loop setup throughout this paper is in line with the general Reinforcement Learning framework modeled as a Markov Decision Process (MDP) (Sutton and Barto, 2018). States are given by $s \in S$ and actions are given by $a \in A$. We assume the individual transitions, given by $p(s_{t_2}|s_{t_1}, a)$, with s_{t_1} being the state at time t_1, a the action taken in s_{t_1} and s_{t_2} being the resulting state after the action is performed at t_2, to be deterministic. A policy is denoted as $\pi(a|s)$, which denotes the probability that action a is selected in state s; such transitions incur rewards r that the agent aims to maximize over the run. The achieved rewards are summarized by the Q-function $Q^{\pi}(s, a)$ which expresses the reward that is accrued when, starting in state s and selecting first action a, the agent proceeds to follow policy π. In traditional MDPs, one seeks to find $Q^*(s, a)$ which maximises this value over all possible policies. Given a state s, an optimal action in the state can be directly read off this quantity, by selecting the action a that achieves the highest $Q^*(s, a)$ for the state s. This forms the basis of the reward structures we consider in the following.

In the following we will introduce two models: The standard kinematic/location-based (K/L) model which we will use as a baseline and our new proposed acceleration/velocity (A/V) model which includes velocity and acceleration. This allows us to model and handle inertia of the agent though we will not be specifically modeling mass or other physical implications. We will limit ourselves to look at abstract simple one-dimensional grid-like models to make the salient differences between the two setups as apparent as possible.

Kinematic/Location-based Model without Velocity

A typical way to represent navigation or movement tasks in reinforcement learning models uses actions that comprise the agent taking a single step from one location to a neighbouring one. Technically, this means accelerating the agent, moving it one step and immediately stopping it. Interpreted physically, this can be considered a model for high-friction where a movement stops immediately when the applied action a ceases.

As defined by the MDP framework there exists a state space (S) and an action set (A). The state space (S) consists of a set of discrete positions (P) aligned in one dimension. The action set (A) consists of actions that move the agent to an adjacent state, here *move left* $(m = -1)$ and *move right* $(m = +1)$. Every action incurs a cost of 1. We apply no discount over time, but consider episodic tasks only. Specifically, we assume there exists a single goal state which can be any state of S or a set of such states. Any goal state is modeled as a trapping state, i.e. once reached, it does not allow further action and does not incur further costs. In RL terminology, an episode ends once the agent reaches a goal state. Note that the trapping property is important for an appropriate calculation of the informational costs. The grid world is finite and limited by walls. An action that pushes the agent into a wall leaves the agent in the same state but still incurs the usual cost of 1. Since in the present paper we only consider optimal policies, no agent will walk into walls.

Acceleration/Velocity Model

In the following we describe the extended model. In this model, the states are not only defined by their position on the grid world but also the agent's velocity. Thus, each state of the MDP is now a tuple of position and velocity. The states are now tuples of positions (P) and velocities (V). For simplicity, we only consider integer velocities[1]. The state space is now $S = P \times V$. In our model, during each time step, the dynamics of the world moves the agent to a new position based on its current position and velocity:

$$p_{t+1} = p_t + v_t \tag{1}$$

In our simplistic one-dimensional model, the agent can chose between three actions: *positive acceleration* to the right $(a = +1)$ and *negative acceleration* to the left $(a = -1)$, and *no change* $(a = 0)$ which are added to the velocity:

$$v_{t+1} = v_t + a_t \tag{2}$$

Note the agent can not directly affect its position. It can only influence its velocity which then affects the position change mediated through the velocity. The agent's change of velocity will happen at the same time as the change in position which means any position change due to the choice of an action will only be observable at the subsequent time step.

The cost function incurs a cost of 1 for each time step outside of the goal regardless of the action taken. This grid world is limited by walls keeping the agent inside the world. Just like in the K/L model the agent will on its own try to avoid hitting the wall when it would not be optimal otherwise (note, though, that there are starting position-velocity pairs in which the agent cannot avoid hitting a wall).

Reaching the goal We will use two types of goal sets. Both are specified by a goal position.

[1]A velocity specification includes directionality

The first type allows for any velocity: $\{p_g\} \times V$ (i.e. a single given position, but with arbitrary velocity). When reaching/passing p_g, the agent will automatically be stopped, effectively reducing its velocity to 0 instantaneously, even if the agent would overshoot the goal otherwise. In this scenario the responsibility to decelerate the agent is placed on the environment (we interpret this as offloading the informational cost of an instantaneous stop to the embodiment of the agent). As a real-life analogue, one can compare this to the arresting gear for airplanes landing on an aircraft carrier.

The second type of goal set, on the other hand, requires the velocity to reach precisely zero for the goal to be considered satisfactorily achieved: $(p_g, 0)$. This goal type requires the agent to explicitly decelerate/break before reaching the goal position. In particular, overshooting will be considered a miss.

Cognitive Cost

Cognitive processing in natural agents requires neural activity which is energetically expensive yet crucial for the survival of the agent in question (Laughlin, 2001; Polani, 2009). As such, keeping the processing cost as minimal as possible without losing optimality with regards to some value function becomes an important secondary objective. In vivo, measurements of the cognitive load or neural activity of a living being can be measured using EEG (Niedermeyer and da Silva, 2005), fMRi (Huettel et al., 2004) or intrusive methods like implanted optical fibres (Becker and Person, 2019).

However, our minimal and theoretical model employs a different method of determining the cognitive cost. Neural computation which processes sensory inputs to make a decision which then in turn is communicated to the actuators of the agent can be directly translated to a message sent from the sensors to the actuators (Tanaka and Sandberg, 2015). This opens the way to use information theory as the basis to measure the information flow through the agent's perception-action loop which can be interpreted as the cognitive cost of any agent — theoretical or not (Polani, 2009). The main objective in our work is the consideration of utility-optimizing behaviours in the MDP while respecting the secondary objective of minimizing this cognitive cost. In general, one can further reduce cognitive cost by trading in some utility (Polani et al., 2006). However, for simplicity, we focus entirely on optimal policies.

Specifically, we will use the formalism of *relevant information* to measure the cognitive cost of the agent to control its movement (Polani et al., 2006). Relevant information for an MDP is defined as the minimal information required about the current state to select an action to achieve a given utility. It represents a lower bound of how much cognitive cost per decision is required to achieve a given utility. Formally, relevant information is defined as

$$\min_{\pi(A|S) s.t. E^{\pi}[Q(s,a)] \overset{!}{=} \bar{Q}} I(S;A) \tag{3}$$

i.e. as the minimum amount of information the actuators A use about the state S as to achieve a given utility, with $\bar{Q}$ being the desired utility of the MDP. We will typically choose $\bar{Q}$ as Q^*, the optimal utility. By introducing a Lagrangian factor β, this constrained minimization can be converted into the unconstrained minimization:

$$\min_{\pi}\big(I(S;A) - \beta E[Q(S,A)]\big) \tag{4}$$

In this paper we will focus on achieving the optimal values only, e.g. the Lagrangian 4 will be considered for β tending towards the infinite limit. We further exclude the goal states from the calculation of the mutual information $I(S;A)$, as these are at the end of an episode and contribute no relevant decision in the policy. Since, $I(S;A)$ is a concave function of $p(s)$ for a fixed $p(a|s)$ and a convex function of $p(a|s)$ for a fixed $p(s)$, the relevant information minimization is formally equivalent to the standard rate-distortion problem known from information theory (Cover and Thomas, 1991) with a different fixed point. The rate-distortion problem is solved with the Blahut-Arimoto fixed-point iteration algorithm (which implements in essence a sequence of mutual projections between two convex sets, see (Cover and Thomas, 1991). We use practically the same algorithm here, replacing the information-theoretical distortion of a signal by the optimal utility Q^* (Polani et al., 2006). [2]

Note that the naive use of Blahut-Arimoto in the present context is only possible since Q^* does not depend on the policy but only on the MDP. When one considers the general case of suboptimal policies, such a simplification is no longer possible (see Polani et al., 2006; Polani, 2009) for details).

Crucially, this optimization gives us two results: The minimal information cost the agent has to pay per step to ensure it reaches the goal with perfect cost expenditure and the policy with which to achieve this. Thus the informational cost of a particular setup which in turn is used to compare the overall cognitive cost of kinematic/location-based agent movement and the acceleration/velocity-based agent.

Model Comparison

We now compare the two presented models (K/L and V/A) directly and use the two goal types of the velocity/acceleration model(V/A model) using two different goal sets $\{p_g\} \times V)$ and $(p_g, 0)$. s We will look at a "border goal"

[2]to see the equivalence with the rate-distortion problem, note that our regret $Q^*(s, a^*) - Q^*(s, a)$ here effectively acts as a distortion, where s is the sent symbol, a^* is the desired transmitted symbol — the correct action — and a the actually received symbol — the actually chosen action.

as one extreme and the goal in the middle as the most general non-border position on the other end of the spectrum. Throughout these experiments we will limit the maximum velocity to one. Further acceleration is possible but does not affect the velocity. Thus, the agent in both models can at most move one position (to the left or right) in each time step.

The agent starts randomly in one of the non-goal states of the MDP. This means in the V/A model the agent can start with a velocity. This of course has an effect on its performance as it may already be on track to the goal or needs to decelerate first. There is no counterpart for this in the Kinematic/location-based (K/L) model.

Since the reward function is entirely based on the amount of time passed, no adjustments are needed. Note that the agent starting with zero velocity will be one time step slower to reach the goal in the V/A case compared to the K/L case. Thus, the comparison is not about directly analyzing the exact performance or specific information cost but rather to investigate the general behavior change of the agent within the proposed V/A model as compared to the K/L model, as well as identifying exactly when behavioral changes take place and where cognitive costs are incurred.

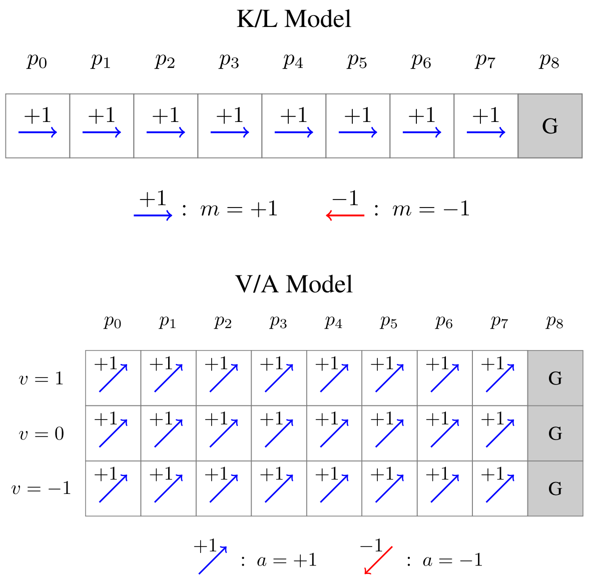
Result 1 — border goal with fully trapping goal position

We observe no significant difference in the behavior of the policy or the information cost between the K/L and the V/A model. As shown in figure 1, both policies only include a single action ($m = +1$ and $a = +1$) in all possible states, resulting in a relevant information of 0 since no states need to be distinguished from others to decide on an action to take.

We observe that none of the agents selects the ”no change”-action. In the case of the K/L model this action is suppressed by the optimality requirement because using this action would mean a “lost” time step and thus suboptimal performance. In the acceleration-based model however, the “no change”-action appears in some optimal policies if one purely optimizes in terms of value (e.g. the MDP solution with Q^*). When additionally optimizing under the relevant information constraint, this constraint reduces the set of possible optimal policies; the policy with the “no change”-action is now suppressed in favour of the policy shown in figure 1 since the former would require distinguishing a state on whether to apply $a = +1$ or $a = 0$ whereas the latter does not.

Result 2 — middle-of-the-field goal with fully trapping goal position

We see the same general behaviour as in the first setup but now the goal can be reached from two directions (see Figure 2). Both agents directly move towards the goal from their respective side. The agent now needs to distinguish at every



K/L Model
p_0 p_1 p_2 p_3 p_4 p_5 p_6 p_7 p_8
+1
G
$\xrightarrow{+1}$: $m = +1$
$\xleftarrow{-1}$: $m = -1$
V/A Model
$v = 1$
$v = 0$
$v = -1$
+1 : $a = +1$
−1 : $a = -1$

Figure 1: Top: The resulting policies of the K/L model in terms of m in which the arrows symbolize the action to take one step in the marked direction. This policy only contains a single action $m = +1$, with an information of 0 bit per step and the goal marked with the letter ”G”. Bottom: The resulting policy of the V/A model in terms of a with again an information of 0 bit per step and the goal marked with the letter ”G”. Here, the arrow indicates the immediate change in velocity and the delayed change in location induced by the action ($a = +1$).

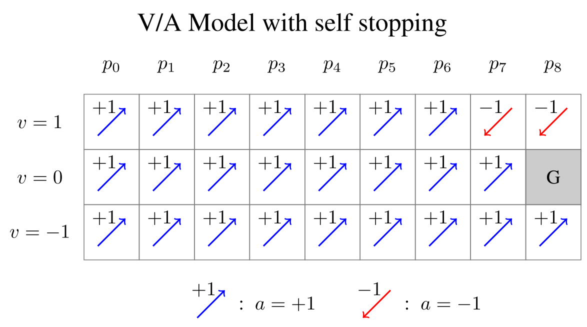

V/A Model with self stopping
p_0 p_1 p_2 p_3 p_4 p_5 p_6 p_7 p_8
$v=1$
$v=0$
$v=-1$
G
$+1$: $a=+1$
-1 : $a=-1$

Figure 3: The resulting policy of the acceleration-based model with self stopping and an informational cost of > 0 bit per step and the goal marked with the letter "G". There two states which require a different action than the rest.

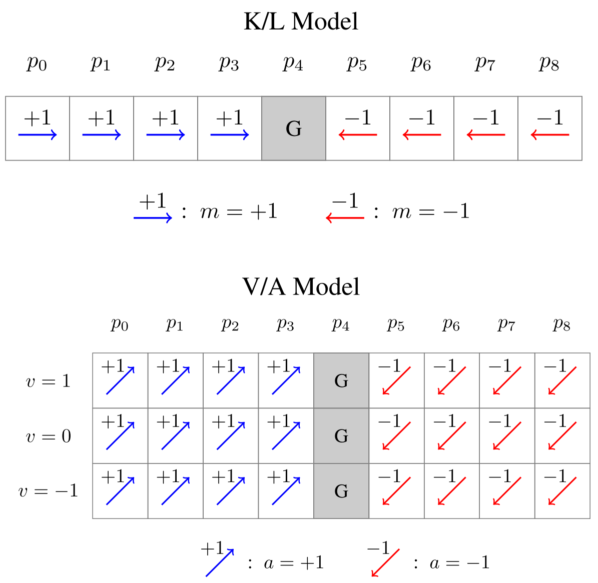


Figure 2: Top: The resulting policy of the K/L model in terms of m in which the arrows symbolize the action to take one step in the marked direction. The policy shows two equally large sets of states (1 bit per step) in which the same action is taken and the goal marked with the letter "G". Bottom: The resulting policy of the velocity/acceleration model in terms of a which also shows two equally large sets of states (1 bit per step) and the goal marked with the letter "G". Here, the arrow indicates the immediate change in velocity and the delayed change in location induced by the action.

time its state amongst two equally large sets of states which results in a relevant information of 1 bit per step. (We note here that the relevant information formalism used here assumes that the agent has no memory, so it has to "look up" its state at each decision point).

Result 3 — border goal with active stopping

Here, we see for the first time a specific behavior of deceleration and its cost near the goal because the latter can only be reached with zero velocity. Far away from the goal the agent behaves the same in all states but, once near the goal, it has to decelerate (see Figure 3). We see now a slight increase in relevant information compared to the 0 in the previous fully trapping border goal. This increase results from the two states $(p_7, +1)$ and $(p_8, +1)$ in the top right corner which require the action $a = -1$ to decelerate the agent while in the rest of the states the action $a = -1$ is taken. The exact value of information depends on the number of states because the relevant information is an average over all states. In this particular example the relevant information is 0.39 bit per step. Importantly, the agent now does not only need to move towards the goal but needs also to plan (slightly) ahead to arrive and stop once reaching the goal position.

Result 4 — middle goal with active stopping

We observe a combination of the behaviors in setups 2 and 3. Again observe that around the goal the state space is partitioned into two behaviors, now not purely based on the position but also on the velocity. In contrast to the K/L model (see Figure 2), however, we do not have the same action overall on one side of the goal (see Figure 4). Instead, we can see that the process of decelerating timely is more difficult in this scenario.

From these experiments we see that velocity-based movements only show differences in strategy if the responsibility to stop is placed on the agent itself.

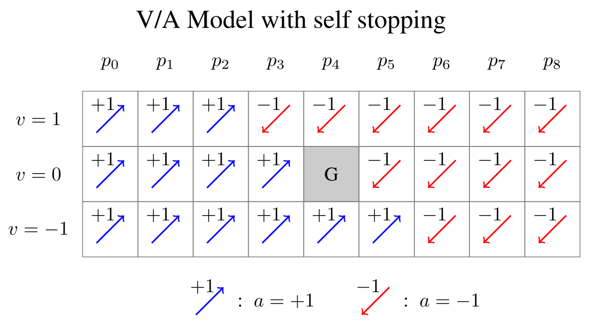

V/A Model with self stopping
p_0 p_1 p_2 p_3 p_4 p_5 p_6 p_7 p_8
$v = 1$
$v = 0$
$v = -1$
G
$+1$: $a = +1$
-1 : $a = -1$

Figure 4: The resulting policy with active deceleration. This policy also has only two equally large sets of states (1 bit per step) and the goal marked with the letter ”G”. However, we clearly see that the area around the goal is the important part.

Increasing the Maximum Velocity

In our previous experiments we limited the possible velocities in our V/A model to create agent trajectories comparable to the K/L model. In the following experiments we now allow higher velocities and thus faster movements. This has no counterpart in the K/L model without expanding the model significantly.

Again, we investigate the cost of stopping at the right position e.g. $(p_g, 0)$. The agent again starts randomly in one of the non-goal states of the MDP.

In the first experiment we restrict the agent to a maximum velocity of 2 in both directions and in the second we discuss the theoretical case of no restriction to velocity. To avoid dealing with infinite state spaces in our simple framework, we consider various maximum velocities $v_{max} = k$ where $k \in \mathbb{N}$. The agent can still only accelerate or decelerate by 1 at each time step which means it may have to overshoot the goal or hit walls if it starts with a too high velocity at the wrong location.

	Setup			Relevant Information	Braking Distance	”No Change” utilised
	Goal	vel_{max}	Self Stopping			
K/L	Border	-	-	0 bit	0	no
	Middle	-	-	1 bit	0	no
V/A Model	Border	1	No	0 bit	0	no
	Middle	1	No	1 bit	0	no
	Border	1	Yes	0.39 bit	1	no
	Middle	1	Yes	1 bit	1	no
	Middle	2	Yes	1.06 bit	3	yes
	Middle	k	Yes	1-1.5 bit	$\frac{k(k+1)}{2}$	yes

Table 1: Cognitive cost, in relevant information, braking distance, and use of the ”No-Change” action for all tested setups.

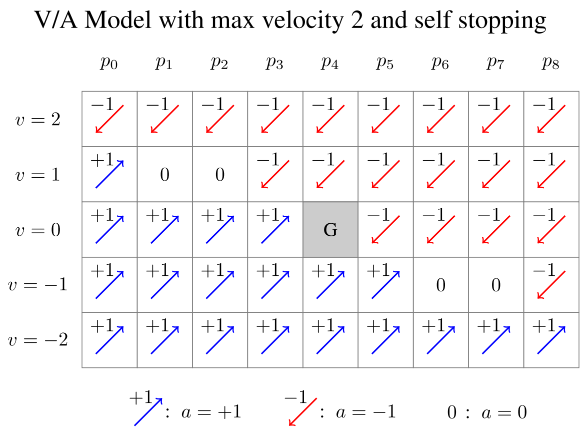

V/A Model with max velocity 2 and self stopping
p_0 p_1 p_2 p_3 p_4 p_5 p_6 p_7 p_8
$v = 2$
$v = 1$
$v = 0$
$v = -1$
$v = -2$
G
$+1$: $a = +1$
-1: $a = -1$
0 : $a = 0$

Figure 5: The resulting policy of an agent with a maximum velocity 2 and the goal marked with the letter ”G”. The cost of this policy is more than 1 bit per step because the ”no change”-action (0) is the only optimal action in four states.

Result 5 — setting the maximal velocity v_{max} to 2

We see an extension of the effects in experiment 4. The larger the velocity, the further away from the goal the deceleration process needs to be initiated to avoid overshooting the goal. We observe for the first time the necessity of a “no change”-action $a = 0$ to achieve optimal cost. In the states $(p_2, +1)$, $(p_1, +1)$, $(p_6, -1)$ and $(p_7, -1)$ in which the agent is two positions away and already moves towards the goal, the agent can neither accelerate nor decelerate without wasting time but rather has to keep its velocity steady (shown as 0 in Figure 5). This further increases the cognitive cost of deceleration and managing velocity. As a result, the policy has a relevant information of more than 1 bit per time step, again the exact increase depends on the amount of positions in the world as positions further away from the goal are not affected. For this particular setup the relevant information is 1.06 bit per step.

In summary, this experiment shows that stopping with higher velocity requires measurably more complex decision-making.

Result 6 — larger maximal velocities v_{max}

We again see a continuation of the effect in experiment 5. Higher velocities require even earlier deceleration and planning (see Figure 6 resulting in a relevant information of 1.31 for the shown example. In fact, the agent needs to start decelerating $\sum_{i=1}^{k} i = \frac{k(k+1)}{2}$ positions away from the goal, where k is the current velocity. The agent can only reduce its velocity by 1 each time step but will still be moving towards the goal, in other words the braking distance of the agent is that long. The “no change”-action appears more often and on all levels of velocity in a specific pattern: 2 state for velocity 1, 3 for velocity 2, 4 for velocity 3 and so on

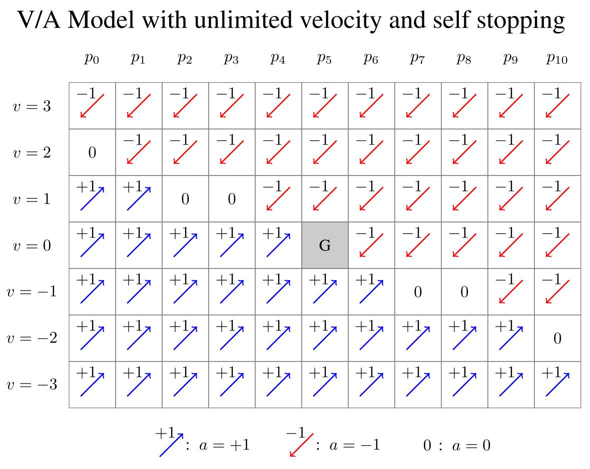

V/A Model with unlimited velocity and self stopping
p_0 p_1 p_2 p_3 p_4 p_5 p_6 p_7 p_8 p_9 p_{10}
$v=3$
$v=2$
$v=1$
$v=0$
$v=-1$
$v=-2$
$v=-3$
G
$+1$: $a=+1$
-1 : $a=-1$
0 : $a=0$

Figure 6: Part of the policy with unlimited velocities showing the longer deceleration phase and the pattern of "no-change"-actions($a = 0$) necessary to reach the goal optimally.

directly before the position in which the deceleration phase starts. For a velocity of 2 there are three states in which the agent has to maintain its velocity in order to neither stop before reaching the goal position nor overshooting it. This increases to four states when the velocity is 3 and continues growing linearly with the velocity. The number of states in which the agent has to maintain its velocity is tied to the displacement the agent will experience before its next decision — e.g. before it can start the actual deceleration phase.

Discussion

We have introduced a model for movement or navigation of an agent that extends the typically studied perspective from a kinematic to a dynamic view. This proposed Velocity/Acceleration model (V/A model) expands the agent's state space to include its current velocity. In the V/A model, the actions are accelerations which directly affect only the velocity which in turn affects the position. We took this as a step to understand the agent's dynamics when it has to contend with momentum and inertia as opposed to the typical high-friction scenarios where this is not necessary.

In the first two experiments (trapping goals) we have seen that both models effectively function in the same fashion when it is the environment that is responsible for deceleration (e.g. $\{p_g\} \times V$) (compare row 1–4 in Table 1). Once we transfer this responsibility to the agent (Experiment 3 to 6, non-trapping), the agent needs to carry out a deceleration behavior. Around the goal position the policy shows a distinctive pattern of actions to generate this deceleration behavior. Our first minimalistic model offers a glimpse into how we can model movement and understand the recent findings by Becker and Person, investigating the neural activity of a mouse reaching for an object (Becker and Person, 2019). Their results showed that mice show an increase in neural activity — which we interpret as investment of cognitive processing cost — in the cerebellum while decelerating and correcting [3].

In experiments 5 and 6 our model predicts that agents with richer velocity spaces require more cognitive cost and planning. Perhaps the introduction of more semantic actions — decelerate to zero — via options (Sutton et al., 1999), scripts (Riegler et al., 2021) or subgoals (van Dijk and Polani, 2011) might be interesting approaches to reduce the cognitive cost at the decision-making level.

Maex and Steuber have suggested that specific neural circuits in the cerebellum are capable of mathematical integration (Maex and Steuber, 2013). This would theoretically provide the computational capabilities which would allow the integration-based forward planning when velocities are involved.

The idea of the present work is to explicitly consider the necessity to manage momentum and inertia and suggest possible consequences for the cognitive processing and possibly the brain structure of the respective biological organisms as compared to organisms that live in high-friction environments where they only manage positioning directly. In particular, we propose that information-processing considerations may directly suggest evolutionary pressures towards brain structures geared towards processing of particular types of movement control information and thus help contributing to the prediction of the presence and functionality of certain components of the brain across organisms.

References

Abbeel, P., Coates, A., Quigley, M., and Ng, A. (2006). An application of reinforcement learning to aerobatic helicopter flight. *Advances in neural information processing systems*, 19.

Becker, M. I. and Person, A. L. (2019). Cerebellar control of reach kinematics for endpoint precision. *Neuron*, 103(2):335–348.

Boubaker, O. (2013). The inverted pendulum benchmark in nonlinear control theory: a survey. *International Journal of Advanced Robotic Systems*, 10(5):233.

Cover, T. M. and Thomas, J. A. (1991). Information theory and statistics. *Elements of information theory*, 1(1):279–335.

Furuta, K., Yamakita, M., and Kobayashi, S. (1992). Swing-up control of inverted pendulum using pseudo-state feedback. *Proceedings of the Institution of Mechanical Engineers, Part I: Journal of Systems and Control Engineering*, 206(4):263–269.

Huettel, S. A., Song, A. W., McCarthy, G., et al. (2004). *Functional magnetic resonance imaging*, volume 1. Sinauer Associates Sunderland, MA.

[3]Specifically, Figure 4 of Becker and Person's paper show the connection of outward velocity and neural activity

Kajita, S., Kanehiro, F., Kaneko, K., Fujiwara, K., Harada, K., Yokoi, K., and Hirukawa, H. (2003). Biped walking pattern generation by using preview control of zero-moment point. In *2003 IEEE International Conference on Robotics and Automation (Cat. No.03CH37422)*, volume 2, pages 1620–1626 vol.2.

Laughlin, S. B. (2001). Energy as a constraint on the coding and processing of sensory information. *Current opinion in neurobiology*, 11(4):475–480.

Maex, R. and Steuber, V. (2013). An integrator circuit in cerebellar cortex. *European Journal of Neuroscience*, 38(6):2917–2932.

Mitobe, K., Capi, G., and Nasu, Y. (2000). Control of walking robots based on manipulation of the zero moment point. *Robotica*, 18(6):651–657.

Niedermeyer, E. and da Silva, F. L. (2005). *Electroencephalography: basic principles, clinical applications, and related fields*. Lippincott Williams & Wilkins.

Polani, D. (2009). Information: currency of life? *HFSP journal*, 3(5):307–316.

Polani, D., Nehaniv, C. L., Martinetz, T., and Kim, J. T. (2006). Relevant information in optimized persistence vs. progeny strategies. In *In: Artificial Life X: Proceedings of the Tenth International Conference on the Simulation and Synthesis of Living Systems*. Mit Press.

Riegler, B., Polani, D., and Steuber, V. (2021). Embodiment and its influence on informational costs of decision density—atomic actions vs. scripted sequences. *Frontiers in Robotics and AI*, 8:24.

Shalev-Shwartz, S., Shammah, S., and Shashua, A. (2016). Safe, multi-agent, reinforcement learning for autonomous driving. *CoRR*, abs/1610.03295.

Sutton, R. S. and Barto, A. G. (2018). *Reinforcement learning: An introduction*. MIT press.

Sutton, R. S., Precup, D., and Singh, S. (1999). Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning. *Artificial intelligence*, 112(1-2):181–211.

Tanaka, T. and Sandberg, H. (2015). Sdp-based joint sensor and controller design for information-regularized optimal lqg control. In *2015 54th IEEE Conference on Decision and Control (CDC)*, pages 4486–4491. IEEE.

van Dijk, S. G. and Polani, D. (2011). Grounding subgoals in information transitions. In *2011 IEEE Symposium on Adaptive Dynamic Programming and Reinforcement Learning (ADPRL)*, pages 105–111. IEEE.

String: *a programming language for the evolution of ribozymes in a new computational protocell model*

Mohiul Islam[1], Nawwaf Kharma[1] and Peter Grogono[1]

[1]Concordia University, Montreal, Canada
moh_i@ece.concordia.ca, nawafkharma@gmail.com

Abstract

String is a new computer language designed specifically for the implementation of 'ribozymes', the active entities within a new (highly simplified) model of protocellular life. The purpose of the model (which is presented here, only in outline) is the study of the abstract nature of simple cellular life and its relationship to computation. This model contains passive and active entities; passive entities are data and active ones are executable data (or programs). All programs in our model are written or evolved in String. In this paper, we describe String and provide examples of both hand-written and evolved String programs belonging to different functional categories needed for cellular operation (e.g., mass transporter, information transporter, transformer, replicator and translator). Results from the evolutionary runs are presented and discussed, where almost all ribozymes reached their optimum fitness.

Introduction

The cell is the unit of all known life forms, and all cells have three common sub-systems: a heritable information-laden genome partly coding for proteins, a metabolic network of chemical reactions mostly catalyzed by protein enzymes, and a selective protein-infused lipid membrane that greatly influences material and informatic exchanges with the environment. It is generally agreed that all cells descended from a Last Universal Common Ancestor (or LUCA), which is likely to have had an embryonic form of all three sub-systems. And, LUCA must have had its own actual and abortive pre-cellular ancestors, which could legitimately be called protocells. The messy transitions from open non-living systems to (semi-) closed proto-cellular systems to LUCA and beyond are of great interest to scientists, as they offer a host of intriguing questions. However, our interest is in simple models of protocellular 'life', including models that may (/not) have had material reifications in pre-LUCA times. We believe this line of investigation will enhance our understanding of the computational nature of manifestation of life, since everything in a computational simulation must be described in purely informatic terms.

Evolving self-replicating digital organisms in the logical/informational medium (using assembly like instructions) is a well studied phenomenon in Artificial Life (Rasmussen et al., 1990; Ray, 2003; Ofria et al., 2009). Many of these automatons are developed using string molecules (bases) in artificial chemistry (Varetto, 1993; Clark et al., 2017). Recently a whole cell simulation was presented by (Das and Mitra, 2021) where protein clusters are densely connected and spatially localized, but it lacks in genetic representation.

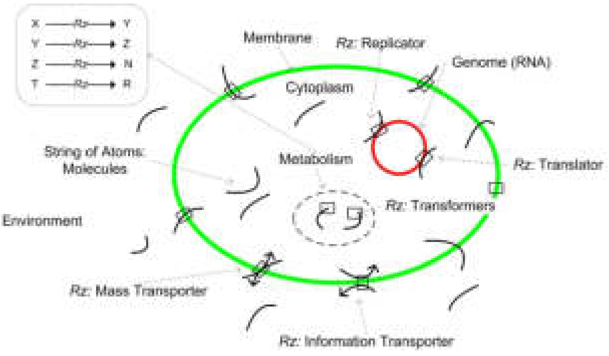


Figure 1: A highly abstracted diagram of the ProtoCell model. It shows the three sub-systems of ProtoCell: the Membrane, Metabolism and Genome.

The ProtoCell Model. We describe a computational protocell (figure 1), inspired by models of biological protocells – all of which are hypothetical – a model that is as simple as possible, but not simpler. The model aims to mimic, in computational terms, the essential processes of cellular life as a system, but without copying every little detail of current cellular biology. Ultimately, our aim is to build an autonomous computational entity that exhibits the common properties of life, which we assert are (A) self-maintenance and growth within a non-trivial dynamic environment, leading to (B) conservative, but imperfect self-reproduction, generating (C) a diverse population, which under selection pressure leads to inter-generational adaptation (i.e., evolution). These are more than the necessary properties of life as, for example, a cell that fails to reproduce is still alive, but these appear to be necessary for the phenomenon of life to exist and persist. The current ProtoCell model has three interacting sub-systems, each with specific central functions, which are thought to be sufficient but possibly not necessary for the emergence of properties A-C above. The sub-systems are (1) the genomic sub-system (or Genome); (2) the metabolic sub-system (or Metabolism); (3) the membrane sub-system (or Membrane).

The Genome functions as inherited 'institutional memory' influencing, both directly and indirectly, the operations taking place in a cell. The metabolism is the 'cell's factory', through which a cell is able to make, break and re-cycle those molecules, including energy-carrying molecules, that a cell needs to survive and proliferate, but is unable to efficiently get from the environment. The Membrane is essentially the cell's 'selective boundary', which maintains certain working conditions, such as molecular concentrations and temperature, within optimal or at least survivable ranges; the exportation of waste and importation of nutrients is central to that work.

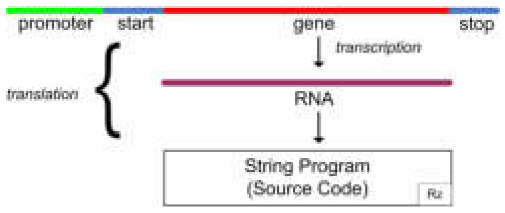


Figure 2: Gene Structure and Translation

In ProtoCell, everything is made of atoms and energy. (A) Atoms are the indivisible building blocks of molecules. From a computational perspective, molecules are strings of data atoms from the set c,h,n,o,p,s linked by bonds with energies, which are a function of the bonded atoms. Atoms and most molecules are 'passive' entities that do not enter any kind of reaction without the aid of the 'active' entities: the ribozymes. (B) Energy in ProtoCell has two forms: 'diffuse' and 'concentrated'. Diffuse energy is 'heat', which is either provided from the external environment or is given out by exothermic chemical reactions. In contrast, concentrated energy is stored in the 'bonds' between the atoms of molecules, which in ProtoCell means the transformation (making/breaking) and movement of molecules (= data).

All operations inside a ProtoCell – including its membrane – are carried out by 'ribozymes.' A ribozyme is an 'RNA' molecule, a string of nucleotides (A, U, C, G) made, as all other molecules, from strings of atoms. This RNA molecule 'folds' into an active molecule, which is a computer program (in String) capable of processing passive molecules. Ribozymes can be divided into a small number of 'pure' classes – as a ribozyme can deliver a mix of functionalities – that include (A) Transformers, which are analogous to typical 'enzymes' catalyzing the various reactions of the metabolism; (B) Translators, which transcribe (figure 2) a 'gene' into an RNA molecule, which then 'folds' into a working ribozyme/program; (C) Transporters, which are embedded in the membrane and selectively pass atoms/molecules or information across it; (D) Replicators, which are similar to translators, except that they make a mutated copy of the whole genome, to be passed on to a daughter ProtoCell.

The goal of ProtoCell is to develop a computational protocell model consisting of the three essential inter-dependent sub-systems of a cell (membrane, genome, metabolism). A distinguishing feature of the model is a new programming language (String), which is designed and will be developed to maximize the robustness and evolvability (Hu and Banzhaf, 2018) of programs (= ribozymes) written in it. In fact, there should not exist a genetic encoding that maps into invalid (i.e., unexecutable) programs and, the genetic encoding of a program should be both robust in the presence of mutations, and (theoretically) evolvable into any other program (needed by the cell). In the following sections, we describe String, and present examples of different types of hand-written and evolved ribozymes (in String).

String, the Language

A String program is a sequence of symbols. It follows from the discussion above that a genome can be interpreted as a program. This is the basis of the ProtoCell simulation. Like other programming languages, String is defined by an instruction set. This might suggest that String programs are fragile in the same way as programs written in other languages: a small change to the program results in a "syntax error" or "semantic error" that renders the program non-executable. String, however, is designed to be extremely robust: arbitrary sequences of symbols can be interpreted without failure and, in the worst case, parts of the program have no effect. This robustness ensures that all programs, including mutated programs, will execute without "crashing", although they are not guaranteed to do anything useful.

$$
\begin{aligned}
\mathit{Program} &= \{\ \mathit{Operator}\ \{\mathit{Operand}\ \}\ \}\ .\\
\mathit{Operator} &= \texttt{cut} \mid \texttt{emit} \mid \texttt{joinl} \mid \texttt{joinr} \mid \texttt{jump} \mid \\
&\quad \texttt{jumpf} \mid \texttt{jumpt} \mid \texttt{match} \mid \texttt{move} \mid \\
&\quad \texttt{sensee} \mid \texttt{sensei} \mid \texttt{shiftl} \mid \texttt{shiftr} \mid \\
&\quad \texttt{sitee} \mid \texttt{sitei} \mid \texttt{show}\ .\\
\mathit{Operand} &= \mathit{Pattern} \mid \mathit{Operator} \mid [\ \mathit{Sign}\ ]\ \mathit{Number}\ .\\
\mathit{Pattern} &= <^{+}\ \{\ \mathit{Atom} \mid \mathit{Wild}\ \}\ >^{+}\ .\\
\mathit{Atom} &= \texttt{c} \mid \texttt{h} \mid \texttt{n} \mid \texttt{o} \mid \texttt{p} \mid \texttt{s}\ .\\
\mathit{Wild} &= \texttt{?} \mid \texttt{*}\ .\\
\mathit{Sign} &= \texttt{+} \mid \texttt{-}\ .\\
\mathit{Number} &= 0 \mid 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7\ .
\end{aligned}
$$

Syntax. In String grammar (presented below) the symbol x^+ stands for "one or more instances of x". Each String program is a collection of operators with operands. There are 16 operators with three operand types *Pattern*, *Operator* and signed numbers.

There are eight registers, denoted by a digit in $\{0, 1, \ldots, 7\}$. Most ribozymes will need less than eight registers, but we provide eight for particularly complex ribozymes. A register stores a molecule. There are three kinds of registers. An *external* register provides access to the environment (figure: 1) of the cell while an *internal* register provides access to the cytoplasm (figure: 1). Other registers store molecules within the ribozyme.

Semantics. Each instruction can accept zero, one, or two operands. If an operator does not have one of the choices of operand given in table 1, it has no effect. Here we describe the convention used in table 1: X and Y stand for arbitrary registers Rn. A register is either empty or contains a molecule. "$X = E \longrightarrow$ actions" means that register X has value E before the operations on the right of the arrow are performed. $X\frown Y$ stands for the concatenation of X and Y. E.g., if $\texttt{R1} = \texttt{chn}$ and $\texttt{R2} = \texttt{ops}$, then $\texttt{R1}\frown\texttt{R2} = \texttt{chnops}$. P stands for a pattern. $X \sim P$ stands for the value (in $[0, 1]$) obtained by matching molecule X and pattern P. In all cases, $X \sim P = P \sim X$. $X \equiv M$ stands for the value obtained by comparing molecule X and molecule M. An exact match yields 1; anything else yields 0. In all cases, $X \equiv M = M \equiv X$. $X := \varnothing$ means that the contents of register X are emitted into the environment and X is left empty.

Operator	Operand	Description
`cut`		$\texttt{R0} = L\frown R \longrightarrow \texttt{R0} := L$; $\texttt{R1} := R$.
`cut`	X	$X = L\frown R \longrightarrow X := L$; $\texttt{R1} := R$.
`cut`	$X\ Y$	$X = L\frown R \longrightarrow X := L$; $Y := R$.
`emit`		Move contents of all registers into the environment.
`emit`	X	Move contents of register X into the environment.
`emit`	$X\ Y$	Move contents of registers X and Y into the environment.
		`joinl` / `joinr`
`join`*d*		$\texttt{R0} := \texttt{R0}\frown\texttt{R1}$; $\texttt{R1} := \varnothing$. / $\texttt{R0} := \texttt{R1}\frown\texttt{R0}$; $\texttt{R1} := \varnothing$.
`join`*d*	X	$\texttt{R0} := \texttt{R0}\frown X$; $X := \varnothing$. / $\texttt{R0} := \texttt{X}\frown R0$; $X := \varnothing$.
`join`*d*	$X\ Y$	$X := X\frown Y$; $Y := \varnothing$. / $X := Y\frown X$; $Y := \varnothing$.
`jump`*b*	N	Transfer control N operations or to the end of the program.
`jump`*b*	O	Transfer control to the next instance of operation O.
`jump`*b*	$O\ N$	Transfer control an instance of operation O as defined below.
`match`		$\text{CR} := \texttt{R0} \equiv \texttt{R1}$.
`match`	X	$\text{CR} := \texttt{R0} \equiv \texttt{X}$.
`match`	$X\ Y$	$\text{CR} := X \equiv Y$.
`match`	$X\ P$	$\text{CR} := X \sim P$.
`match`	$P\ X$	$\text{CR} := P \sim X$.
`move`		Emit R1; $\texttt{R1} := \texttt{R0}$; $\texttt{R0} := \varnothing \equiv \texttt{move R0 R1}$.
`move`	Y	Emit Y; $Y := \texttt{R0}$; $\texttt{R0} := \varnothing \equiv \texttt{move R0}\ Y$.
`move`	$X\ Y$	Emit Y; $Y := X$; $X := \varnothing$.
`sense`*d*	P	Set CR depending on whether there are molecules matching P in the cytoplasm ($d = \texttt{i}$) or the environment ($d = \texttt{e}$).
`shift`*d*		Move the cursor of R0 to the extreme left/right end.
`shift`*d*	X	Move the cursor of X to the extreme left/right end.
`shift`*d*	P	Search left/right for the best match $\texttt{R0} \sim P$ and set CR.
`shift`*d*	$X\ P$	Search left/right for the best match $X \sim P$ and set CR.
`shift`*d*	$P\ X$	Search left/right for the best match $P \sim X$ and set CR.
`site`*d*	X	Designate X as an internal/external site. CR is not changed.
`site`*d*	$X\ P$	Designate X as an internal/external site and look for a molecule that matches P to bind to it. Set CR to 1 if a match is found, 0 otherwise.
`site`*d*	$X\ Y$	Designate X as an internal/external site and look for a molecule that matches the molecule in Y to bind to it. Set CR to 1 if a match is found, 0 otherwise.

Table 1: String Instructions

Pattern Matching. The operations `jump`, `match`, `sense`, `shift`, and `site` accept patterns as operands. A *pattern* is a string that may contain atoms (`c`, `h`, `n`, `o`, `p`, `s`) and wild cards (`?`, `*`). In general, when a molecule is matched to a pattern, `?` matches exactly one atom and `*` matches any sequence of atoms.

Operations. We describe the operations in alphabetical order. Most operations set the value of the condition register, CR, which almost always is set to either 0 or 1.

The **Cut** operation splits a molecule into two parts and stores them in separate registers. **Emit** disposes of molecules into the environment. If the register is an external register, it is part of the external environment. If it is an internal register, it is the cytoplasm. There are two types of **Join** operations that concatenate groups: `joinl` (join left) and `joinr` (join right). The second operand is joined to the first in the manner suggested by the qualifier. In table 1, `jump`*b* stands for all three possibilities: $b \in \{\varepsilon, \texttt{f}, \texttt{t}\}$, where ε stands for the empty string. **Match** compares two strings and sets CR accordingly. If either or both of the strings are patterns (i.e., contain "?" or "*"), then the simulator performs a pattern match (indicated by $\sim$) yielding a result in $[0, 1]$. **Move** is used to transport molecules around the cell or, more precisely, from one register to another. **Sense** is used to detect the presence of molecules outside or inside the cell. The **shift** operations have two forms: `shiftl` (shift left) and `shiftr` (shift right). Their effect is to move the cut point of the molecule in the specified direction. There are two operations that define binding **sites**: `sitee` (external) and `sitei` (internal).

Translation. *Translation* is (for our purposes) the process of constructing a ribozyme (program) from RNA (figure 2). Although RNA is ultimately composed of atoms, this section treats it as a sequence of bases.

Gene Structure. A gene is a sequence of bases with the following structure:

$$\mathit{Gene} = \mathit{Operation}^* \ ;$$
$$\mathit{Operation} = \mathit{Junk}\ \mathit{OpCode}\ (\mathit{Junk}\ \mathit{Type}\ \mathit{Value}\ )^* \ .$$

A gene consists of zero or more operations. An operation has an opcode followed by zero or more operands. An operand has a type and a value. Opcodes and types may be preceded by *junk*. Junk means simply any sequence of base pairs that cannot be interpreted as an opcode or type. Whenever junk is allowed in the gene, the translator performs a matching operation which works as suggested by the following pseudo-code:

```
match($g[i]$) ≡ if $g[i,i+7] \in \mathrm{nb}_2$(opcodes[$o$])
                    return opcode opcodes[$o$]
                else if $g[i,i+4] \in \mathrm{nb}_2$(types[$t$])
                    return type types[$t$]
                else $i := i+1$ .
```

The procedure `match` takes as input a gene g with cursor position at base i. The sequence $g[i,i+7]$ consists of 8 bases starting at base i; the procedure checks whether this sequence is in the 2-neighbourhood[1] of each opcode and returns a matching one. If no opcodes match, the procedure checks with the 5-base sequence $g[i,i+4]$ within the 2-neighbourhood of an operand type, returning the matching type if there is one. No junk is allowed between an operand type and its value. If there is no matching type or opcode, the procedure tries again at $g[i+1]$ and continues until it finds a match or reaches the end of the gene.

Encoding. This section explains how a String program is encoded as a sequence of bases. Each **operator** has an 8-base code, as shown in Table 2. The codes are chosen so that $\mathrm{dist}(o,o') \geq 5$ for any pair of operators, with $o \neq o'$.

Genetic encoding	Operator
AUCGAUCG	cut
UCGAUCGA	emit
CGAUCGAU	joinl
GAUCGAUC	joinr
ACACACAC	jump
UGUGUGUG	jumpf
CACACACA	jumpt
GUGUGUGU	match
AAAAGGGG	move
GGGGCCCC	sensee
CCCCUUUU	sensei
UUUUAAAA	shiftl
AAAUAUUU	shiftr
GGAAACGU	sitee
ACGUCAUG	sitei
CAUGAUGA	show

Table 2: Operator encoding

Genetic encoding	Operand Type
AAAAA	Register
UUUUU	Pattern
CCCCC	Operator
GGGGG	Signed number

Table 3: Type encoding

Genetic encoding	Atom
AUC	c
CAU	h
GCU	n
CUG	o
CGA	p
UAG	s
UUU	*
AAA	?
CCC	>
GGG	>

Table 4: Pattern Encoding.

Each **type** has a 5-base code, as shown in Table 3. The codes are chosen so that $\mathrm{dist}(t,t') = 5$ for any pair of operators with $t \neq t'$. As mentioned above, no junk is allowed between the type and **value** of an operand. For registers and the signed numbers required for `jump` operations, we assign the following numerical values to bases: $\mathtt{A} = 0$; $\mathtt{U} = 1$; $\mathtt{C} = 2$; $\mathtt{G} = 3$. The total v obtained by adding the base values varies from $\mathtt{AAAAA} \to 0$ to $\mathtt{GGGGG} \to 15$. We use $v/2$, giving values in $[0,7]$. The **register** $r \in [0,7]$ is encoded as a 5-base sequence b_1, b_2, b_3, b_4, b_5 with $r = \frac{b_1+b_2+b_3+b_4+b_5}{2}$. **Signed numbers** use the same encoding as registers, but with an offset of -3 to allow negative values. **Operators** are encoded with the same 8-base codes as in figure 2. **Patterns** consist of the atoms `c,h,n,o,p,s` and the special symbols `*,?,>`.

Algorithm 1 Simple genetic programming algorithm

```
procedure GeneticProgramming
    $population = initializePopulation()$
    $generation = 0$
    while $generation < maxGenerationNo$ do
        $parentPool = parentSelection()$
        $childList = createChildPopulation(parentPool)$
        $childrenSelection(population, childList)$
        $generation = generation + 1$
    end while
end procedure
```

Evolution of Ribozymes

To develop an efficient evolutionary algorithm for the evolution of String programs, we evolved instances

[1]The *k-neighbourhood* of a sequence g, written $\mathrm{nb}_k(g)$ is the set of sequences within k point mutations of g.

(programs) of five ribozyme categories using a simple Genetic Programming (GP) algorithm (algorithm 1) (Koza, 1992, 1994; Krawiec, 2016). The parameters used for GP for an individual category are listed in table 7.

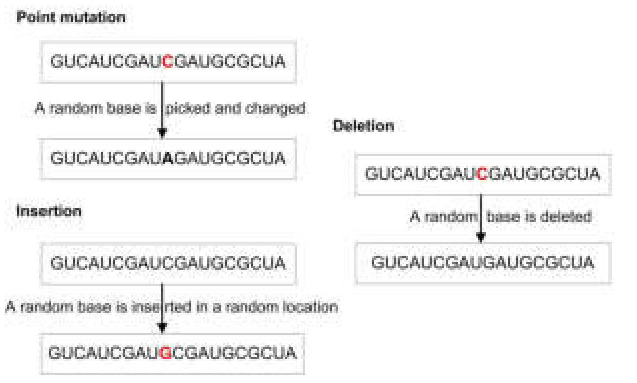
Simple GP Algorithm

The simple GP algorithm (algorithm 1) that we used to evolve ribozymes, begins by initializing a population of genes. This initialization happens by continuously generating random bases (`A`, `U`, `C` & `G`) and concatenating them, up to a fixed length. The maximum length of each gene is presented in table 7. Once a population is generated, the next step is to select a parent population. We use tournament selection fot that purpose. Next, we generate an offspring population from the parents, using several variation operators. We used a crossover and three different mutation operators for doing that.

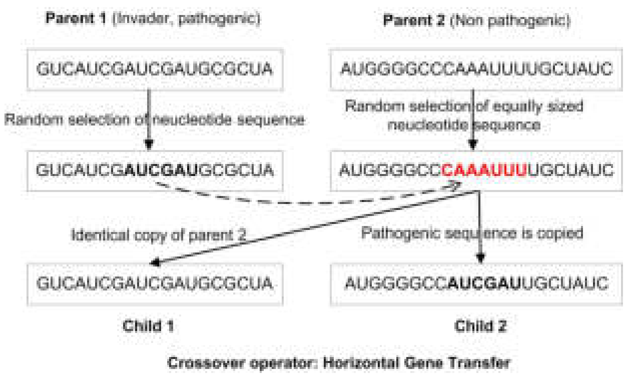
Algorithm 2 Mutation

```
 1: procedure POINTMUTATION(gene)
 2:     location ← Random location between 0 to max gene length
 3:     newChar ← Random nucleotide different than in location of gene.
 4:     Replace newChar in location of gene.
 5: end procedure
 6: procedure INSERTION(gene)
 7:     location ← Random location between 0 to max gene length
 8:     newChar ← Random nucleotide different than in location of gene.
 9:     Insert newChar in location of gene.
10: end procedure
11: procedure DELETION(gene)
12:     location ← Random location between 0 to max gene length
13:     Delete nucleotide at location of gene.
14: end procedure
```

Horizontal Gene Transfer Crossover. In Nature, Horizontal Gene Transfer (HGT) happens mostly in bacteria but also in other species. In bacteria a section of the genome of a pathogenic bacteria is inserted into a non-pathogenic one, resulting in the reproduction of two daughter cells. We designed a crossover operation inspired by this process. Here, two parents are selected from the parent population, and a random portion of the gene from one parent is placed at a random location in the gene of the other parent, creating a child that is identical to a parent, and another containing a modified genome, part of which is from the other parent (figure 3(b), algorithm 3).



(a) Three different mutation operators used in the GP algorithm. Point mutation, Insertion and Deletion.



Parent 1 (Invader, pathogenic)
Parent 2 (Non pathogenic)
GUCAUCGAUCGAUGCGCUA
AUGGGGCCCAAAUUUUGCUAUC
Random selection of neucleotide sequence
Random selection of equally sized neucleotide sequence
GUCAUCGAUCGAUGCGCUA
AUGGGGCCCAAAUUUUGCUAUC
Identical copy of parent 2
Pathogenic sequence is copied
GUCAUCGAUCGAUGCGCUA
AUGGGGCCAUCGAUUGCUAUC
Child 1
Child 2
Crossover operator: Horizontal Gene Transfer

(b) The crossover operator used in GP algorithm is inspired by bacteria's horizontal gene transfer.

Figure 3: Variation operators used in Genetic Programming

Mutation. We use three different mutation operators (figure 3(a), algorithm 2), replacement (or point) mutation and insertion and deletion mutations; some are applied more often than others (table 7). In **Point Mutation** a base within the gene is selected at random, and changed to a different base vale. **Insertion** happens when a base is selected at random and hence, is inserted at a random location within the gene. For **deletion** we pick a random base from the gene and simply delete it.

Algorithm 3 HGT Crossover

```
1: procedure APPLYHGTCROSSOVER(parent1, parent2, childList)
2:     subGene ← Find random sub gene (sub string) in parent1.gene
3:     location ← Find random location in parent2.gene
4:     child1.gene ←Position subGene in location of parent2.gene replacing equal number of bases.
5:     child2.gene ← parent2.gene
6:     Add child1 and child2 to childList return childList
7: end procedure
```

Algorithm 4 Ribozyme Fitness

```
1:  procedure GETRIBOZYMEFITNESS(gene)
2:      cyt ← Create instance of Cytoplasm
3:      ect ← Create instance of Environment
4:      cell ← Create Cell instance using cyt and ect
5:      Insert input molecules in cyt
6:      rib ← Create Robosome instance using gene
7:      Transcribe gene using ribosome to produce ribozyme
8:      Insert ribozyme (String code) into cytoplasm
9:      Execute the ribozyme (String program)
10:     Test output molecule type and count in cyt and ect to
    compare with expected output using MED distance.
        return sum total of MED distance
11: end procedure
```

Table 5: Mass transporter: ideal vs evolved program

Ideal Mass Transporter	**Evolved** (Normalized fitness: 1)
sitee 0 <chnops*> sitei 0	sitee +1 <c*> joinr

Table 6: Mass transporter test cases.

Input mol.	Output mol.
chnops	chnops
chnopsc	chnopsc
chnopsch	chnopsch
chnopschn	chnopschn
chnopschno	chnopschno
chnopschnop	chnopschnop
chnopschnops	chnopschnops
hnops	‘’
nops	‘’
ops	‘’
ps	‘’
s	‘’

Fitness evaluation. To evaluate fitness of an individual, the algorithm (algorithm 4) places one or more molecules at an input site, and expects a certain molecule at an output site of the program, at the end of execution. Multiple test cases are used to evolve each program. The various input and output molecules used as test cases for every type of ribozyme are listed in tables 6, 9, 10, 13 & 14. The output molecules from each evolved program is compared with the expected molecules using MED (Levenshtein, 1966) distance. The sum of distances between the actual and expected molecules, over all test cases is normalized, and the resulting value is considered the fitness of the evaluated program.

Transporter. In nature, there are different kinds of transporters, some passive, others active, some have pores in them and others not. A ProtoCell has a much simpler organization where pure transporters may transport molecules or information. Mass transporters actually move a molecule from one site to another, typically from outside/inside the cell to inside/outside the cell, respectively. Information transporters do not move molecules; they typically recognize the binding of a molecule to an internal/external site and hence, act on another molecule bound to an opposite site, in response. As such, what moves here is a signal, not a molecule.

Table 8: Replicator and Translator: ideal vs evolved program

Ideal Replicator	**Evolved** (Normalized fitness: 0.97)	Ideal Translator	**Evolved** (Normalized fitness: 0.97)
sitei 6 <*ccohnn cconncchnn cchonnccohnn*> shiftr 6 <ccohnn cconncchnn cchonnccohnn> cut 6 0 joinr 0 6 shiftl 0 shiftr 0 <cc*nn> jumpf sitei 2 cut 0 1 sitei 5 0 jumpf sitei 1 joinr 4 5 joinr 2 0 move 1 0 shiftr 0 <cc*nn> jumpf sitei 2 cut 0 1 sitei 5 0 joinr 4 5 joinr 2 0 move 1 0 jump shiftr -2 sitei 5 0 joinr 4 5 joinr 2 0	sitei 3 <*nc*> sitei 0 <*co*> sitei 1 <*> joinr sitei 1 <*> joinr move 7 sitei 0 <*oh*> sitei 1 <*> sitei 1 <*o*> joinr sitei 1 <*co*> sitei 1 <*co*> joinr sitei 1 <*> joinr sitei 1 <*oh*> joinr sitei 1 <*> joinr sitei 1 <*co*> joinr sitei 1 <*o*> sitei 1 <*> joinr sitei 1 <*oh*> joinr sitei 1 <*> joinr sitei 1 <*co*> joinr sitei 1 <*> joinr sitei 1 <*oh*> joinr move move 7 <c> joinr sitei 1 <*> jumpt -2 cut joinl move 3 shiftr <ccho> shiftr <co> shiftr <c?oh> cut joinr 1	sitei 6 <*ccohnn cconncchnn cchonncconn *cchonncchnn ccohnncconn cchonn*> shiftr 6 <ccohnn cconncchnn cchonncconn> shiftr 6 <cc*nn> shiftr 6 <cc*nn> shiftr 6 <cc*nn> shiftr 6 <cc*nn> shiftr 6 <cc*nn> cut 6 0 shiftr 0 <cchonn cchnnccohnn cconncchonn> cut 0 7 shiftl 0 shiftr 0 <cc*nn> jumpf sitei 2 cut 0 1 sitei 5 0 jumpf sitei 1 joinr 4 5 joinr 2 0 move 1 0 shiftr 0 <cc*nn> jumpf sitei 2 cut 0 1 sitei 5 0 joinr 4 5 joinr 2 0 move 1 0 jump shiftr -2 sitei 5 0 joinr 4 5 joinr 2 0 joinr 6 2 joinr 6 7	sitei 2 <cc?nn> sensee sitei 2 <cc?nn> joinr 1 <h> joinl 2 sitei 2 <cc??nn> sitei 0 2 sitei 2 <cc?nn> sitei 2 <cc?nn> joinl 2 1 sitei 2 2 joinl 2 jumpf 4 sitei 2 <cc?hnn> sitei 1 <cc?onn> joinl 1 2 sitei 2 <cc?nn> joinr 1 ? joinl 2 sitei 2 <cco?n> joinl 2 jumpf 4 sitei 2 <cc?h?n> joinl 2 0 sitei 0 <cc?onn> joinl 2 sitei 2 <cc?nn> joinr 1 joinl 2 sitei 2 <cco?n> joinl 2 jumpf 4 sitei 2 <cc?h?n> joinl 2 0 sitei 0 <?c?onn> joinl 2 sitei 2 <cco?n> joinl 2 0 joinl 2 jumpf 4 sitei 2 <cc?hnn> joinr 2 match jumpf sensee -1

We designed a mass transporter using String (table 5), which accepts molecules of the pattern `chnops*`, and simply moves it from the external environment register `0` to the internal one, from which it is emitted to the cytoplasm. The ideal information transporter (table 11) senses the molecule `chnops` in the environment. When it finds it's presence, the transporter performs a transformation in the cytoplasm, by collecting and joining two molecules of the pattern `c*` and `n*`, and releasing it back in the cytoplasm. Table 6 & 9 contain the mass and information transporter test cases respectively.

Transformer. In Nature, real transformers are enzymes, and they accelerates the rate of a chemical reaction, which results in the re-organization of the atoms in the reactants to make the products. In ProtoCell the pro-

Table 7: Parameters for Genetic Programming used for each Ribozyme

Ribozyme type	Population	Gene Size	Parent Pool Size	Tournament Size	Reproduction Ratio	Crossover Ratio	Mutation Ratio	Point Mutation Ratio (% of Total Mutation)	Insertion Ratio (% of Total Mutation)	Deletion Ratio (% of Total Mutation)	Mutation Steps	Number of Generation	Child population size multiple
Mass Transporter	1000	100	40%	50	20%	0%	100%	80%	10%	10%	20	1000	15
Information Transporter	10000	150	40%	500	20%	0%	100%	80%	10%	10%	20	4000	15
Transformer	10000	150	40%	500	20%	0%	100%	80%	10%	10%	20	4000	15
Replicator	1000	500	20%	50	20%	20%	80%	70%	16%	14%	20	10000	1
Translator	1000	500	20%	50	20%	20%	80%	70%	16%	14%	20	10000	1

Table 9: Information Transporter Test Cases

Ouput Mol	Input mol 1	Input mol 2	Expected input mol
chnops	chno	no	chno no
chnops	chnops	nops	chnops nops
chnops	chnopsch	nopsch	chnopsch nopsch
chnops	chnopschno	nopschno	chnopschno nopschno
chnops	chnopschnops	nopschnops	chnopschnops nopschnops

Table 10: Transformer Test Cases

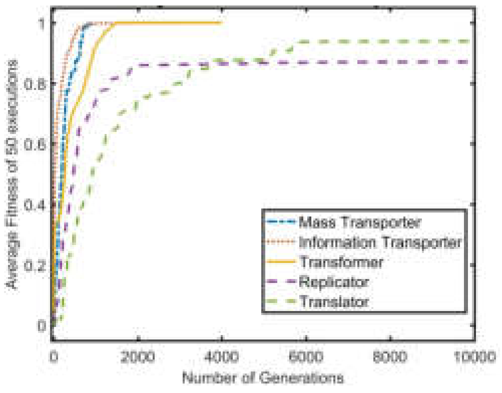
Input mol 1	Input mol 2	Input mol 3	Expected input mol
chno	no	p	chno no p
chnops	nops	ps	chnops nops ps
chnopsch	nopsch	psch	chnopsch nopsch psch
chnopschno	nopschno	pschno	chnopschno nopschno pschno

Table 11: Information Transporter and Transformer: ideal vs evolved program

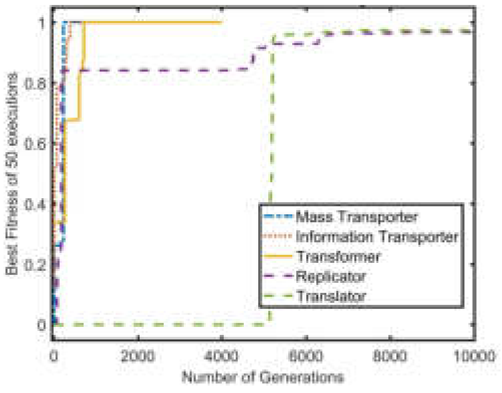
Ideal Information Transporter	**Evolved** (Normalized fitness: 1)	**Ideal Transformer**	**Evolved** (Normalized fitness: 1)
sensee <chn ops> jumpf show sitei 0 <c*> sitei 1 <n*> joinr 0 1 show 0	sitei +1 <n*> sitee 4 <*> jumpf <c?ch> sensee 7 <?shp> sitei +0 <**> joinr	sitei 0 <c*> sitei 1 <n*> sitei 2 <p*> joinr 0 1 joinr 0 2	sitei +1 <p*> sitei +0 <n*> shiftl 3 2 sitei 7 <*> joinr joinr 7 <c*po ccpnpcnc ?cppsc>

cess of transformation starts when a transformer binds and then accepts the reactant molecules. In table 11 we present a simple transformer where the ideal program accepts the patterns `'c*'`, `'n*'` and `'p*'`. Hence, the molecule with the first pattern is joined with the second molecule, and their product is then ligated to the third molecule. Finally the resulting large molecule is emitted in to the cytoplasm. The test molecule sequences for the transformer test cases are listed in table 10.

Replicator. In Nature, cells have DNA genomes. In contrast, ProtoCell has an RNA genome, which it must replicate, in order to reproduce. This process is carried out by a replicator program written in String. The ideal String replicator presented in table 8 initiates the replication process upon encountering the origin of replication (OR) sequence of `GUCAG`. When the OR sequence is identified, the program continues to read chunks of the pattern `cc*nn`, from the original sequence, and keeps



(a) Average fitness evolution of Ribozymes



(b) Best fitness evolution of Ribozymes

Figure 4: Results of evolution of Mass and Information Transporter, Transformer, Translator and Replicator. In these figures the x-axes show the number of generations, while the y-axes show the increase in normalized fitness.

joining copies of the identified molecules to the new sequence. This continues until the complete sequence is copied, when the program breaks and releases both the original genome and an identical copy into the cytoplasm. Table 13 presents all the replicator sequences used in evolution of a replicator. String programs are only capable of identifying atoms and strings of atoms (molecules); table 12 shows how the conceptual four

Molecule (Base)	Atoms
A	cchonn
C	cchnn
G	ccohnn
U	cconn

Table 12: Atomic encoding of molecules (bases)

Input mol (*OR*: GUCAG)	**Output mol \Count**	**A**	**U**	**C**	**G**	**Actual Output**
AGCUAG **GUCAG**CU	**GUCAG**CU AGCUAG	3	3	3	4	GU**G**AGCU**C**GCUA**A**
GCAUCA **GUCAG**G	**GUCAG**G GCAUCA	3	2	3	4	GU**G**AG**C**GC**C**U**A**A
AGCUAG **GUCAG**GUA	**GUCAG**GUA AGCUAG	4	3	2	5	G**AU**AG**C**U**CG**G**G**UA**A**
GCAUCA **GUCAG**AUCG	**GUCAG**AUCG GCAUCA	4	3	4	4	G**AU**AG**C**UCG**CGC**U**A**A
UGACAG **GUCAG**AUCGAU	**GUCAG**AUCGAU UGACAG	5	4	3	5	**U**U**A**AGAU**A**GAU**CGCGC**G
AGAGCU **GUCAG**AUCGAUCG	**GUCAG**AUCGAUCG AGAGCU	5	4	4	6	GU**A**AGAU**A**GAUCG**C**G**C**GCU
AGCUAG **GUCAG**AUC GAUCGAUCG	**GUCAG**AUCGAU CGAUCG AGCUAG	6	5	5	7	GU**A**AGAU**A** GAUCGA**G**CG**CC**CU**U**G
UGCAGU **GUCAG**AUC GAUCGAUGCCGCAU	**GUCAG**AUCGAUC GAUGCCGCAU UGCAGU	6	7	7	8	GU**A**AGAU**A**GAUCGA**GCG** C**C**C**CC**U**UUU**G**G**

Table 13: Comparison between the expected vs the actual output of the evolved replicator. **Bold** molecules in the last column (Actual Output) differentiates the expected.

Input mol (Start: GUCAU End: ACGUA)	**Output mol \Count**	**A**	**U**	**C**	**G**	**Actual Output**
AGCUAG **GUCAU** AUG **ACGUA** GCUAAGC	AUG	1	0	1	1	AUG
UGCAUA **GUCAU** AGUU **ACGUA** AGCUAG	AGUU	1	2	0	1	**UUAG**
CGAUCC **GUCAU** CUAGG **ACGUA** GCAUCA	CUAGG	1	1	1	2	CUAGG
AGAGCU **GUCAU** UGCAGU **ACGUA** UGACUA	UGCAGU	1	2	1	2	U**U**CAG**G**
GCAUCA **GUCAU** CAUAUCG **ACGUA** CUGAUA	CAUAUCG	2	2	2	1	**G**A**C**AUC**U**
UGACAG **GUCAU** CAGAUCGAU **ACGUA** GCAUCA	CAGAUCGAU	3	2	2	2	**G**A**C**AUCGAU
AGAGCU **GUCAU** CGAAUC GAUCG **ACGUA** UGACUA	CGAAUCGAUCG	3	3	3	2	**GAC**AUCGAUCG
UGCAUA **GUCAU** GGAAUC GAUCGAUGC **ACGUA** AGCUAG	GGAAUCGAUCGAUGC	4	3	3	5	G**AC**AUCGAUCGAUG**G**
UGCAGU **GUCAU** UACAUC GAUCGAUGCCGCAU **ACGUA** AUGCUA	UACAUCGAUCGAUGCCGCAU	5	5	6	4	**G**ACAUCGAUCGAUGCC**U**CAU

Table 14: Comparison between the expected vs the actual output of the evolved translator. **Bold** molecules in the last column (Actual Output) differentiates the expected.

bases are encoded as strings of atoms.

Translator. In Nature, proteins expression starts with DNA to RNA transcription followed by RNA to protein translation. In contrast, in the ProtoCell model, a gene is an RNA sequence (on the genome) sandwiched between universal start and stop sequences, and its translation is defined as making (or 'transcribing') an RNA copy of the gene by a 'translator' ribozyme (or program). Hence, the RNA copy autonomously 'folds' into its own active ribozyme (also, a program), via the application of (hidden) laws of Nature within the simulation (figure 2). The ideal translator in table 8 starts reading the genome until it finds the start sequence. In our test cases (table 14) we used the start pattern GUCAU. At this point the translator will start copying the RNA gene to a new molecule by identifying the pattern cc*nn, one molecule at a time. This continues until the stop pattern ACGUA is found when, the translator will release the newly created molecule, and leave the genome. To evolve the translator, we use all the patterns presented in table 14), which present the start and stop patterns, as well as the gene in between.

Results and Analysis

The results of the evolutionary runs, for all five ribozyme categories, are presented in figure 4. Three mutation operators were used: insertion, deletion and replacement. Figure 3(a) exhibit how these operators mutate a gene. The HGT crossover operator is introduced to evolve the most difficult String programs, the replicator and the translator (figure 3(b)). Figure 4 shows the average and best fitness curves of 50 GP runs, for all the ribozymes. We can observe that three ribozymes (mass, information transporter and transformer) have reached their best fitness within 1000 generations. A larger population of 10,000 is used to evolve the transformer and the information transporter, as they are comparatively large and sophisticated programs. The best evolved programs and their fitness values are presented in tables 5, and 11 along with their hand-written equivalents. Evolution produced replicators and the translators with > 96% accuracy, but only after utilizing a 1:4 measure of crossover relative to mutation (figure 4), and after long runs of up to 10,000 generations. Table 7 contains the optimum parameter set, used during evolution. Table 13 and 14 present the imperfect outputs of the best evolved replicator and translator, respectively.

Conclusion. This paper presents a new language, String, that is used to write and evolve all active entities, called ribozymes, within a new computational protocell model (under development). We present both hand-written and functionally-equivalent evolved String programs, from five functional categories: mass and information transporters, transformers and replicators as well as translators. For all of these categories, we evolved programs that are functionally optimal and almost equivalent to the hand-written ones. In realizing that, we utilized three traditional mutation operators: insert, delete and replace. For the replicator and the translator cases, we also developed a new biologically-inspired crossover operator (HGT). This improved the result of the evolutionary runs, giving us a replicator with 0.96 normalized fitness and a translator with a 0.97 one. In addition, the sizes of the various String programs stayed within tolerable limits too (< 100 lines of code). Next, we intend to evolve the whole set of programs needed for a stable ProtoCell simulation, observe its behavior and hence, evolve the whole cell towards different behaviours. Our current results provide evidence that such effort is worth pursuing.

References

Clark, E. B., Hickinbotham, S. J., and Stepney, S. (2017). Semantic closure demonstrated by the evolution of a universal constructor architecture in an artificial chemistry. *J R Soc Interface*, 14(130).

Das, B. and Mitra, P. (2021). High-performance whole-cell simulation exploiting modular cell biology principles. *Journal of Chemical Information and Modeling*, 61(3):1481–1492. PMID: 33683902.

Hu, T. and Banzhaf, W. (2018). *Neutrality, Robustness, and Evolvability in Genetic Programming*, pages 101–117. Springer International Publishing, Cham.

Koza, J. R. (1992). *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. MIT Press, Cambridge, MA, USA.

Koza, J. R. (1994). *Genetic Programming II: Automatic Discovery of Reusable Programs*. MIT Press, Cambridge, MA, USA.

Krawiec, K. (2016). *Behavioral Program Synthesis with Genetic Programming*, volume 618 of *Studies in Computational Intelligence*. Springer International Publishing. http://www.cs.put.poznan.pl/kkrawiec/bps.

Levenshtein, V. I. (1966). Binary Codes Capable of Correcting Deletions, Insertions and Reversals. *Soviet Physics Doklady*, 10:707.

Ofria, C., Bryson, D., and Wilke, C. (2009). *Avida: A Software Platform for Research in Computational Evolutionary Biology*, volume 10, pages 3–36.

Rasmussen, S., Knudsen, C., Feldberg, P., and Hindsholm, M. (1990). The coreworld: Emergence and evolution of cooperative structures in a computational chemistry. In *Proceedings of the Ninth Annual International Conference of the Center for Nonlinear Studies on Self-Organizing, Collective, and Cooperative Phenomena in Natural and Artificial Computing Networks on Emergent Computation*, CNLS '89, page 111–134, NLD. North-Holland Publishing Co.

Ray, T. S. (2003). *An Evolutionary Approach to Synthetic Biology: Zen in the Art of Creating Life*, pages 479–517. Springer Berlin Heidelberg, Berlin, Heidelberg.

Varetto, L. (1993). Typogenetics: an artificial genetic system. *Journal of theoretical biology*, 160(2):185—205.

Is Prediction Required? Using Evolutionary Robotics to Investigate How Systems Cope with Self-Caused Sensory Stimuli

James Garner[1] and Matthew Egbert[1,2]

[1] School of Computer Science, The University of Auckland, Auckland, New Zealand
[2] Te Ao Mārama – Centre for Fundamental Inquiry, The University of Auckland, Auckland, New Zealand
james.garner@auckland.ac.nz

Introduction

Living systems process sensory data to facilitate adaptive behaviour, but the same sensors can receive inputs both from purely external (environmental) sources, and as the result of internally driven activity. We can hear sounds in the world around us, but we can also hear our own voice when talking, and our own footsteps when walking. We can see our environment, but we also see our own bodies. Not only do we perceive both the world and the results of our own actions, but the exact same sensory stimulus can be the result of an external event, or caused by our own activity. For example the sight of a hand being waved before our eyes could be your own hand or a friend snapping you out of a daydream. The phenomenology of a self-caused stimulus can be very different from that of an externally caused one. A great example of this is the sensation of touch, which can reduce you to helpless laughter when externally applied - but trying to tickle yourself just isn't the same! (Blakemore et al., 2000)

In psychology, research on the sensory attenuation of self-caused stimuli studies how these stimuli are perceived as diminished in comparison to externally caused stimuli (Hughes and Waszak, 2011). A clear example of this effect is seen in the force-matching paradigm. Here an external force is applied to a subjects finger, after which they must use their other hand to recreate that force as precisely as possible. This takes place under two conditions. In the direct condition, the subject applies force to their finger in a manner as close to pressing on their own finger as possible. In the indirect condition, they apply the force via a mechanism, such as a lever to one side. Healthy subjects consistently apply too much force when pressing directly on their finger, indicating that the perceived force is attenuated compared to the other conditions (Pares et al., 2014).

The canonical explanation of this effect is that when the brain issues a motor command, an internal model receives a copy of that command, from which it predicts the sensory consequences of the resulting motor activity. The predicted sensory input is then subtracted from the actual sensory input, resulting in the attenuation of the stimuli (Klaffehn et al., 2019).

Methods

We developed a model of a simple embodied system with self-caused sensorimotor dynamics. Following the evolutionary robotics methodology, we explored the space of possible solutions using a genetic algorithm (GA) (Harvey et al., 2005). We aimed to learn whether solutions like the predict-and-subtract approach would evolve, and to assess the viability of non-predictive solutions for coping with self-caused sensory inputs interfering with perceiving the world.

The embodiment is a simulated, two-wheeled robot with a pair of light sensors. It moves about an infinite, flat plane which contains a light source. Its motor activity is specified by a continuous-time, recurrent neural network (CTRNN) (Beer, 1995) with six fully connected interneurons and two neurons each for sensor inputs and motor outputs. The activation of each sensor is a linear combination of environmental stimulation (determined by the sensor's distance and facing relative to the light) and interference generated from the ipsilateral motor's activity by by one of three interference functions (Figure 1).

This model is designed to allow for both the canonical, representationalist solution and alternative, non-representationalist solutions to emerge. The canonical solution can be realised because the interfering dynamics are produced by simple, smooth functions, and thus can be fully modelled by a CTRNN (Beer, 2006). Since the interference is summed with the actual sensor data, the problem can be solved by predicting the interference and subtracting it out. However, as the interfering dynamics are a function of the system's motor activity, and are coupled to the controller in a tight sensorimotor loop, this model embraces situated, embodied and dynamical explanations of cognition, and allows for the emergence of other (non-representationalist) solutions that do not involve an internal, predictive model.

We used a GA to select parameters for the CTRNN, and examined the best solution found under several conditions. Our GA is tournament based, like the microbial GA (Harvey, 2011), although it does not use crossover. We evolved a population of 50 individuals to seek the light (phototaxis) without any motor-driven interference. We then evolved five populations of solutions for each of the three interference functions, in each case starting from the population evolved without interference rather than from an initially random population. This lets us study how an existing phototactic system can be adapted to continue to perform successfully in the presence of various forms of motor-driven interference.

Results

Here we catalogue the adaptations observed in the fittest single individual evolved with each interference function, none of which rely on predicting the interference. In each case, the evolved system performs phototaxis successfully.

Avoidance: When self-caused sensory interference is only triggered by certain motor outputs, and the task at hand can be accomplished while avoiding those outputs, it may be easiest for a control system to simply modify its behaviour to do exactly that. We observed this with the sigmoidal interference. With the squared interference, we instead saw interference minimisation via reduced motor activity.

Coordination: The timing of motor-driven interference with a sensor may be regulated to coincide with environmental stimulation of that same sensor. With a one dimensional sensor like those used in this model, this leads to a sort of constructive interference, where the coincidence of motor-driven and environmental stimuli amplifies the effect of the environmental stimuli on the sensor. We observed this with the squared interference.

Time scale: The previous solutions don't work for interference which is continually varying in such a manner that its extrema are not determined purely by the motor activity (e.g. Figure 1C). However if such interference is of a high enough frequency relative to the frequency of environmental stimuli, then this difference in time scale can be leveraged to separate the interference from the environmental stimuli. Slowly varying stimuli can be perceived through quickly varying interference, which we observed in the case of the sinusoidal interference. We also found that this system evolved elevated motor activity, which raises the frequency of the interference and amplifies the time scale difference.

Shaping environmental stimuli: The solution evolved with no interference made use of sharp spikes of environmental stimuli. We found that spikes like these could be completely lost in the high frequency sinusoidal interference. In addition to raising the frequency of the interference, the behaviour of the solution evolved with sinusoidal interference tended to lower the frequency of environmental stimulation compared to the no interference solution.

Incorporating interference functionally: Removing motor-driven interference from a system optimised to perform a task in the presence of that interference does not necessarily improve performance, and may instead degrade it significantly. Furthermore, we found that motor-driven sensor stimulation played a functional role in the successful phototactic behaviour of some evolved systems.

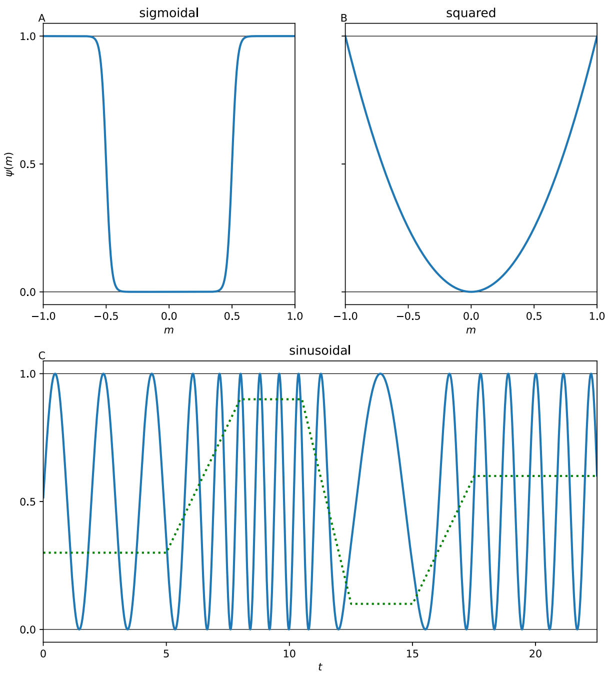


Figure 1: Three functions which depend on motor activity are used to generate sensory interference. Interference determined by the right motor is added to the right sensor's input stream, and likewise for the left sensor and motor. Figures A and B plot pure functions of motor activity, while Figure C plots a function of time whose frequency is determined by the motor activity. The solid blue line shows the interference, while the dotted green line shows the motor activity.

Conclusions

This all suggests that prediction and subtraction do not tell the whole story when it comes to coping with self-caused sensory stimuli. In some ways this is obvious, as self-caused sensory stimuli are involved in a range of activities in which they do not play an interfering role. For example, the sensation of self-touch when kneading an aching muscle, or occlusion of the visual field when engaging in visually guided reaching and grasping. In these activities, self-caused sensory stimuli are actually desirable. However, our model shows that even in a situation where clear perception of the environment seems obviously beneficial, self-caused sensory stimuli may not play an entirely interfering role. Furthermore, we can see that even when responsiveness to the environment is needed, prediction and subtraction are not the only games in town.

Acknowledgements

This extended abstract summarizes research under review for publication in Frontiers in Neurorobotics - preprint available (Garner and Egbert, 2022).

References

Beer, R. D. (1995). On the Dynamics of Small Continuous-Time Recurrent Neural Networks. *Adaptive Behavior*, 3(4):469–509.

Beer, R. D. (2006). Parameter space structure of continuous-time recurrent neural networks. *Neural computation*, 18(12):3009–3051.

Blakemore, S.-J., Wolpert, D., and Frith, C. (2000). Why can't you tickle yourself? *Neuroreport*, 11(11):R11–R16.

Garner, J. and Egbert, M. (2022). Embodiment enables non-predictive ways of coping with self-caused sensory stimuli. preprint, `https://arxiv.org/abs/2205.06446`.

Harvey, I. (2011). The Microbial Genetic Algorithm. In Kampis, G., Karsai, I., and Szathmry, E., editors, *Advances in Artificial Life. Darwin Meets von Neumann*, pages 126–133, Berlin, Heidelberg. Springer Berlin Heidelberg.

Harvey, I., Paolo, E. D., Wood, R., Quinn, M., and Tuci, E. (2005). Evolutionary Robotics: A New Scientific Tool for Studying Cognition. *Artificial Life*, 11(1-2):79–98.

Hughes, G. and Waszak, F. (2011). ERP correlates of action effect prediction and visual sensory attenuation in voluntary action. *NeuroImage*, 56(3):1632–1640.

Klaffehn, A. L., Baess, P., Kunde, W., and Pfister, R. (2019). Sensory attenuation prevails when controlling for temporal predictability of self- and externally generated tones. *Neuropsychologia*, 132:107145.

Pares, I., Brown, H., Nuruki, A., Adams, R. A., Davare, M., Bhatia, K. P., Friston, K., and Edwards, M. J. (2014). Loss of sensory attenuation in patients with functional (psychogenic) movement disorders. *Brain*, 137(11):2916–2921.

Augmenting Evolution with Bio-Inspired “Super Explorers”

Vincent Ragusa[1,3,4] and Cliff Bohm[2,3,4]

[1]Department of Computer Science and Engineering
[2]Department of Integrative Biology
[3]BEACON Center for the Study of Evolution in Action
[4]Michigan State University, East Lansing, MI 48824
ragusavi@msu.edu

Abstract

Natural evolving populations experience constantly fluctuating selection strength, which also creates a fluctuating trade-off between exploration and exploitation. Range expansion, for example, creates semi-persistent spatially-distributed differences in selection strength, particularly among the pioneering agents along the leading edge of each range expansion. The pioneers experience reduced selection strength and in turn experience greater potential for exploration, while selection on the remainder of the population ensures that prior discoveries are not lost.

Here we describe a method to augment pre-existing selection algorithms inspired by the exploration-boosting properties of range expansion events. The key insight is that for productive exploration on deceptive landscapes, mutations must be able to accumulate and persist in some, but not all, lineages. We create artificially drifting lineages of “super explorers” and show that they can be used to improve the performance of another selection algorithm.

Introduction

All search algorithms, be they natural selection or computer models of evolution, are subject to the fundamental limitations of the no-free-lunch theorems (Ho and Pepyne, 2002), and particularly to the explore-exploit tradeoff (Millidge et al., 2021). Managing this tradeoff is typically a main concern in the development of computational selection algorithms. In this work, we introduce the “super-explorer method” which can be sued to augment pre-existing selection algorithms. The super-explorer method allows us to tune the trade-off between exploration and exploitation to better align with the ruggedness of a fitness landscape.

The super-explorer method augments other selection algorithms by adding agents (“super explorers”), that are allowed to drift (i.e, freely accumulate mutations). In essence, our method allows a pre-existing selection algorithm to focus on exploitation, while super explorers enhance exploration. We compare three implementations of the super-explorer method on two different kinds of fitness landscapes: an NK-fitness landscape (Kauffman et al., 1993), and a saw-tooth fitness landscape (Ragusa and Bohm, 2021), and across a wide array of configurations. We show that for both types of fitness function, there are configurations where evolution augmented with super explorers preforms better than without.

Biological inspiration

Natural selection does not act with consistent strength; Shifting balance theory (Wright, 1932, 1982), range expansion (Slatkin and Excoffier, 2012; Peischl et al., 2013; Peischl and Excoffier, 2015; Peischl et al., 2015; Gilbert et al., 2017; Burton and Travis, 2008), environmental noise (Wang and Zhang, 2011; Van Egeren et al., 2018; Ragusa and Bohm, 2021), population size changes (Jain et al., 2011; Ochs and Desai, 2015; Rozen et al., 2008), sexual selection (Bohm et al., 2019), and mass-extinction events (Mathias and Ragusa, 2016; Engholdt and Mathias, 2021) all describe scenarios where selection strength changes over time or space.

As a consequence of naturally occurring fluctuations in selection strength, a population’s ability to explore their fitness landscape also fluctuates. Furthermore, during the periods of increased exploration, the population may discover new fitness peaks that, under stronger selection, would have been unlikely or impossible to discover. Range expansion events are one particular scenario that can cause a reduction in selection strength. As a species enters new territory, a lack of competition can result in an accumulation of deleterious mutations in lineages on the leading edge of the expansion for as long as uncolonized territory remains (Burton and Travis, 2008). The continued accumulation of deleterious mutations in one lineage can result in adaptation via valley-crossing, a process known to be critical for evolutionary adaptation (Covert et al., 2013; Oliveto et al., 2018).

In this work, we present a method designed to augment existing selection algorithms with the exploration-boosting power of range expansion, without requiring that populations have spatial structure. In addition, unlike range expansion, the exploratory advantages are maintained indefinitely. The key insight underlying our method is that for sufficient exploration to occur, mutations must be able to accumulate in some—but not all—lineages before purifying selection acts on them. To achieve this, we introduce super explorers, agents that always have exactly one offspring every generation. As super explorers propagate regardless of fitness, they experience the same mutation-accumulation as organisms at the leading edge of a range expansion.

Similar selection algorithms

The hybrid selection algorithms resulting from augmentation with the super-explorer method share similarities with previously defined selection algorithms.

Particle Swarm Optimization (PSO) (Poli et al., 2007) models agents as particles and simulates forces that attract the particles to the highest observed fitness (analogous to selection). However, the individual particles in PSO do not move directly to the higher gradients, but instead chaotically trend towards it, often exploring areas of the fitness landscape more distant from known optima.

Lexicase selection (Helmuth et al., 2014; La Cava et al., 2016) and real-valued tournament selection (Ragusa and Bohm, 2021) are examples of selection algorithms that probabilistically ignore some or all fitness gradient information. Novelty search (Lehman and Stanley, 2011) foregoes following the fitness gradient altogether and instead focuses on collecting a catalog of functionally distinct solutions, evaluating them for relevance to the fitness function *ex post facto*.

Island models (Whitley, 2001) protect innovations by removing the competitive interactions of agents between separate mutually-exclusive subpopulations, called islands. When all agents are considered together, the overall effect is a reduction in the selection strength. Restricting competition to be within individual islands also creates small-population drift effects that can facilitate valley-crossing.

Systems augmented with super explores are different from the examples above in key ways. While super explorers do experience a form of occasional selection during the decay-replace process (described in Methods), they do not individually experience a fitness gradient, even in the form of selection for novelty. While some other selection algorithms, such as lexicase and real-valued Tournament, can occasionally allow for several generations of drift on some lineages, super explorers are maintained separately from the main population, ensuring their lineages experience sustained periods free of selection. While island models do break up a population into separate pools, they generally restrict selection and reproduction to be within each island. The super-explorer method makes every agent visible to the selection algorithm in use, allowing for the immediate adoption of a beneficial mutation. Furthermore, although island models facilitate drift, it does not result in sustained drift on lineages for long periods as in the super-explorer method.

Methods

Augmenting evolution with super explorers

Super explorers can augment any well-mixed discrete-generation agent-based selection algorithm. In our implementation, super explorers are added to a selection algorithm by dividing a population into two mutually exclusive pools, the *active selection pool* (or *ASP*) and *super-explorer pool* (or *SEP*) (see Fig. 1). The ASP evolves via the rules of an externally defined selection algorithm (such as, roulette, tournament, lexicase, etc.), except that parents can be drawn not only from the ASP, but also from the SEP. On the other hand, the agents in the SEP each produce one offspring, with the standard mutation load, regardless of fitness. While lineages in the ASP survive and perish following the rules of the selection algorithm in use, lineages in the SEP end only by decay. Whenever an agent in the SEP is about to reproduce, there is a chance (the *decay rate*) that the agent will be removed and replaced by some other agent in the population. The number of agents in the ASP and SEP, the decay rate, the process used to replace decayed SEP agents, the mutation settings, and the selection algorithm used in the ASP (and related settings) are all parameters established by the user.

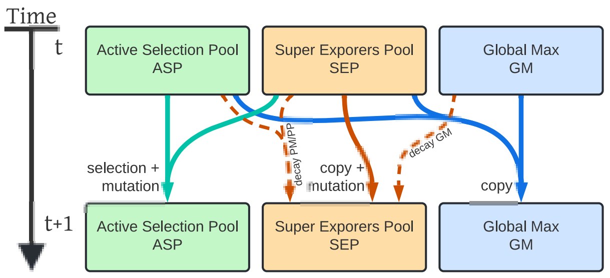


Figure 1: A diagram of an evolving system augmented with super explorers. Here, the population is divided into *the active selection pool* (ASP) and the *super-explorer pool* (SEP). While selection in the ASP (green arrows) is determined by a pre-existing selection algorithm (such as roulette, tournament, lexicase, etc.) with parents drawn from the ASP and SEP, agents in the SEP are free from selection and simply accumulate mutations (red solid arrow). From time to time, agents in the SEP decay and are replaced. Replacements (red dashed arrows) are drawn from either the ASP and SEP, or the global max, depending on the replacement process, where the global max maintains a copy of the highest scoring agent seen to date (blue arrows).

Because the super explorers reproduce regardless of their fitness, they are able to accumulate mutations without consequence. The decoupling of survival from fitness allows super explorers to explore the fitness landscape in directions that selection might prohibit; they may descend into, and possibly cross, fitness valleys. The mutational paths of super explorers are, in fact, random walks (i.e., undirected and unfocused), so they are an inefficient selection algorithm on their own. However, by introducing super explorer decay and replacement, we provide a degree of focus to help keep the search process "on track." The search pattern of the super explorers can be seen as a genotypic radiation where the center of the radiation is determined by the replacement process, the radiation distance is determined by the decay rate, and the density of the search is determined by the number of agents in the SEP.

Since the selection algorithm used to choose parents in the ASP has access to both the ASP and the SEP, super explorers can migrate into the ASP if their fitness is competitive.Thus, a selection algorithm augmented with super explorers has the advantage of simultaneously exploring freely (in the SEP) while exploiting discoveries (in the ASP), compared to a non-augmented selection algorithm that must balance the explore-exploit trade-off with only an ASP.

Tunable parameters of super explorer systems

When employing the super-explorer method, the user must choose a selection algorithm to augment, as well as a mutation scheme. These choices will introduce a number of parameters. In addition, the supper-explores methods introduces three more parameters: SEP size, decay rate, and replacement process, which we describe below.

Parameter: super-explorer pool size SEP size sets the number of agents in the SEP and can be set to any integer value, from 0 to population size. SEP size controls the intensity of the search conducted by the super explorers. Note that ASP size is not a parameter and is simply defined as population size minus SEP size.

Parameter: decay rate Decay rate is the per-generation probability that each agent in the SEP is replaced, and directly controls the explore-exploit trade-off of the SEP. The decay rate can be set to any value r, such that $0 \leq r \leq 1$. The decay rate determines the average amount of time, $\bar{t} \sim 1/r$, that a super-explorer lineage persists before replacement, which in turn determines how far super explorers can drift (i.e., explore). While lower decay rates can result in more exploration, they can also be wasteful if the valleys in the fitness landscape do not require such distant explorations.

Parameter: replacement process The replacement process defines the process used to replace a decayed agent in the SEP. In this work, we consider three replacement processes, shown in Fig. 1, though many others are possible.

The first process, the *PopSelect* (PS) process, replaces decayed SEP agents with a new agent generated using the same selection algorithm used for reproduction in the ASP.

The second process, the *PopMax* (PM) process, replaces decayed SEP agents with the max of the population (i.e., the max of all the agents in the combined ASP *and* SEP).

The third process is the *GlobalMax* (GM) process. In addition to the ASP and SEP, this replacement requires that the global max, a copy of the highest fitness agent seen to date in either pool, is maintained. The global max is used to replace decayed SEP agents.

Fitness functions

We use two types of fitness functions to collect performance data for the parameter sweeps. First, we chose a typical NK-landscape to provide an intuition regarding the performance of super explores in a familiar context. Then, we turn to a saw-tooth function to provide a different perspective on the operation of the super-explorer method under more controlled conditions.

Fitness function: NK-landscape The first fitness function is the well studied NK-landscape (Kauffman et al., 1993). This landscape has a rugged topology and a global optimum. The NK function, $\text{NK\_eval}(i, g_i|N, K)$, decides the fitness contribution of the i-th gene, g_i. The fitness landscape is randomly generated with parameters $N = 20$ and $K = 5$ which control genome size, $G = \{g_1, ..., g_N\}$, and strength of epistasis respectively. The score assigned to each organism by NK is given by

$$\text{score}(G) = \frac{1}{N} \sum_{i=1}^{N} \text{NK\_eval}(i, g_i|N, K) \qquad (1)$$

Addressing diminishing returns The NK-landscape exhibits diminishing returns (Orr, 2005), which means that the ratio of beneficial to deleterious mutations and the fitness gain per beneficial mutation decreases as fitness increases. As a result, the optimal explore-exploit ratio changes over the course of evolution. Strong exploitation initially maximizes the rate at which fitness increases, but later is likely to result in getting stuck on local optima. Conversely, stronger exploration, though slower at first, can achieve higher final performance in the long run, because it is able to escape local optima.

In order to disambiguate the effects of diminishing returns from other dynamics, we included the saw-tooth landscapes, which do not exhibit diminishing returns and allow us to study the behavior of a system with a constant difficulty.

Fitness function: saw-tooth landscape The second fitness function, the saw-tooth fitness function, (Ragusa and Bohm, 2021) defines a saw-tooth mapping function from genomic values to scores along an infinite set of ever higher fitness peaks (like the teeth of a saw) as shown in Fig. 2.

The saw-tooth landscape has two special properties. First, it has no global optimum. Second, every fitness valley is self-similar; regardless of absolute position on the fitness landscape, the difficulty of descending into and the benefit of crossing every valley is the same. These two properties together result in a landscape that presents a constant challenge to an evolving population, without diminishing returns. In this landscape, agents have 10 independent genes, $G = \{g_1, ..., g_{10}\}$, where each gene is a single integer value. Applying the mapping function to each gene results in 10 gene scores that are summed to produce the agent's overall score. The saw-tooth function $\text{saw}(x|w, p, b)$ is specified by a valley width (w), a fitness penalty per mutation into each valley (p), and a fitness benefit per valley crossed (b):

$$\text{score}(G) = \sum_{i=1}^{10} \text{saw}(g_i|w, p, b) \tag{2}$$

Here we test four versions of saw-tooth functions. In all four, $p = -0.05$ and the $b = 1.0$. We varied w to create four versions of the function with different difficulties, from $w = 4$ (easy) to $w = 7$ (hard). These functions are shown in Fig. 2.

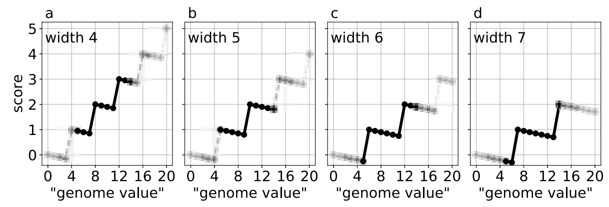


Figure 2: The saw-tooth functions that map gene values to "scores", used in Fig. 4. Panels [a] through [d] show the valley widths (w) 4 through 7, respectively. The values for penalty ($p = -0.05$) and benefit ($b = 1.0$) are the same in all 4 functions.

Roulette selection

We use roulette selection in the ASP to determine which parents in the current population will produce the next generation. The fitness of every agent in the current population (i.e., all agents in the ASP and SEP) is calculated, and then for each new ASP agent, each parent has a chance to reproduce proportional to their share of the total population fitness. Roulette selection is known to suffer from a diminishing selection pressure as the absolute value of agents' scores increases during evolution. In order to ensure that equivalent relative increases to score result in the same relative increase in offspring production, the scores generated from both the NK-landscape and the saw-tooth landscape are exponentiated before fitness shares are determined (i.e., when agent fitness is calculated).

$$f(G) = \exp(\text{score}(G)) \tag{3}$$

Experiment conditions

In order to determine the effectiveness of super explorers, we compare roulette selection with and without super-explorer augmentation on two classes of fitness function (NK and saw-tooth). We use a total population (ASP + SEP) size of 1024 in all experiments. For each combination of selection regime and function, we run a three-dimensional sweep of SEP size, decay rate, and replacement process. We run decay rates {0.001, 0.002, 0.005, 0.01, 0.02, 0.05, 0.1, 0.2, 0.5, 1.0}, pool sizes {0, 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024}, and replacement processes {GM, PM, PS}.

We run 101 replicates for each condition using the NK-fitness landscape ($N = 20$ and $K = 5$) for 10,000 generations and record the final maximum fitness from each replicate. Each replicate evolves on the *same* NK landscape, which is randomly generated *a priori*. A bit genome was used where mutations are introduced with a bit-flip mutation operator, with a 0.0005 per-site mutation rate (i.e., a 0.01 per-agent mutation rate).

We run one replicate for each condition using the saw-tooth landscape for 40,000 generations and record the number of valleys crossed (running one replicate on the saw-tooth function and counting the number of valleys crossed efficiently generates the same results as running many replicates of shorter duration (Ragusa and Bohm, 2021)). In addition to the parameter sweep described above, we used four saw-tooth functions with $w = \{4, 5, 6, 7\}$ (shown in Fig. 2). An integer genome was used where mutations are introduced with a point-offset mutation operator that modifies a genome value by ±1, with a 0.05 per-site mutation rate (i.e., a 0.5 per-agent mutation rate, as each agent has 10 loci).

The MABE evolution framework was used to run experiments (Bohm et al., 2017). Code and instructions to allow for replication can be found at `https://github.com/cliff-bohm/SuperExp_ALIFE_2022`.

Results

NK landscape

Figure 3 displays the data collected from 101 replicates run on the NK-landscape with parameters $N = 20$ and $K = 5$. The figure presents three grayscale maps (labeled [1], [2], and [3]) each showing results generated using a different replacement process (GM, PM, and PS). The cells in each map show the averages, across replicates, of the maximum fitness detected at the end of each replicate. Note that values in each plot associated with SEP size = 0 are the same, since decay rate only matters if there are super explorers. The fitness differences between the max-fitness, min-fitness, and control configurations of each panel are computed and checked for significance with a two sample z-test (shown in Table 1).

The NK-landscape results show that the GM and PM processes provide similar results, both of which are quite different from the PS data. In the GM and PM results, we see that the addition of any super explorers improves performance under almost all conditions. In addition, as SEP size increases, low decay rates tend to improve adaptation while high decay rates tend to have the opposite effect. In fact, a population made up entirely of super explorers (SEP

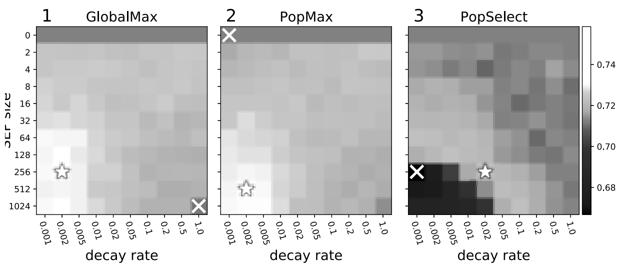


Figure 3: Average maximum final fitness on an NK landscape from 101 replicates using super explorers with three replacement processes: [1] GM, [2] PM, and [3] PS. Lighter shades correspond with higher final fitness, and darker shades correspond with lower final fitness. The color range is set so that gray is associated with the control condition where SEP size = 0 (i.e., no super explorers). The stars and crosses indicate the highest-fitness and lowest-fitness configuration of each panel, respectively. The largest increase and decrease of fitness, relative to the control, are shown in Table 1.

size = 1024) combined with decay rate = 1.0, shows little to no signs of improvement over a non-augmented population.

We note a band of relatively high performance in the PS data, highlighting the conditions with the greatest final scores. This band appears to show a trade-off between SEP size and decay rate. To the right and above the high-performance band, we see values that conform closely to the non-augmented system, while to the left and below the band, we see that final recorded scores are lower than the non-augmented system. The area of low values correlates with a large SEP size and low decay rate.

Note that the values in the column associated with decay rate = 1.0 in the PS data all match the values associated with SEP size = 0. In these conditions, SEP agent replacement happens every generation and uses the same replacement process used in the ASP. As a result, the two pools act as a single ASP (small fluctuations are the result of sampling noise).

Decay Method	Largest Increase	Largest Decrease
GM	$+2.76\%$ *	-0.15%
PM	$+2.61\%$ *	$\pm 0.00\%$
PS	$+1.51\%$ *	-6.44% *

Table 1: The fitness differences between the max-fitness, min-fitness, and control configurations of the NK-landscape data in Figure 3. '*' indicates a p-value of $p < 1. \times 10^{-6}$ (two sample z-test).

Saw-tooth landscape

Figure 4 shows grayscale maps illustrating the number of valleys crossed at the end of the 40,000 generations, with agents evolved on saw-tooth fitness functions. Each of the letters [a] though [d] indicate a different valley width, from 4 to 7. The numbers [1] through [3] indicate the replacement process used (GM, PM, and PS). Note that values in each panel associated with SEP size = 0 are the same within each row, since decay rate only matters if there are super explorers.

As we saw in the NK data, the saw-tooth landscape results show that the GM and PM processes provide similar results that are quite different from the PS data.

Across all panels of the saw-tooth data, there are vertical trends, associated with SEP size, and horizontal trends, associated with decay rate. As we move from top to bottom and introduce a higher ratio of super explorers (larger SEP size) relative to the size of the ASP, we generally see a smooth transition from SEP size = 0 to SEP size = 1024. Conversely, as we move from low decay rates to higher decay rates, particularly larger SEP sizes, we see that the change in performance is not as simple to describe. In all GM and PM results, while we see that large SEP combined with a high decay rate reduces fitness gain relative to the non-augmented system, the effect of large SEP and low decay rate is not

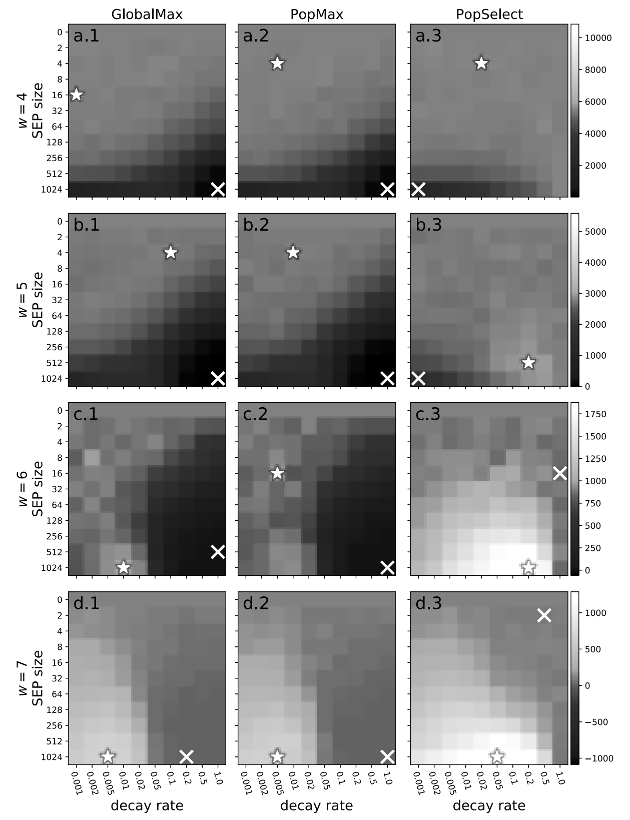


Figure 4: Total number of valleys crossed after 40k generations on saw-tooth landscapes. Horizontal panel groups [a], [b], [c], and [d] show data for $w = \{4, 5, 6, 7\}$ saw-tooth functions, respectively. Vertical panel groups [1], [2], and [3] show data for replacement processes GM, PM, and PS, respectively. The stars and crosses indicate the highest-fitness and lowest-fitness configuration of each panel, respectively. Note: the scales on each color bar (i.e., for each row) are different.

constant. Rather, in the data for large SEP and low decay rate, the final scores flip from worse than non-augmented to better as we move from less deceptive to more deceptive landscapes.

In the PS results Fig. 4[a,b,c,d][3], we see a different trend. As in the NK data, when decay rate = 1.0 agents in the SEP experience the same selection as agents in the ASP (i.e., replaced every generation using the selection algorithm from the ASP). Therefore, as in the NK data, the two pools act as a single ASP (small fluctuations are the result of sampling noise). Unlike the NK data, we see that larger SEP sizes perform as well as or better than the non-augmented system, except in the lower left of [a, b].[3] where SEP is very large, and decay rate is very low. The highest rates of valley crossing appear to correlate with large SEP sizes and middling decay rates (i.e., around 0.05 to 0.1).

Discussion

In this work, we show that augmenting a selection algorithm with super explorers can result in improved adaptation to a range of landscapes and parameter settings. In this discussion, we are primarily interested in describing the super-explorer augmentation method.

The conditions where SEP size = 0 show the behavior of the non-augmented system. In these conditions, there are no super explorers, so decay rate has no effect. Hereafter we will refer to these conditions as the 'control' and they will serve as the baseline for all of our comparisons.

In our experiments, we test conditions with different SEP sizes, but we do not alter the number of agents in the total population or the mutation inflow per agent. Therefore, the populations always experiences the same number of agent evaluations and have the same potential for discovery in the form of mutations. As a result, differences in performance between the control and other conditions must have some other explanation. As we will see, the observed differences can be explained in terms of a trade-off between exploration and exploitation.

The explore-exploit trade-off of super explorers

In evolutionary systems (natural or computational) the explore-exploit trade-off explains how the conditions that allow for effective navigation on smooth fitness landscapes hinder navigation on deceptive fitness landscapes and *vice versa*. On a smooth landscape the paths towards higher fitness are easy to discover, so unfocused distant explorations are wasteful: it is prudent to spend energy exploiting the local fitness gradient. Conversely, on deceptive landscapes the paths to higher fitness often require crossing fitness valleys, so focusing on local search only results in short-term superficial gains: it is prudent to spend time on long shots.

Our initial inspiration for the super-explorer method was the biological phenomenon of range expansion. There, we observed that the freedom from selection along the leading edge of a range expansion creates lineages in which mutations could accumulate, while the remainder of the population experiences purifying selection. To mimic this effect, we designed a method that divides a population into two pools: the SEP that specializes in exploration, and the ASP that specializes in exploitation.

Decay rate Generally speaking, the amount of genetic change that any lineage can accumulate is related to the strength of selection, and how long the lineage can avoid extinction. SEP lineages (unbroken phylogenies in the SEP) are special in that they are free from selection, and so their potential for genetic change is only limited by how long they persist, which in turn depends on the decay rate. Meanwhile, the decay rate *also* establishes the only selective force at work in the SEP: how often replacements occur. Thus, the conditions likely to enhance exploration, high drift potential and low selection strength, are conditions generated by low decay rates while the conditions likely to enhance exploitation, low drift potential and strong selection, are conditions generated by high decay rates.

Replacement process The three replacement processes we chose to test correlate with a range of selection strengths. The GM process provides the strongest selection strength because it always maintains the best solution. The GM process insures that small fitness changes (potentially undetectable by the ASP's selection algorithm) are always exploited, but this comes at a cost: the inability to forget can inhibit escape from local optima. Moreover, when the GM process is combined with an SEP size equal to population size and decay rate equal to 1.0, the resulting system is synonymous with an elitism selection algorithm; every agent tests one mutation, and then either becomes the new global max or is forgotten.

Compared to the GM process, the PS process results in weaker selection, which cannot exceed the selection strength of the ASP's selection algorithm. This is because at decay rate = 1.0, all agents in the population are replaced using the ASP selection algorithm every generation (regardless of SEP size).

The PM process represents a middle ground in terms of selection strength. However, since we are using a relatively large population size relative to decay rate, forgetting high quality solutions is unlikely, so the GM and PM processes generate similar results.

SEP size The SEP size parameter determines the degree to which each pool (ASP and SEP) drives the system. As SEP size increases, particularly when decay rates are low, there are more low quality solutions in the total population, which increases the chance that low quality solutions will be selected during ASP reproduction. As a result, larger SEP sizes result in weaker selection in the ASP. This demonstrates another way that low decay rates can enhance exploration.

Analyzing the NK data

GM and PM Processes Fig. 3[1,2] show the results of using the GM and PM processes to evolve populations on the NK-landscape. We see that enhancing exploration with low decay rate improves performance, while maximizing exploitation, with either low SEP size or high decay rate, results in performance that is not significantly different from the control. The fact that the results of elitism are similar to the control suggest that the ASP is already generating elitist-like behavior that hinders exploration. It follows then that improved adaptation associated with lowered decay rates results from increased exploration.

The sudden jump in fitness from SEP = 0 to SEP = 2 in Fig. 3[1,2] occurs because the GM and PM processes select max *perfectly* when making replacements into the SEP. Using the GM process, it is *impossible* to forget a high-value solution once it has been discovered; forgetting is only very unlikely when using the PM process. As a result, the GM and PM process are able to identify and exploit fitness improvements that the ASP selection algorithm alone may not detect.

Finally, the decrease in fitness from top to bottom of the right-most column (decay rate = 1.0) shows that the behavior of the ASP alone is not exactly the same as elitism. While the ASP can allow for the accumulation of small (i.e., nearly-neutral) deleterious mutations, a system experiencing elitism can never move in a direction that results in *any* loss of fitness. As a result, the highest fitness in this column occurs at SEP size = 2, indicating that the best performance is generated by a large ASP (that does have the ability to explore, if only locally), augmented by a SEP that does not forget (but cannot explore).

PS Process Fig. 3[3] shows the results of the PS process used to evolve populations on the NK-landscape. The low fitness recorded in the low left is the result of weak selection that is unlikely to find effective solutions, and it is likely to forget what it does find. The most interesting feature in this panel is the relatively bright band—indicating the highest scores—that appears to show a trade-off between SEP size and decay rate. SEP size = 1024 and decay rate = 0.1 marks the bottom of this band, and suggests that lineages that survive for about 10 generations are optimal when the entire population is in the SEP. As we decrease the size of the SEP, we see that lower decay rates, (i.e., drifting lineages that persist for longer) are optimal. This is because the larger ASP results in more exploitation and less exploration. Apparently, the trade-off exists because when there are fewer SEP agents they need more time to drift in order to make discoveries, but larger ASPs are better at exploiting beneficial fitness gradients. The entire right side of the panel evolves similarly to the control because along this edge agents are replaced every generation using the same algorithm used in the ASP.

Analyzing the saw-tooth data

GM and PM Processes Fig. 4[a].[1,2] show the results of the GM and PM processes on a saw-tooth function with only limited deception. Here we see that small SEP sizes generate the best performance, while larger SEP size degrades performance. This indicates that the ASP selection settings are well tuned for this function, and evolution does not improve with the addition of super explorers. Moreover, elitism results in the lowest levels of performance, since valley crossing is required to optimize this function. In Fig. 4[b,c,d].[1,2] we find the results from the other three saw-tooth landscapes, each with an increasing level of deception. As the functions become harder to adapt to, we see that the control does not produce the best performance because it does not allow enough exploration. As a consequence, larger SEP sizes become more effective. In Fig. 4[d] (the hardest function), the best performance occurs when the entire population is SEP agents and SEP lineages persist for about 200 generations (decay rate = 0.005). This suggests that navigation on this landscape requires a high degree of exploration. The fact that the highest performance values are not at the lowest decay rates means that too much exploration, beyond what is necessary, is wasteful.

PS Process The results generated by the saw-tooth landscape using the PS process are displayed in Fig. 4[a,b,c,d][3]. As in the NK data, and for the same reason, the entire right side of the panel evolves similarly to the control. However, the main results differ significantly from the NK-landscape PS process results (in part due to the absence of diminishing returns — see Methods). Except in the case of the least deceptive function ([a]), the best performance across the three replacement processes is found in the PS data. This supports the idea that weakened selection is beneficial for valley crossing on the harder saw-tooth functions.

Super explorers alone

The bottom row of each panel in Figures 3 and 4 corresponds to conditions where the entire population is in the SEP; there are no agents in the ASP. The high performance observed in some SEP-only configurations demonstrates that the combination of drift and replacement is a viable search process in its own right. In fact, for several of the fitness functions, a configuration with SEP = population size achieves maximum or nearly maximum performance. We call this new drift-and-replace search process "Drift Pool Optimization" or DPO.

There is a surprising similarity between DPO and particle swarm optimization (PSO). Both DPO and PSO operate as a swarm of particles, as opposed to implementing evolution by natural selection. The main difference is that while the particles in PSO tend to converge towards known optima, the particles in DPO originate near current optima and are allowed to drift. In both cases, the degree to which solutions can diverge from know optima is critical to success: too little divergence will stifle innovation, while too much diffusion will result in chaotic behavior incapable of effectively making discoveries.

Super explorers and recombination

In addition to the experiments shown in this work, we tested the saw-tooth functions with recombination (results not shown). When producing offspring in the ASP and when picking replacements using the PS process, we select two parents and perform three-point recombination to generate offspring (while GM and PM processes replacements are still asexual). We found that while scores with recombination tended to be higher, the trends in the data were almost identical.

Extending the super-explorer method

In this work, we presented the super-explorer method, an augmentation that can be used to enhance pre-existing selection algorithms. Here we present some potential modifications and extensions.

While a super-explorer-augmented system is already able to simultaneously support enhanced exploration (in the SEP) and exploitation (in the ASP), the method could be enhanced so that it automatically modifies both decay rate and SEP size based on observed rates of fitness gain in an adaptive manner, for example by monitoring the time the population spends in stasis.

Another possible modification to the super explorer-method would be to encourage diversity in the SEP, which would ensure that the SEP explores genetic space more uniformly. Both genetic diversity and phenotypic diversity could be investigated. Furthermore, an archive of high fitness agents, like the kind used in MAP-Elites(Mouret and Clune, 2015), could replace the global max pool and offer a more diverse alternative to elitism.

An interesting alternative to super-explorer augmentation would be to simply use another selection algorithm in place of the SEP. This would allow two selection algorithms with different behaviors to synergize. We are particularly interested in investigating augmenting Particle Swarm Optimization (PSO) with super explorers, and also in using PSO in place of the SEP to augment other selection algorithms.

There is no reason that a system must be limited to only two selection algorithms. A population could be subdivided into any number of pools, each acting as an ASP or SEP with unique selection algorithms and parameterization. Automation in the form of pool-size balancing and parameter adjustments could also be considered.

Finally, super explores could be used to augment selection algorithms that do not have discrete generations, but instead implement overlapping or other less standard evolutionary modules. In these cases, the method would need to be enhanced to determine when reproduction and replacement in the SEP should occur. The method could otherwise be unaltered.

Using the super-explorer method to study evolution

The super-explorer method could be used to further our understanding of general evolutionary processes related to the trade-off between exploration and exploitation. Since the behavior of the SEP can be related directly to the explore-exploit trade-off, super explorers could be used to gauge the relative explore-exploit trade-off of various selection algorithms. For example, in Fig. 4[c,d][1,2], SEP size = 1024 with decay rate = 0.05 performs about as well as the control. We theorize that the explore-exploit trade-off in these conditions is similar, but further investigations would be required to support this conjecture. If correct, we believe this hypothesis could be extended to state that any conditions that share performance values likely also experience similar explore-exploit trade-offs.

In this work, we only considered a single selection algorithm for the ASP: roulette selection. Other configurations of roulette-wheel selection, representing different selection strengths, as well as other selection algorithms should be tested. This approach could allow us to ask a number of different questions, such as: how does tournament size affect the explore-exploit trade-off?

Conclusion

In this work, we introduced a new bio-inspired optimization technique called “the super-explorer method” and we demonstrated the efficacy of the method on a number of deceptive fitness landscapes. There is a wealth of literature exploring how population size, selection strength, and mutation rate affect rates of adaptation. In addition to these, we argue that the frequency at which selection is applied and the distance a lineage can drift before it is evaluated also affect the success of a selection algorithm. Other selection algorithms have experimented with changing the frequency with which selection is applied (e.g., lexicase and real-valued tournament) or ignoring fitness altogether (e.g., novelty search), but we believe we are the first to design a method that intentionally protects lineages in order to promote discovery by drift.

Acknowledgments

Computational resources were provided by the Institute for Cyber-Enabled Research (iCER) at Michigan State University. We thank Chris Adami, for insights that helped develop this work.

References

Bohm, C., Ackles, A. L., Ofria, C., and Hintze, A. (2019). On sexual selection in the presence of multiple costly displays. In *ALIFE 2019: The 2019 Conference on Artificial Life*, pages 247–254. MIT Press.

Bohm, C., C G, N., and Hintze, A. (2017). MABE (modular agent based evolver): A framework for digital evolution research. *Proceedings of the European Conference of Artificial Life*, pages 76–83.

Burton, O. J. and Travis, J. M. (2008). The frequency of fitness peak shifts is increased at expanding range margins due to mutation surfing. *Genetics*, 179(2):941–950.

Covert, A. W., Lenski, R. E., Wilke, C. O., and Ofria, C. (2013). Experiments on the role of deleterious mutations as stepping stones in adaptive evolution. *Proceedings of the National Academy of Sciences*, 110(34):E3171–E3178.

Engholdt, K. and Mathias, H. D. (2021). A biologically-inspired model for mass extinction in genetic algorithms. In *2021 IEEE Congress on Evolutionary Computation (CEC)*, pages 1078–1085. IEEE.

Gilbert, K. J., Sharp, N. P., Angert, A. L., Conte, G. L., Draghi, J. A., Guillaume, F., Hargreaves, A. L., Matthey-Doret, R., and Whitlock, M. C. (2017). Local adaptation interacts with expansion load during range expansion: maladaptation reduces expansion load. *The American Naturalist*, 189(4):368–380.

Helmuth, T., Spector, L., and Matheson, J. (2014). Solving uncompromising problems with lexicase selection. *IEEE Transactions on Evolutionary Computation*, 19(5):630–643.

Ho, Y.-C. and Pepyne, D. L. (2002). Simple explanation of the no-free-lunch theorem and its implications. *Journal of optimization theory and applications*, 115(3):549–570.

Jain, K., Krug, J., and Park, S.-C. (2011). Evolutionary advantage of small populations on complex fitness landscapes. *Evolution: International Journal of Organic Evolution*, 65(7):1945–1955.

Kauffman, S. A. et al. (1993). *The origins of order: Self-organization and selection in evolution*. Oxford University Press, USA.

La Cava, W., Spector, L., and Danai, K. (2016). Epsilon-lexicase selection for regression. In *Proceedings of the Genetic and Evolutionary Computation Conference 2016*, pages 741–748.

Lehman, J. and Stanley, K. O. (2011). Novelty search and the problem with objectives. In *Genetic programming theory and practice IX*, pages 37–56. Springer.

Mathias, H. D. and Ragusa, V. R. (2016). An empirical study of crossover and mass extinction in a genetic algorithm for pathfinding in a continuous environment. In *2016 IEEE Congress on Evolutionary Computation (CEC)*, pages 4111–4118. IEEE.

Millidge, B., Tschantz, A., Seth, A., and Buckley, C. (2021). Understanding the origin of information-seeking exploration in probabilistic objectives for control. *arXiv preprint arXiv:2103.06859*.

Mouret, J.-B. and Clune, J. (2015). Illuminating search spaces by mapping elites. *arXiv preprint arXiv:1504.04909*.

Ochs, I. E. and Desai, M. M. (2015). The competition between simple and complex evolutionary trajectories in asexual populations. *BMC evolutionary biology*, 15(1):1–9.

Oliveto, P. S., Paixão, T., Heredia, J. P., Sudholt, D., and Trubenová, B. (2018). How to escape local optima in black box optimisation: When non-elitism outperforms elitism. *Algorithmica*, 80(5):1604–1633.

Orr, H. A. (2005). The genetic theory of adaptation: a brief history. *Nature Reviews Genetics*, 6(2):119–127.

Peischl, S., Dupanloup, I., Kirkpatrick, M., and Excoffier, L. (2013). On the accumulation of deleterious mutations during range expansions. *Molecular ecology*, 22(24):5972–5982.

Peischl, S. and Excoffier, L. (2015). Expansion load: recessive mutations and the role of standing genetic variation. *Molecular ecology*, 24(9):2084–2094.

Peischl, S., Kirkpatrick, M., and Excoffier, L. (2015). Expansion load and the evolutionary dynamics of a species range. *The American Naturalist*, 185(4):E81–E93.

Poli, R., Kennedy, J., and Blackwell, T. (2007). Particle swarm optimization. *Swarm intelligence*, 1(1):33–57.

Ragusa, V. R. and Bohm, C. (2021). Connections between noisy fitness and selection strength. In *ALIFE 2021: The 2021 Conference on Artificial Life*. MIT Press.

Rozen, D. E., Habets, M. G., Handel, A., and de Visser, J. A. G. (2008). Heterogeneous adaptive trajectories of small populations on complex fitness landscapes. *PloS one*, 3(3):e1715.

Slatkin, M. and Excoffier, L. (2012). Serial founder effects during range expansion: a spatial analog of genetic drift. *Genetics*, 191(1):171–181.

Van Egeren, D., Madsen, T., and Michor, F. (2018). Fitness variation in isogenic populations leads to a novel evolutionary mechanism for crossing fitness valleys. *Communications biology*, 1(1):1–9.

Wang, Z. and Zhang, J. (2011). Impact of gene expression noise on organismal fitness and the efficacy of natural selection. *Proceedings of the National Academy of Sciences*, 108(16):E67–E76.

Whitley, D. (2001). An overview of evolutionary algorithms: practical issues and common pitfalls. *Information and software technology*, 43(14):817–831.

Wright, S. (1932). The roles of mutation, inbreeding, crossbreeding, and selection in evolution. *Proc. 6th Intern. Congress Genetics*, 1:356–366.

Wright, S. (1982). The shifting balance theory and macroevolution. *Annual review of genetics*, 16(1):1–20.

Simulations and the evolution of consciousness

Joshua Bensemann[*], Padriac O'Leary, Yang Chen, Ludmila Miranda-Dukoski, and Michael Witbrock

University of Auckland

Abstract

We hypothesize that the emergence of consciousness in humans is directly related to the complexity, number of, and evolution of specialized cognitive systems. Here, we present our rationale and plan for an ongoing project to investigate the pathway to the emergence of consciousness via computer simulations of humans' evolutionary niche using artificial-intelligence agents. Agents will contain subsets of the specialized cognitive systems and will complete tasks modeled after pressures encountered by early humans. We will observe whether the increase in cognitive complexity, measured by the number and complexity of the specialized cognitive systems, leads to an increase in task performance.

Introduction

Consciousness has been the topic of speculation and analysis across multiple fields in the academic community. A drawback of the broad appeal that the study of consciousness has is that there is no consensus on the most basic concepts, including its definition. Some authors define a conscious agent as one that possesses a cognitive architecture with features such as memory, internal representations of the world, *et cetera* (Aleksander, 2007; Arrabales et al., 2010; Bensemann and Witbrock, 2021; Tononi and Koch, 2015). Proponents of Global Workshop Theory (Baars, 2005) go further along this route, suggesting that consciousness is itself a system built from interactions between highly-specialized cognitive systems (e.g., attention, memory, etcetera.). This definition implies that consciousness is an emergent property of highly-specialized cognitive systems (Zlomuzica and Dere, 2022). In other words, the evolution of consciousness in *Hominini* (i.e., modern humans and their ancestors) is taken to be directly related to the evolution of these specialized systems.

Our project focuses on investigating the emergence of consciousness in humans by replicating the conditions under which *Hominini* evolved. We will be defining consciousness using a third-person introspective model (Choifer, 2018). The specialized system central to this definition is the Theory of Mind (ToM; Premack and Woodruff, 1978). ToM is the capacity to create models that are used to predict the knowledge and motivations of others. Once an agent can model the minds of others, that agent can also apply that capability to itself, resulting in a model of its own mind.

We hypothesize that various evolutionary pressures lead to increases in complexity as well as the interconnectedness between various specialized cognitive systems such as attention, communication, and, critically, ToM. In order to methodically examine the characteristics of the set of interconnected specialized cognitive systems required to produce given levels of cognitive complexity along with the ecological niches that lead to these characteristics, we will build digital environments and agents using computer simulations and artificial intelligence (AI). This approach is known as synthetic ethology (MacLennan, 2007).

The core of our research strategy is the creation of a computer-simulated environment based on early hominin ecology. However, allowing computer scientists to create environments for testing can introduce an implicit bias towards producing desired results by only incorporating what they consider important (Laird, 2001). To minimize this, we are using current models of evolutionary ecology to guide the development of the simulation. Our initial approach will build on the human evolution models of Kim Sterelney, who views our cognitive evolution as based on the gradual progression of interaction between individual and social feedback loops (Sterelny, 2012; Sterelny et al., 2013).

Our work on creating artificial replicas of early hominin environments is in progress. Once completed, we will introduce AI agents. Various agents controlled by different cognitive models of varying complexity will be tested. Controlling both the composition of the environment and the agent's cognitive makeup will enable us to experimentally identify any advantages that various sets of specialized cognitive systems afford us. In essence, we are performing a step-wise regression to generate enough data to analyze the characteristics of the set of specialized cognitive systems that might underpin human consciousness while accounting for factors such as model complexity.

[*]Email: josh.bensemann@auckland.ac.nz

Conscious Agents

Our AI agents will have various subsets of the specialized cognitive systems whose development is thought to correlate to the emergence of consciousness. The agents' goal will be to complete tasks thought to have been encountered by early humans. We will observe whether cognitive complexity leads to improvements in task performance.

Our approach to building artificial consciousness is similar to those suggested by others in prior work (e.g., Aleksander, 2007; Horton et al. 2013). One such approach is the ConsScale (Arrabales et al., 2010), an ordinal scale developed to incorporate other consciousness models. To reach each level in the scale, the agent must possess a minimum set of architectural and behavioral features from all previous levels, with levels ordered based on the likely phylogenetic path to our species' consciousness. Pre-existing plans such as the ConsScale will be used as starting point for our agents. However, the development will be guided by the work of others to ensure its cognitive development is consistent with our working definition of consciousness.

We turned to evolutionary psychology to make an educated guess about the highly-specialized cognitive systems that we must be included. Mounting evidence from the field suggests that sociality and solving other niche-specific problems played a significant, if not pivotal, role in the evolution of human consciousness and cognition. Of the number of systems involved in human socialization, ToM is of particular interest to us and the current research.

The first generation of our agents will be built using components from the various existing computational ToM models. These core components include beliefs, desires, and memory (or functional equivalents). We will inject ToM into AI agents and provide them with basic knowledge of the conditions of their virtual environment and an array of preferences for manipulating the contents of the environment. Our ultimate aim is to build an agent capable of cooperative behavior and formulate causal stories about their environments and other agents.

We know from the literature that various versions of cognitive capacities will produce varied results. For example, variations of ToM components can have noticeable effects on an agent's performance when learning to compete or cooperate with other agents. Experiments with the recursive aspects of ToM have shown that agents who could model deeper levels of recursion increased group performance when cooperating and bested competitors who possessed lesser recursive capabilities (Devaine et al., 2014). Similarly, having a more extended memory of the past actions of other agents allows an agent to better compete or cooperate with other agents (Anh et al., 2011). However, increasing memory length increases the cognitive cost, which might outweigh the gain in performance (Han et al., 2012).

The Environment

Deriving the environment that led to the emergence of consciousness is a critical requirement in understanding the conditions that gave rise to consciousness. By recreating the environment that human ancestors evolved in digitally, other researchers have uncovered conditions that may have led to the evolution of communication (Gong and Shuai, 2013; MacLennan, 2007; Miikkulainen and Li, 2016). For example, Miikkulainen and Li created a "jungle world" where pairs of agents' fitness increased if both agents chose to hunt or both chose to mate. They demonstrated that the evolution of communication was unnecessary when the simulation was full-observable to both agents. Communication became necessary when the environment was partially-observable and increased fitness when shared information was needed for cooperation. While our simulation will not necessarily require evolution, the principle is the same; our environment will be designed to test whether cognitive components provide any fitness advantage to agents that possess them. This allows a theoretical test for the utility of consciousness by adding its theoretical precursors to the agents.

Our core design principle for the environment is replicating resource-gathering and predation problems from our evolutionary niche. There will be multiple agents within an environment, each having partial information about the world. An agent's ability to forage individually and in groups will affect their fitness. Group activities include hunting or division of labor to increase overall fitness. By having both group and independent tasks, the environment will test fitness values of agents when acting alone compared to acting within a group.

The first version of the environment will be a 2D turn-based grid word. These settings were chosen to begin testing our agents as quickly and efficiently as possible as our agents will not require complex visual recognition systems as in the agents tested by the animal-AI testbed (Crosby et al., 2020). By creating a grid-based system, we can provide the agents with a simpler world representation. However, we will implement various grids - each containing information from a different sensory modality - to increase environmental complexity.

Multiple metrics such as survival time, resources gathered, and others will be used to measure evolutionary fitness. We will also measure cognitive efficiency by comparing the cognitive complexity of the agents to their task scores. Cognitive complexity will be measured using metrics such as the number of operations required for decisions, memory size, learning rates, or any commonly used metric in either psychology or the computer sciences. Suppose we normalize any survival metrics we use concerning an agent's cognitive complexity and show that an agent's gains in survival exceed the price. In that case, we have evidence that such cognitive capacities evolved due to their significant advantage.

References

Aleksander, I. (2007). Why axiomatic models of being conscious? *Journal of Consciousness Studies*, 14(7):15–27.

Anh, H. T., Moniz Pereira, L., and Santos, F. C. (2011). Intention recognition promotes the emergence of cooperation. *Adaptive Behavior*, 19(4):264–279.

Arrabales, R., Ledezma, A., and Sanchis, A. (2010). Consscale: A pragmatic scale for measuring the level of consciousness in artificial agents. *Journal of Consciousness Studies*, 17(3-4):131–164.

Baars, B. J. (2005). Global workspace theory of consciousness: toward a cognitive neuroscience of human experience. *Progress in brain research*, 150:45–53.

Bensemann, J. and Witbrock, M. (2021). The effects of implementing phenomenology in a deep neural network. *Heliyon*, 7(6):e07246.

Choifer, A. (2018). A new understanding of the first-person and third-person perspectives. *Philosophical Papers*, 47(3):333–371.

Crosby, M., Beyret, B., Shanahan, M., Hernández-Orallo, J., Cheke, L., and Halina, M. (2020). The animal-ai testbed and competition. In *NeurIPS 2019 competition and demonstration track*, pages 164–176. PMLR.

Devaine, M., Hollard, G., and Daunizeau, J. (2014). Theory of mind: did evolution fool us? *PloS One*, 9(2):e87619.

Gong, T. and Shuai, L. (2013). Computer simulation as a scientific approach in evolutionary linguistics. *Language Sciences*, 40:12–23.

Han, T. A., Pereira, L. M., and Santos, F. C. (2012). Corpus-based intention recognition in cooperation dilemmas. *Artificial Life*, 18(4):365–383.

Horton, J. D., Francis, M., and Sußenburger, E. (2013). A long term proposal to simulate consciousness in artificial life. In *ICAART (2)*, pages 389–394.

Laird, J. E. (2001). Using a computer game to develop advanced ai. *Computer*, 34(7):70–75.

MacLennan, B. (2007). Making meaning in computers: Synthetic ethology revisited. In *Artificial Cognition Systems*, pages 252–283. IGI Global.

Miikkulainen, R. and Li, X. (2016). Evolving artificial language through evolutionary reinforcement learning. In *ALIFE 2016, the Fifteenth International Conference on the Synthesis and Simulation of Living Systems*, pages 484–491. MIT Press.

Premack, D. and Woodruff, G. (1978). Does the chimpanzee have a theory of mind? *Behavioral and brain sciences*, 1(4):515–526.

Sterelny, K. (2012). *The evolved apprentice*. MIT press.

Sterelny, K., Joyce, R., Calcott, B., and Fraser, B. (2013). *Cooperation and its evolution*. MIT Press.

Tononi, G. and Koch, C. (2015). Consciousness: here, there and everywhere? *Philosophical Transactions of the Royal Society B: Biological Sciences*, 370(1668):20140167.

Zlomuzica, A. and Dere, E. (2022). Towards an animal model of consciousness based on the platform theory. *Behavioural brain research*, 419:113695.

A Modeling and Experimental Framework for Understanding Evolutionary and Ecological Roles of Acoustic Behavior Using a Generative Model

Reiji Suzuki, Shinji Sumitani, Chihiro Ikeda and Takaya Arita

Nagoya University
Furo-cho, Chikusa-ku, Nagoya 464-8601, Japan
reiji@nagoya-u.jp

Introduction

The emergence and evolution of acoustic interactions have been studied in ALife (Suzuki and Cody, 2019), including the evolution of communication and language (Nolfi and Mirolli, 2010). There is also interest in applying complex systems approaches to ecoacoustics (Farina and Gage, 2017) for conservation studies (Eldridge, 2021).

Agent-based modeling has contributed to understanding the evolution of complex signals, and recent research focused on the evolution of audible or acoustic signals (Eldridge and Kiefer, 2018; Kadish et al., 2019; Suzuki et al. 2021). The emerging signals purely from computational processes may have different types of complexity from those in natural ecological systems, making it difficult to discuss their roles in ecological contexts and limiting interactions between artificial and natural systems. On the other hand, deep learning techniques enabled us to generate artificial objects that have equivalent complexity to natural objects, such as images, videos, audio, and texts, by using a large network with a large data set of natural objects, including animal vocalizations (Sainburg et al., 2020).

We propose a research framework for understanding the evolutionary and ecological roles of acoustic behavior by combining agent-based modeling and machine learning (Fig. 1), focusing on bird vocalizations, which is one of the significant components that create natural soundscapes. We use a latent space of a generative model as a genotype space and regard a generated object as a corresponding phenotype in an evolutionary model, then further observe the roles of the evolved phenotypes in a real ecological context.

This paper introduces two independent trials to show the feasibility of the approach. We first introduce an agent-based evolutionary model of syllables in Zebra Finch songs based on the coevolution of syllable structures and preferences. Then, we show that artificially generated songs of Japanese Bush Warbler can affect the behavior of conspecifics in the wild.

Evolutionary Model

The model is inspired by a seminal mathematical model of sympatric speciation by sexual selection (Higashi et al., 1999), assuming that the spectral structure of syllables of Zebra Finch (*Taeniopygia castanotis*) songs could have effects on the mating preferences of females. We created a two-

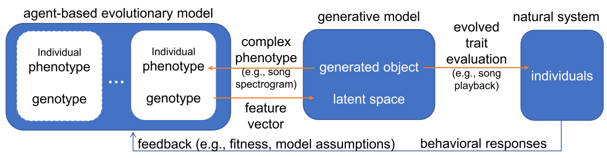

agent-based evolutionary model
Individual phenotype
genotype
Individual phenotype
genotype
complex phenotype (e.g., song spectrogram)
feature vector
generative model
generated object
latent space
evolved trait evaluation (e.g., song playback)
natural system
individuals
feedback (e.g., fitness, model assumptions)
behavioral responses

Fig. 1: Proposed framework.

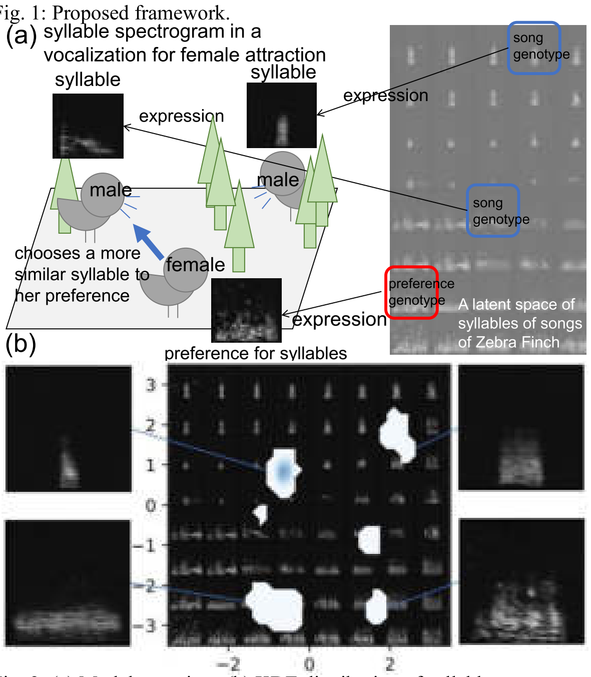

(a)
syllable spectrogram in a vocalization for female attraction
syllable
syllable
expression
expression
male
male
chooses a more similar syllable to her preference
female
expression
song genotype
song genotype
preference genotype
A latent space of syllables of songs of Zebra Finch
(b)
preference for syllables

Fig. 2: (a) Model overview, (b) KDE distribution of syllable genotypes.

dimensional latent space of syllables of captive Zebra Finch. We trained a variational autoencoder (an encoder that has 8 convolutional layers and three fully connected layers that represents the two-dimensional latent space, and a decoder that has the symmetric to the encoder) using 128 x 128 spectrogram images of syllables in songs (18,000 syllables).

We assume male and female populations, each composed of 200 individuals. Each individual has two positions on the latent space as genotypes (Fig. 2 (a)). Each male individual expresses a spectrogram image as his syllable generated from the decoder network using a syllable genotype as an input. Each female also expresses a spectrogram generated from a preference genotype as her preference for syllables. In the mating process, each female evaluates every male using the formula: $exp(-Wx)$, where x is the average difference in the pixel values between the syllable spectrogram of the focal

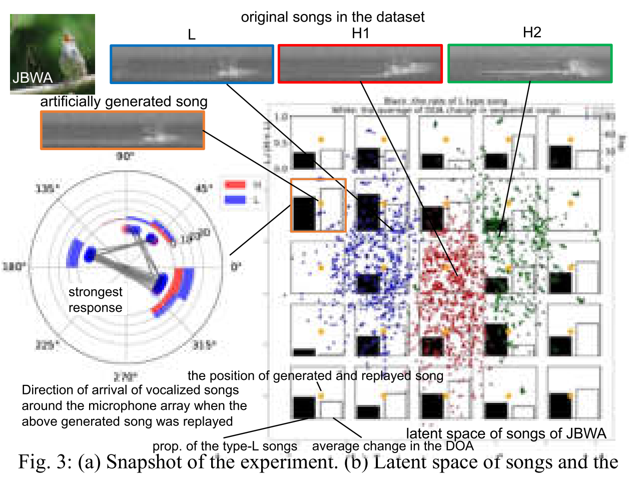


Fig. 3: (a) Snapshot of the experiment. (b) Latent space of songs and the response of the focal individual against replayed artificial songs.

male and the preference spectrogram of the focal female, and W is a coefficient. Then, the female selects a male stochastically proportional manner to the evaluation values and produces a pair of male and female offspring using a BLX-alpha crossover and a mutational change in the genotypic values with a small probability.

Fig. 2 (b) illustrates the KDE (kernel density estimation) distribution of syllable genotypes of males accumulated over 600 generations in a typical trial. The background images represent spectrograms generated using a genotype on the corresponding position, showing that the space well reflects the acoustic property of syllables. A prezygotic isolation of individuals emerged through a kind of runaway process in that the population segregated into a few subgroups. The population was composed of a majority group with a relatively simpler vocalization pattern (top left) while the other small subgroups had more unique and complex acoustic features. This is because simpler phenotypes would be moderately preferred by many females while complex phenotypes would be chosen by the limited number of females with a high probability. The evolved complex syllables tended to exist on the peripheral or outer side of the distribution of natural syllables in the space. The next step is to see whether and how the real Zebra Finch individuals may respond to artificially modified songs that incorporate these evolved syllables, and to incorporate song structures into the model.

Playback Experiment

We conducted a playback experiment to see if artificially created sounds using a generative model can provoke responses from wild songbirds in the forest. We focused on the Japanese Bush Warbler (*Horornis diphone*) (Fig. 3), a popular songbird species in Japan. Males sing two types of songs: type-H and type-L, similar but slightly different (Hamao, 2007). The frequency of the type-H song is relatively high, and the type-L song has intermittent whistles, and the frequency is relatively low. The type-L song is known as a threat to rivals in the vicinity because territory owners frequently use this type in the periphery of their territory.

We trained a variational autoencoder, which has a similar structure to in the previous section, using the spectrograms of type-H songs and type-L songs (3,000 songs) recorded in another field observation of a single male individual. The dots (green: type-L, blue and orange: type-H) in Fig. 3 (right) represent the positions of original data on the 2D latent space.

We used a similar protocol to that employed in (Suzuki et al. 2018). A loudspeaker was placed in an open space surrounded by trees in the experimental forest in Nagoya University. We conducted a 15-minute playback session for each artificial song generated from a corresponding position on the latent space indicated by an orange dot in Fig. 3 (right) on the same individual (25 trials in total). The song was replayed at an interval of 30 seconds. A male individual of the same species came into the space and then wondered and sang songs around the speaker, which is known as more aggressive responses against conspecific song playback according to the field observation and the previous reports (Hamao 2007, Suzuki et al. 2018). We measured the proportion of the type-L songs among all response songs and the average change in the direction of arrival of the consecutive songs estimated by a microphone array around the loudspeaker using HARKBird (Sumitani et al. 2020). The higher values of these indices indicate more aggressive behavior wandering and singing type-L songs around the loudspeaker more actively.

Each bar graph in Fig. 3 (right) indicates the two indices in the corresponding case of the replayed song (generated with the feature (orange dot) on the graph). This illustrates that the target male responded actively against artificially created bird songs generally, except for a few cases when the individual flew away (no bars). This implies that those artificial sounds have essential acoustic properties to be recognized as conspecific songs. A circular histogram (left) illustrates the behavioral pattern in the case of the most aggressive response of the focal individual. Notably, this replayed song (orange) located close but a bit outside position from the clusters of original type-L songs, implying that its artificial but unique property of the type-L-like song could bring about active responses. The next research step is to explore the fitness definition or model assumptions that will bring about the evolution of the unique trait in the evolutionary model.

Conclusion

We introduced an individual-based evolutionary model of sexual selection for song syllables and preferences using a generative model, showing that the complexity of spectral features brought about the asymmetric segregation of syllables. We also demonstrated that artificially generated songs can evoke aggressive responses of wild birds in the forest. Still, these are from independent works, they showed that combining modeling and experimental approaches using a generative model can contribute to further understanding of evolutionary and ecological roles of bird vocalizations.

Acknowledgements

We thank Kazuhiro Wada (Hokkaido Univ.) for providing the dataset of Zebra Finch. This work was supported in part by JSPS/MEXT KAKENHI: JP21K12058, JP20H00475, JP19KK0260 and JP17H06383 in #4903 (Evolinguistics).

References

Eldridge, A. and Kiefer, C. (2018). Toward a synthetic acoustic ecology: Sonically situated, evolutionary agent based models of the acoustic niche hypothesis. In *Proceedings of ALIFE 2018: The 2018 Conference on Artificial Life*, pages 296-303.

Eldridge, A. (2021). Listening to ecosystems as complex adaptive systems: toward acoustic early warning signals. In *Proceedings of ALIFE 2021: The 2021 Conference on Artificial Life*, Paper No: isal_a_00450, 20 (8 pages).

Farina, A. and Gage, S. H. (2017). Ecoacoustics: The Ecological Role of Sounds. John Wiley and Sons.

Hamao, S. (2007). Japanese Bush Warbler. *Bird Research News*, 4 (2) (2 pages).

Higashi, M., Takimoto, G. and Yamamura, N. (1999). Sympatric speciation by sexual selection. *Nature*, 402 (6761): 523-526.

Kadish, D., Risi, S., and Beloff, L. (2019). An artificial life approach to studying niche differentiation in soundscape ecology. In *Proceedings of ALIFE 2019: The 2019 Conference on Artificial Life*, pages 296–303.

Nolfi, S., Mirolli, M. (eds) (2010). *Evolution of Communication and Language in Embodied Agents*. Springer, Berlin.

Sainburg,T., Thielk, M. and Gentner, T. Q. (2020). Finding, visualizing, and quantifying latent structure across diverse animal vocal repertoires, *PLoS Computational Biology*, 16 (10): e1008228.

Suzuki, R. and Cody, M. L. (2019). Complex systems approaches to temporal soundspace partitioning in bird communities as a self-organizing phenomenon based on behavioral plasticity, *Artificial Life and Robotics*, 24, 439-444.

Suzuki, R., Sumitani, S., Matsubayashi, S., Arita, T., Nakadai, K. and Okuno, H. G. (2018). Field observations of ecoacoustic dynamics of a Japanese bush warbler using an open-source software for robot audition HARK, *Journal of Ecoacoustics*, 2: #EYAJ46 (11 pages).

Sumitani, S., Suzuki, R., Chiba, N., Matsubayashi, S., Arita, T., Nakadai, K. and Okuno, H. G. (2019). An Integrated Framework for Field Recording, Localization, Classification and Annotation of Birdsongs Using Robot Audition Techniques - Harkbird 2.0, *Proceedings of 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP2019)*, pp. 8246-8250.

Suzuki, R., Seki, R., Banno, T., Kawai, K. and Arita, T. (2021). An artificial creature approach to the origin of acoustic communication, *Proceedings of the 2021 Conference on Artificial Life (ALIFE2021)*, Paper No.: isal_a_00421, 87 (3 pages).

Evolution of Developmental Strategies in NK Fitness Landscapes

Jacob Ashworth[1], Yujung Lee[1], Jackson Shen[1], Edward Kim[1], Zach Decker[1] and Jason Yoder[1]

[1]Rose-Hulman Institute of Technology, Terre Haute, IN 47803
ashworjs@rose-hulman.edu

Abstract

Evolution and development are related processes although their relationship is still not well understood. Attempts to explore their relationship are challenged by scales of time and space, but also by the limitations of studies focused on specific constraints of model organisms. To help gain insight into these phenomena, we create an abstract, general model of a developmental process that guides an agent's trajectory through a "tunably rugged" NK fitness landscape. The developmental process is represented by a genotype that is evolved and allows us to investigate periods of exploration and exploitation as they relate to periods of an agent's lifetime and a given landscape's difficulty. Results show that evolution selects for time-sensitive periods of exploration and exploitation, which vary with the difficulty of the landscapes being traversed. Furthermore, our analysis suggests that phenotypic diversity via random exploration present in both early and mid-life can aid the development of superior phenotypes.

Introduction

Learning, development, and evolution work together to produce diverse forms of adaptive organisms. Whereas the relationship between learning and evolution has frequently been explored, development is often overlooked or regarded as a subset of the learning process in many models (Soltoggio et al., 2018). Yet, development is distinct from learning in that it involves structural changes in both the body and brain over time according to a set of processes, though such processes themselves may allow plastic adaptations to environmental cues (West-Eberhard, 2005). Sensitive periods exemplify one such process, and necessitate that development is distinguished from learning with varying degrees of plasticity at different life periods (Knudsen, 2004). Given the limited efforts to explore the relationship between development and evolution, there has been little progress in understanding why certain developmental processes have emerged from evolution as well as how the specific forms observed in nature have arisen (Frankenhuis and Walasek, 2020).

Most models of development have been focused on specific organisms and the various constraints on them. There is an opportunity, however, to develop more abstract models that can explore developmental plasticity generally across different developmental processes (Belew, 1990; Sommer, 2009; Smart, 2019). One example of such an abstraction is the NK model by Kauffman and Levin (1987), which has been successfully applied to a variety of modeling tasks (Altenberg, 1996; Geard et al., 2002; Fragata et al., 2019; Rhodes and Dowling, 2018). By employing this model as a "tunably rugged" fitness landscape, our work seeks to explore the evolution of developmental strategies in an abstract, but general manner. Our approach is to evolve a representation of different permutations of developmental processes throughout an organisms' full development. This simple developmental "program" is then used to guide an individual from a starting location within an NK model toward, hopefully, locations with higher fitness.

The model and analysis in this work extend previous work in three important directions. First, evolution is shown to produce developmental steps that explicitly explore the landscape in a random manner. Second, evolution drives development to be time-sensitive, with non-uniform patterns of exploration and exploitation comprising the developmental strategy. Third, evolution selects for a more complex developmental strategy marked by multiple transitions between predominantly exploitative and exploratory phases, as the landscape becomes more complex with a greater number of interdependent factors. Furthermore, this work presents a simple yet effective model that can simulate various interactions between evolution and development.

The primary aim of this work is to explore the evolution of development in an abstract model. The rest of this paper is organized to accomplish this goal. First, we summarize related work in in evo-devo models, abstract models in biology, developmental models, and adaptive walks. Next, we provide an overview of the model in this work, including the landscape, agents, actions, and evolution. We then present results from experiments in a gradual sequence of generalization for starting locations, different landscapes, and evolutionary runs. Subsequently, we discuss the major findings and explore them in the context of evolution and development. Lastly, future work identifies major directions to extend upon the results presented in this work.

Related Work

In this section we will discuss areas of relevant literature and their shortcomings. First, we provide a brief background on evo-devo models to offer insight into their particular challenges and limitations. Next, we review the use of abstract models, especially relating to understanding complex phenomena similar to our own domain of interest. Finally, we review developmental models and identify a lack of desirable features which we seek to address.

Evo-Devo Models

Evolutionary developmental biology (evo–devo) has stimulated biological research enormously, both empirically and theoretically (Müller, 2007). For the last two decades, Evo-devo models have been mostly based on evo-devo model organisms such as *Drosophila melanogaster* and *Caenorhabditis elegans* by building on the analysis of those organisms. Although there has been a fruitful expansion in this era of research, the past trend—increasing numbers of organism species such as *Tribolium castaneum* and *Nasonia vitripennis* (Roth and Hartenstein, 2008; Lynch et al., 2006) has led to severe challenges in the scientific methodology and technical difficulties (Sommer, 2009). These new models have driven researchers to build more sophisticated tool kits to investigate the mechanisms of evolutionary change in developmental processes. Developing these involve gene knockout or knockdown, and experimental manipulation which necessitates high complexity and scientific precision. Furthermore, these methods mostly depend on empirical optimizations, which are largely species specific such that protocols cannot be transferred from one organism to another (Sommer, 2009). To overcome these limitations and to capture interactions among complicated phenomena—evolution, development, learning—the models must necessarily become extremely simplified (Belew, 1990) such that an abstract computational model would be advantageous for modeling evo-devo.

Abstract Models

Abstract models have been successfully used to study high level processes, which are otherwise difficult to investigate due to their complexity and scale in time and space. Various abstract models have been developed to explore the relationship between learning and evolution.

One early abstract model developed by Hinton and Nowlan (1987) demonstrated how learning can guide evolution through an idealized simulation model. Similarly, Kauffman and Levin (1987) introduced another abstract model, the NK model, which is a tunably rugged landscape adjustable by two parameters N and K. As this model exhibits how interactions affect dynamics on rugged landscapes, it has been applied to various fields such as learning strategies (Campbell et al., 2020) and ontological development (Panchanathan and Frankenhuis, 2016; Walasek et al., 2021). NK landscapes are commonly used for examining adaptive walks, which proceed to fitter neighbors resulting from strategies such as ascent, steepest ascent, or minimum ascent (Pitzer and Affenzeller, 2012; Wilke and Martinetz, 1999; Hebbron et al., 2008; Kauffman and Levin, 1987). Extending the NK model, Hebbron et al. (2008) and Wilke and Martinetz (1999) demonstrated significant behavior changes of adaptive walks, while Park et al. (2015) implemented greedy adaptive walks to study haploid asexual population.

Recently, Todd et al. (2020) developed an idealized model of lifetime and evolutionary learning and examined the effect of task-difficulty on the optimal trade-off between learning and evolution. In their model, there are two types of lifetime learning—stochastic hill-climbing and steepest hill-climbing, but the model focuses on learning rather than development and fails to include mechanisms for random exploration, which we believe to be an important component of lifetime adaptation. Overall, NK fitness landscapes have been successfully used to study high level processes in multiple fields and as a result appear viable as a candidate model for studying evo-devo as well.

Developmental Models

There is a need for abstract developmental models which are specific to the field of evo-devo that could address scientific questions in a computational manner. By applying abstract models to developmental, there is an opportunity to gain a deeper understanding of the evolution of development.

Some developmental models have examined the evolutionary selection pressures that produce sensitive periods (Panchanathan and Frankenhuis, 2016; Walasek et al., 2021; Frankenhuis and Walasek, 2020). Unlike a two-stage life history, in which organisms first obtain environmental cues and later develop phenotypes, Walasek et al. (2021) proposed a model where individuals incrementally respond to local environmental conditions with sensitive periods emerging during life. In this model, if an environmental change or migration occurs during its lifetime, then an organism must infer the environmental state and consider environmental change–becoming a complicated inferential task. Due to the complicated inferential nature, the model is highly specific which is a limitation in a developmental context.

Given that most previous models are complex, lack significant mechanisms, and lack tasks with epistatic interactions, we have an opportunity to formulate a novel way of modeling developmental processes. Using this approach, we hope to identify sufficient conditions for the evolution. Furthermore, we aim to explore the relationship between difficulty and different developmental strategies. Ultimately, this paper's goal is to enable work utilizing a computational model to challenge and reformulate existing ideas and produce new hypotheses.

Methods

In this section, we explain the methods used to simulate our model, conduct experiments, and analyze results. First, we will discuss the NK fitness landscape and how it is used within this context. Next, we explain the constraints on the agents' development within the landscape. Afterwards, we describe the process in which the development strategies of the agents are evolved. We then explain the baseline models used for comparison to the evolved strategies. Finally, we describe the universal parameters given to the model across our experiments.

NK Model

In order to model the development of organisms within various environments, we used the NK model as described in (Kauffman and Levin, 1987). Within a NK landscape, the n-value presents the dimensionality of a landscape and the k-value represents the number of epistatic interactions when determining a bitstring's fitness. As k increases its "ruggedness" or the number of local maxima increases. All experiments within this paper are run using landscapes of $n = 15$ and k varying from 0 (smooth) to 14 (rugged). Within the real world, a lower k represents a situation where there are fewer factors in play, and a higher k represents a situation where there are many factors in play. The NK model allows us to adjust the difficulty of the landscape an agent traverses through, allowing us to observe trends within those agents across different model parameters. Agents are placed within this landscape and improve as follows.

Agents and Actions

We define agents to start at a specific location (bitstring sequence) within an NK fitness landscape, this represents the starting point of the agent's developmental strategy. Rather than evolving (i.e. mutating) the starting locations in the landscape, in this work we focus on a consistent starting location and focus on the evolution of a "developmental program." Example target organisms for this model include Salmon and other migrant species (e.g. sea turtles) that are consistently born at a particular location and experience a similar lifelong sequence of challenges.

For our model, the "developmental program" consists of encoding two types of actions over an agent's lifetime: looking and walking. By separating looking actions and walking actions, we get further insight into the patterns of exploration and exploitation that emerge. Taking a look action allows an agent to collect information about the fitness of a bitstring location exactly one bit different from the agent's current location. Alternatively, taking a walk action allows the agent to use its information collected through looking actions to move from its current location to whichever location had the highest recorded fitness. If the highest fitness level of the collected information is lower than the current fitness, then the agent does not move. After a walk action, the information gathered from previous looks is discarded. Taking combinations of these actions effectively allow an agent to emulate exploratory and exploitative developmental patterns, in a non-Markovian manner.

When an agent explores every possibility around it before walking, i.e. taking all possible look actions before a walk action, we describe it as a steepest hill climb. Conversely, when an agent does not take any looking actions before a walk action, we describe it as a random walk, as it results in the agent blindly moving to a new location without any knowledge of its surrounding landscape. Figure 1 illustrates the actions of an agent on a small (N=3) landscape with a genome of length 4.

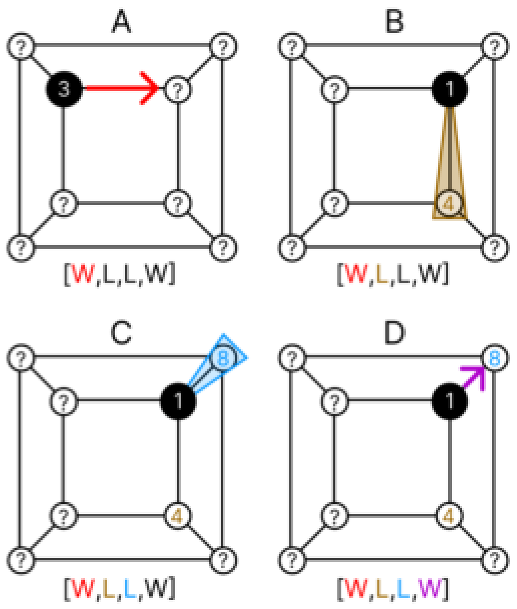


Figure 1: Illustration of an agent performing actions according to its genotype. W's represent walks and L's represent looks. Each graph (A-D) shows the connectivity of an N=3 landscape, with the nodes representing locations and their labels representing fitness (with ?s representing unknown fitness). A: The agent's initial location is represented by the black circle. It takes its first action as encoded by its genotype, a walk (W, red). There were no previous look actions so it takes a random walk (in this case to the right). B: Agent's new location is again represented by the black circle, but this time takes a look action (L, brown), resulting in a random adjacent location's fitness value being observed and recorded. C: Agent is still in the same location as in B and takes another look action (L, blue), resulting in another random adjacent location's fitness value being observed and recorded. D: Agent is still in the same location as it was in B and C, but this time takes a walk action (W, purple). The agent moves up and to the right to the highest fitness location recorded (8) and forgets the previous observations.

In this work, we define the lifespan of an agent to be its full developmental period, with a predetermined number of walk and look actions distributed across its lifespan. We take this model, as constrained in the following section, to provide a generalization for the development of organisms. More specifically, we aim to study high level processes guiding a lifetime learning strategy including exploitation and exploration, as represented by combinations of looking and walking steps. While we primarily focus on the evolution of lifetime learning strategies, the model could easily be extended to capture further biological processes, such as sensitive periods.

Evolutionary Algorithm

Within this model, we consider the combination of looking and walking actions over each agent's lifetime development to be its genotype, that is to say the encoding of a process that guides its developmental trajectory. These genotypes are constrained by having a set number of looking and walking actions, to be distributed over the lifetime of the organism. In other words, we are looking at the times within the agent's lifespan that it chooses to explore versus exploit.

By implementing these constraints, we create a common ground to analyze the genotypes. The genotypes of the organisms are evolved through a genetic algorithm as follows. First, the organisms within the population are selected based on their fitness at the end of their lifetime (i.e. fitness at the final location in the NK model). The top 50% of the organisms are directly copied to the next generation, and the remaining 50% of the new generation is comprised of mutations of these survivors, where one of the walk steps is randomly moved to a different part of the strategy to modify how look steps are distributed throughout the strategy. The new generation of organisms are then allowed to develop starting from the same starting locations as the previous generation. Through this process, the population of agents evolves their developmental strategy, finding a genotype that generally leads to a higher ending fitness.

In essence, an agent can develop using one of two primary developmental processes: exploration and exploitation. Exploration allows an organism to take risks for the sake of improvement, while exploitation focuses on refinement and efficiency. Within this model, we define exploration as taking multiple consecutive walk actions (resulting in "random walk" steps), while exploitation is taking look actions before a walk action. By allowing agents to evolve the distribution of their looking steps, i.e. choose when in their life to have exploratory and exploitative periods, we can observe the tendencies of these exploratory and exploitative periods to emerge from the evolutionary process.

Baseline Genotypes

We establish several baseline strategies for comparison with the evolved strategies. In order to do so, we will use three

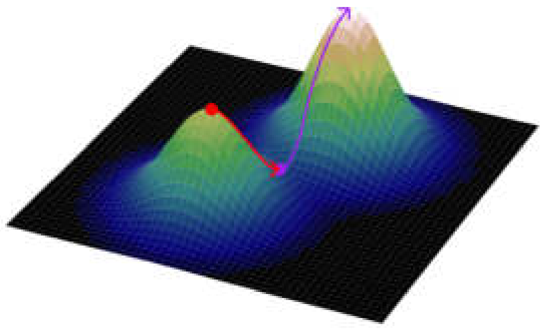


Figure 2: An artistic rendering of an NK landscape to demonstrate the behavior of agents. The starting location of an agent is marked as a red dot. The red path continuing from this depicts a possible path taken with a random walk action. The random walk ends at the red arrow, where the agent then continues its development with a steepest hill climb, reaching a higher optima than its original location. This demonstrates the utility in random walks allowing agents to escape from a local optima.

predetermined types of Walkers for comparison:

A 'Random Walker' (RW) takes exclusively walk actions throughout its entire lifetime. This simulates purely random decision making, with no environmental feedback. This walker is not expected to perform well, but will be used to demonstrate the importance of using look actions.

A 'Steepest Hill Climber' (SHC) walker looks at every possible adjacent location on the landscape before ever taking a walk action - purely steepest hill climb. This means that a 'Steepest Hill Climber' will always take the path of steepest ascent, and creates purely-exploitative behavior.

An 'Alternating SHC/RW' walker was designed to combine the strengths of RW and SHC. It accomplishes this by repeatedly taking two SHC steps and then a single RW step. This means it will spend the majority of its developmental process exploiting and ascending to a local optima, but will also have RW steps that allow it to escape local optima it may find itself in, creating potential performance improvements over the normal SHC strategy. Figure 2 provides a simple visualization how an agent at a local optima can use a RW step followed by a SHC step in order to escape a local optima.

Experimental Setup

Across all of our experiments, the parameters of the evolutionary process were held constant. Each evolutionary process consisted of a population of 100 initially randomized strategies simulated for 50 generations. These parameters