A Comparative Analysis of Foraging Strategies for Swarm Robotics using ARGoS Simulator

1st Ambikeya Pradhan
Dept. of Industrial Engineering (DII)
University of Trento
Trento, Italy

2nd Marta Boavida
Dept. of Industrial Engineering (DII)
University of Trento
Trento, Italy

3rd Daniele Fontanelli
Dept. of Industrial Engineering (DII)
University of Trento
Trento, Italy
daniele.fontanelli@unitn.it

***Abstract*—In the field of exploration strategies for teams of autonomous vehicles, one relevant set of solutions build upon the so called foraging algorithms, which mimic the foraging strategies of animals and insects, such as bugs and/or ant colonies. In the literature, it is most often observed that the choice of the foraging strategy to be applied for a specific swarm robotics problem does not rely on quantitative and objective selection criteria but, rather, it is guided solely by qualitative guidelines. Hence, this paper proposes a quantitative review of four popular foraging strategies, namely solitary foraging, behavioural matching, stigmergical foraging and signalling. A quantitative evaluation of their performance in terms of collectible or goal acquisition in different operating scenarios is proposed together with a comparison of their computation times when the size of the swarm changes. The comparative simulations presented to provide evidence of the different approaches efficiency have been implemented with the ARGoS simulation tool.**

I. Introduction

Swarm robotics is becoming more and more popular in modern applications, ranging from search and rescue [14], patrolling [8] or detection of points-of-interest [4]. The main advantages of multi-agent applications are the intrinsic robustness of the solution and the high performance [1]. Foraging algorithms are biologically inspired and are gaining popularity in robotics. Many real-world tasks, such as exploration, mining or search and rescue, are application examples for foraging approaches.

The most simple foraging approach considers individuals. The bottom line, which is a directly proved proposition, is that at least some individuals must have narrower niches than the overall population niche [7]. In individual foraging, indeed, an individual searches for the goal resources (i.e. food for animals and insects) and plans an optimal path trajectory to the goal from the starting position. Individual foraging is classified into *solitary foraging* and *behavioural matching*. In solitary foraging the goal is to collect a certain number of small resources which are scattered in the individual surroundings. The goal defined generally for the navigational purposes is a path to the collectible objects, while the detection of such collectible objects is usually related to specific features given upfront (e.g. size, weight, distance, etc). As a result, the determined path is used to guide the agent back to the starting point or to the check region [10].

Behavioural matching, instead, comprises an additional step with respect to the solitary foraging. In this case, the individuals have the opportunity to interact with the other agents. In this case, the agent that has found the resource may share the information about the path towards the resource location or share with the other individuals the resource. It is not necessary for the following agent to engage or communicate with the forager in order to share the knowledge [10].

Opposite to the individual foraging, in the *recruited individual foraging* an individual gathers resources either for itself or its whole group. Unlike behavioural matching, in this approach the whole group gain collectively a benefit from an activity completed by an individual. For this class of approaches, in *stigmergy* individuals use their surroundings in order to store information. These individuals are capable of laying pheromone trails where they found possible resources that allow other group members to follow the same path. The probability of selecting the same path increases with the trail strength, which on its turn is usually directly proportional to the number of agents following the path. Resources for these agents are supposed to be in patches and ask for a sufficiently large group to be manipulated or used. Apart from navigation towards a goal, group members also use pheromones to rapidly recruit random individuals for a raid. Hence, the inputs provided to the group members are ”Navigation and resource recognition” as well as ”artificial pheromone recognition and understanding” [10, 7].

Another type of recruitment strategy involves *direct signalling* between individuals. Unlike stigmergy, signalling requires both the signaller and the receiver to be present at the same time, but the environment no longer needs to be altered by trails [2]. In some cases, such a direct contact is even used to enforce pheromone trail following. Similarly to stigmergy, a richer and easier-to-obtain resource has a higher frequency of returning foragers and it thus attracts more individuals. However, unlike in stigmergy, non-linear effects, like structure of a group, may influence the way that information is spread among the individuals [10, 15].

From the previous description, it is clear that different solutions are available and there is not a one-size-fits-all solution. Surprisingly, the selection criteria adopted for the different choice is almost entirely neglected, whereas the main guideline seems to be empirical: if the chosen algorithm works in the provided scenario, it is adopted [7]. Of course, this

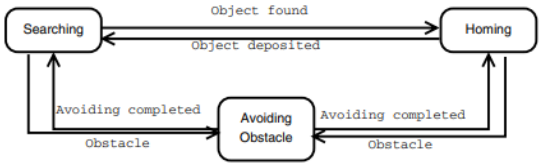


Figure 1. Finite state machine model of the robots implementing the foraging algorithm.

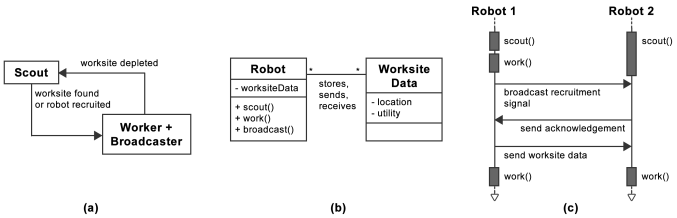


Figure 2. Proposed system model for the communication between *Scout* and *Worker/Broadcaster* (or *Robot 1* and *Robot 2*. (a) State chart, (b) class diagram and (c) sequence diagram.

trial and error approach cannot offer an exhaustive selection criterion, which is instead the aim of this paper. In order to prove the effectiveness of the different solutions and a comparison common-ground, we will use the following problem. We consider large swarms of simple autonomous robots for an exploration task, which covers a vast area. The resources, i.e. the object to be found, are clustered in random locations within an ARGoS simulation. These algorithms will be compared to the each other in the simulation designed arena and a quantitative analysis will be provided.

II. Models

We assume that the autonomous agents can communicate to each other through a range-and-bearing communication device, which allows robots in line-of-sight to exchange messages within a limited range. The peculiarity of this communication system is that a robot, upon reception of a message, computes the relative position (distance and angle) of the sender [3]. We assume, as customary [13], that two modules are implemented in the robot: a sensor (that manages the reception of the messages from other agents) and an actuator (that sets the message to send). In this paper, we assume that the communication is visual: the sender emits signals through a lighting system of changing colours (where each colour is coded for specific messages) and the receivers in range may detect the light locations through a calibrated omnidirectional camera.

To account for the different modalities of the agents, either *Searching* (i.e. agent in exploration mode) or *Homing* (i.e. agent returning to the starting location), a finite state machine is defined (see Figure 1). An additional module of *Avoiding Obstacle* is implemented in order to avoid collisions with other agents and/or fixed obstacles. In each modality, the agent sends a different message (i.e., emits a different colour) to the rest of the group in line of sight and within the limited visual ranging.

A. Autonomous system model

Autonomous systems with multiple entities are in general challenging to be analysed since they most often require a bottom-up approach to behavioural design. The emergent macro behaviour of the team as a whole is specified by the micro behaviour of the individual agents, which makes the programming of each individual very difficult. However, there are solutions in the literature helping to solve this problem, as listed below.

- *Information-Cost-Reward* (ICR) framework [11] for collective behaviour analysis represents a novel approach to understand robot swarms that perform foraging. Indeed, in foraging algorithms, robots search for *work-sites* in an unknown environment and (a) either perform work on them (e.g., in the case of area surveillance), or (b) collect items from the work-sites and bring them to a designated location (e.g., in resource collection applications). The ICR framework allows to relate the way in which robots obtain and share information about the work-site to the swarm ability to use that information to work efficiently given a particular task and environment [11].
- *Design Pattern Catalogue* (DPC) [11] for robot swarm foraging algorithms describes the design pattern that the swarm should follow. A design pattern provides high-level guidelines for the implementation of a particular robot behaviour and describes its impact on swarm performance. In this paper, we explore information exchange design patterns for robot swarm foraging [11]. The patterns are split into two categories that identify the pattern roles: *Information Transmitter Patterns* (entity transmission and information storing) and *Information Aggregation Patterns* (information exchange and type of behaviour data exchange)

The description of pattern feedback loops, parameters, forces and results utilises terminology of the ICR framework and considers a variety of experiments reported in the swarm robotics literature. Of course, not all pattern descriptions include all the properties listed above. For example, when a pattern has no parameters (e.g., for the solitary foraging), it does not influence the agent behaviour. Using this modelling approach, we identify three behaviours for the agents in the swarm, which are *Scout* (i.e. exploration in search for items), *Worker* (i.e. when an item has been found and needs to be processed on spot or brought back to the starting location) or *Broadcaster* (e.g. when the agents spreads its information in the swarm). The general system model for communication stemming from this analysis is summarised in Figure 2, while Figure 3 reports the Behaviour-Data Relations Modelling Language (BDRLM) representation for a generic foraging algorithm and implementing the general system model.

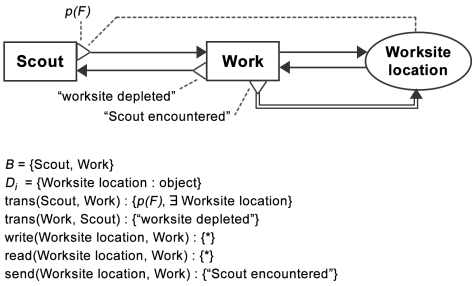


Figure 3. BDRML representation of the system model proposed in Figure 2.

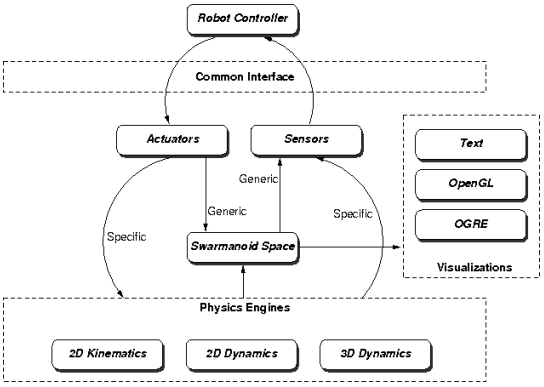


Figure 4. The architecture of ARGoS for a single robot.

III. Implementation

ARGoS [9] is a discrete-time simulator conceived for multi-robot systems acting in 3D spaces. The main code is entirely written in C++ and it is all based on free software libraries. Each entity active in the 3D space is modelled as a combination of: a physical structure; a set of sensors and actuators; a controller module. The simulator includes an extensive set of sensors and actuators that allows to simulate realistic scenarios. The simulator also includes multiple physics engines embedding different levels of realism for the motion and collisions among the agents. What makes ARGoS extremely flexible is that, during the same simulation run, the physics of different groups of robots can be handled by different physical engines [9]. A description of the architecture is given in Figure 4. In particular: the *Swarmanoid Space* is the 3D arena, implemented as a scene graph, where all the entities are simulated; the *Common Interface* is an abstraction layer that allows the simulator to be executed with hardware in the loop [9].

In this paper, each autonomous agent is equipped with: a range and bearing system, comprising 12 RGB LEDs (to send the message) and IR range and bearing sensor (to receive the message in ling of sight); wheel actuators; IR proximity sensor, for obstacle avoidance; gripper actuator, for collecting the detected items.

A. ARGoS simulation

The implementation of the foraging algorithms in the ARGoS simulation has been built on the loop functions and

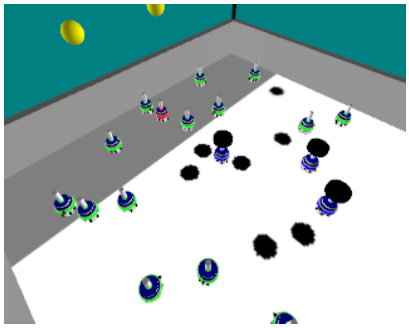


Figure 5. Representation of the foraging arena in the ARGoS simulator through OpenGL.

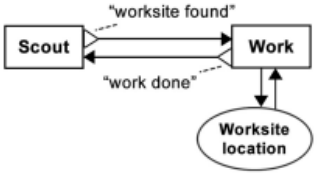


Figure 6. Graphical BDRML representation of the *solitary foraging* robot control algorithm, derived from the individualist design pattern [12].

on the OpenGL graphical visualisation through the Qt user functions. The arena is divided in two areas: a grey area that serves as nest (i.e. homing area), where the robots are initially deployed; a white area where resource items are scattered [9] (see Figure 5). The task of the autonomous robots is to leave the nest, search for target items, grab them and bring them back to the nest. Resource items are represented as black spots on the ground, which are collected with the gripper actuator when the robot goes over it (hence, the agent switches from *Scout* to *Worker*). When a robot is transporting an item and goes back to the nest, a cylinder is represented on top of it (see Figure 5). Each robot can transport only one item per time, as customary [9]. The status of the robot (i.e., begin a *Scout* or a *Worker*) is detectable through light sensors. The selected colours are green for *Scout*, blue for *Worker* and red for *Broadcaster* (see Figure 5).

B. Implemented foraging strategies

To conduct the experiments for the respective strategies mentioned in this paper, we utilised open-source FORDYCA (FOraging Robots use DYnamic CAches) project, built on ARGoS simulator [5]. The robots are modelled as *s-bot*, developed in the Swarm-bots project [3]. In the reported experiments, we make the following assumptions: the robots are homogeneous, have an unlimited battery supply, and are able to communicate directly through range and bearing sensors; robots are randomly distributed in the environment; the arena size is known to the robots.

1) Solitary Foraging: The *solitary* foraging finite state machine robot controller was created based on the individualist design pattern using BDRML and represented in Figure 6. The implementation involved copying the BDRML pattern

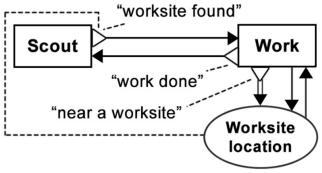


Figure 7. Graphical BDRML representation of the *behavioural matching* or *local broadcaster* foraging robot control algorithm, derived from the *Information Exchange near Worksites*, *Information Storage* and *Information Transmitter* design patterns [12].

primitives and renaming its data structure to "Work site location". In particular, the high-level representation in BDRML given in [12] has been modified for the *Scout* and *Work* behaviours. In the proposed implementation, the "Work" behaviour is represented by two states, "GO_TO_WORKSITE" and "GO_TO_BASE" in order to account for repeated motion from the nest to the resource item location if there is some leftover. The pseudocode provided for the finite state machine controller of this case is shown below.

```
if (state == SCOUT) {
   DoRandomWalk();
   if (sensingWorksite) {
      SaveWorksiteLocation();
      state = GO_TO_WORKSITE;
   }
} else if (state == GO_TO_WORKSITE) {
  TakeStepTowardsWorksite();
  if (isAtWorksite) {
    if(worksiteNotEmpty) {
      LoadResource();
      state = GO_TO_BASE;
    } else state = SCOUT;
  } else if (state == GO_TO_BASE){
    TakeStepTowardsBase();
    if (isInBase) {
       UnloadResource();
       if (worksiteWasNotEmpty) {
         state = GO_TO_WORKSITE;
       } else state = SCOUT;
     }
   }
```

2) Behavioural Matching or Local Broadcaster: In *behavioural matching*, individual foragers learn about promising destinations using the patterns followed by other foragers. A local broadcaster controller was created as a result of combining the *Broadcaster* and the *Information Exchange near Worksites* (IEW) patterns in BDRML (Figure 7). In our implementation, the robots were equipped with a wireless communication module for the Range and bearing system, having a maximum range of 1.25 m. Because of the large availability of this hardware in robotics applications, we decided to modify the *Information Transmitter* and *Information Storage* patterns, where additional devices or chemicals would have to be placed by the robots into the environment [15]. As a consequence, we implemented the *local broadcaster* controller as specified by the following pseudocode.

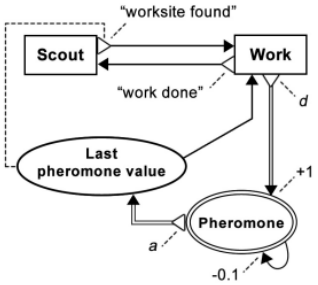


Figure 8. Graphical BDRML representation of the *stigmergical* foraging robot control algorithm, derived from the *Information Storage* and *Information Exchange Anywhere* design patterns [12].

```
if (state == SCOUT) {
   DoRandomWalk();
   if (sensingWorksite or
       sharingFoodLocationSignal) {
      SaveWorksiteLocation();
      state = GO_TO_WORKSITE;
   }
} else if (state == GO_TO_WORKSITE) {
  TakeStepTowardsWorksite();
  if (isNearWorksite) ShareLocation();
  if (isAtWorksite) {
    if(worksiteNotEmpty) {
      LoadResource();
      state = GO_TO_BASE;
    } else state = SCOUT;
  } else if (state == GO_TO_BASE){
    if (isNearWorksite) ShareLocation();
    TakeStepTowardsBase();
    if (isInBase) {
       UnloadResource();
       if (worksiteWasNotEmpty) {
         state = GO_TO_WORKSITE;
       } else state = SCOUT;
     }
   }
```

3) Stigmergical Foraging: In *stigmergy* the detection and release of pheromones on the followed trails should be implemented. The data structures in the resulting control algorithm are renamed to "Pheromone" and "Last pheromone value", as depicted in Figure 8. The pseudocode is similar to the previous reported cases and thus it is not reported in the interest of space.

4) Signalling to guide others: The control algorithm for the *signalling* foraging has four behaviours and one internal data structure common to both patterns. The BDRML primitives are renamed in order to facilitate understanding of the resulting algorithm (see Figure 9). As in the *behavioural matching* algorithm, the *Work* behaviour is represented by two states in the pseudocode implementation, i.e. "GO_TO_WORKSITE" and "GO_TO_BASE". Again, the pseudocode is not explicitly reported for space limitations.

IV. Comparative Results

In this section we will propose the comparison results among the different foraging algorithms detailed in Section III.

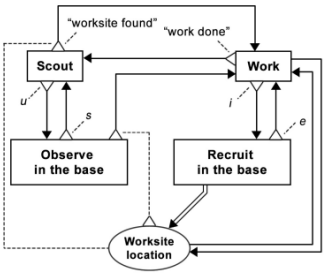


Figure 9. Graphical BDRML representation of the *signalling* foraging robot control algorithm, derived from the *Broadcaster* and *Information Exchange Centre* design patterns [12].

Since the main purpose is to offer quantitative guidelines for helping researchers and practitioners in the selection of the different approaches, we focus on the scouting efficiency of the different solutions and then we offer an analysis of the computation times necessary in each proposed simulation scenario.

The swarm scouting efficiency can be effectively measured by the time needed for the first worksite discovery and retrieval to the nest in a given experimental run since the longer it takes a swarm to discover its first worksite, the worse scouting efficiency it has. Notice that the time of the first detected location is not affected by interference between robots, while instead the last or median worksite discovery are [12]. The time needed to the first discovery (measured in hours) for the different foraging algorithms is reported in Figure 10 as a function of the distance of the worksite D (measured in metres). Since all foraging algorithms are affected by the swarm dimension (i.e. number of agents N), we report the results for $N = \{10, 25, 50\}$ agents. The negative effect of large worksite distance D is observed for all the approaches, even though the number of agents do play a role (note the different scales of the y-axis for different values of N). The scouting efficiency of *stigmergy* swarms and *signalling* to guide other agents is affected more significantly by worksite distance D than that of other foraging solutions, since recruitment requires all the agents to travel the whole way back to the base. One interesting result is that the performance difference is proportionally invariant with respect to the number of robots. This is evident for *stigmergical* and *signalling* when $D > 20$ m. Moreover, when the number of agents increases, the performance differences among the different solutions becomes less evident for short distances: for example, for $N = 10$ (Figure 10-a) the difference becomes evident for a distance $D > 10$ m, while for $N = 50$ (Figure 10-c), the worksite should be placed at a distance greater than $D = 17$ m. In all the scenarios, the *solitary* and *behavioural matching* are the most efficient and has basically the same performance almost everywhere.

A. Computation time analysis

Another parameter used for the simulative comparison is the computation time. We analyse the computation time in different cycles of foraging strategies using box and whiskers plots, depicted in Figure 11. The mean and the median of the computation times are in practice independent from the chosen algorithm when $N = 10$. Instead, when the number of robots increases, the mean value is higher with respect to the median, which denotes the presence of outliers (occasionally large computation times). *Stigmergic* foraging has steadily the highest computation times, while the *individual* and *behavioural matching* proves to be, again, the most efficient in all the considered scenarios.

V. Conclusion

The efficiency of the algorithms for swarm foraging are currently difficult to compare and analyse due to the nonlinear nature of the emergent collective behaviour [6]. The *Information-Cost-Reward* framework developed in this paper demonstrates how information flow in swarms can be formally related to the amount of reward that a swarm receives from the environment during foraging. It can be observed from the scouting efficiency that *signalling* is better than *stigmergy*, while the *individual* and *behavioural* approaches are the best for both swarm efficiency and computation times when the number of robots increases. However, if the swarm is relatively small, *signalling* is definitely the choice to make. The comparison here proposed, albeit not complete, shows some clear guidelines for the choice of the algorithm to implement. Of course, test on real robots is foreseen for the future developments. Furthermore, tests on more structured and complex foraging algorithms will be considered as well, e.g. group hunting which including coordinated foraging and cooperative foraging.

References

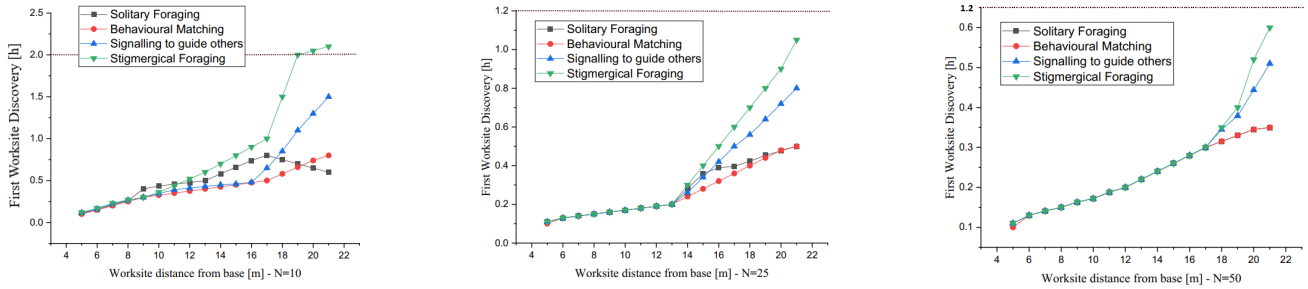
[1] M. Andreetto, M. Pacher, D. Macii, L. Palopoli, and D. Fontanelli. A Distributed Strategy for Target Tracking and Rendezvous using UAVs relying on Visual Information only. *Electronics*, (10), 2018. ISSN 2079-9292.

[2] Alexandre Campo and Marco Dorigo. Efficient multi-foraging in swarm robotics. In *European Conference on Artificial Life*, pages 696–705. Springer, 2007.

[3] Frederick Ducatelle, Gianni A Di Caro, Carlo Pinciroli, Francesco Mondada, and Luca Gambardella. Communication assisted navigation in robotic swarms: self-organization and cooperation. In *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4981–4988. IEEE, 2011.

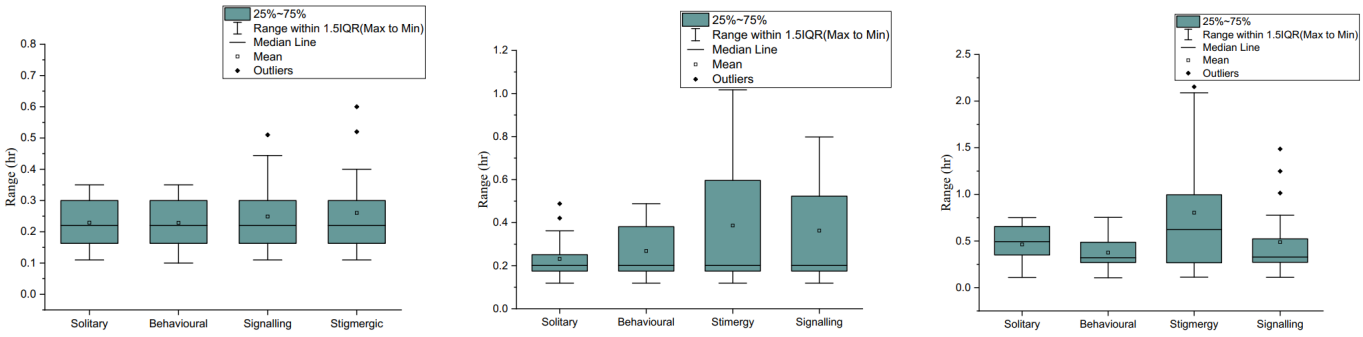
[4] D. Facinelli, M. Larcher, D. Brunelli, and D. Fontanelli. Cooperative UAVs Gas Monitoring using Distributed Consensus. In *2019 IEEE Computer Society Signature Conference on Computers, Software and Applications (COMPSAC)*, volume 1, pages 463–468, Milwaukee, Wisconsin, USA, July 2019. IEEE.

[5] John Harwell and Maria Gini. Broadening applicability of swarm-robotic foraging through constraint relaxation. In *2018 IEEE International Conference on Simulation,*



(a) (b) (c)

Figure 10. Time of the first worksite discovery in the static consumption mission using (a) $N = 10$, (b) $N = 25$ and (c) $N = 50$ robots.



(a) (b) (c)

Figure 11. Box and whiskers of the computation times (in hours) for (a) $N = 10$, (b) $N = 25$ and (c) $N = 50$ agents.

Modeling, and Programming for Autonomous Robots, (SIMPAR), pages 116–122. IEEE, 2018.

[6] Nicholas R Hoff, Amelia Sagoff, Robert J Wood, and Radhika Nagpal. Two foraging algorithms for robot swarms using only local communication. In *2010 IEEE International Conference on Robotics and Biomimetics*, pages 123–130. IEEE, 2010.

[7] Anna Font Llenas, Mohamed S Talamali, Xu Xu, James AR Marshall, and Andreagiovanni Reina. Quality-sensitive foraging by a robot swarm through virtual pheromone trails. In *International Conference on Swarm Intelligence*, pages 135–149. Springer, 2018.

[8] Fabio Pasqualetti, Joseph W Durham, and Francesco Bullo. Cooperative patrolling via weighted tours: Performance analysis and distributed algorithms. *IEEE Transactions on Robotics*, 28(5):1181–1188, 2012.

[9] Carlo Pinciroli, Vito Trianni, Rehan O'Grady, Giovanni Pini, Arne Brutschy, Manuele Brambilla, Nithin Mathews, Eliseo Ferrante, Gianni Di Caro, Frederick Ducatelle, et al. Argos: a modular, parallel, multi-engine simulator for multi-robot systems. *Swarm intelligence*, 6(4):271–295, 2012.

[10] Lenka Pitonakova. Foraging Strategies in Nature and Their Application to Swarm Robotics. *SwarmIntell-Springer*, page 33–63, 2013.

[11] Lenka Pitonakova, Richard Crowder, and Seth Bullock. Information flow principles for plasticity in foraging robot swarms. *Swarm Intelligence*, 10(1):33–63, 2016.

[12] Lenka Pitonakova, Richard Crowder, and Seth Bullock. Information exchange design patterns for robot swarm foraging and their application in robot control algorithms. *Frontiers in Robotics and AI*, 5:47, 2018.

[13] James F Roberts, Timothy S Stirling, Jean-Christophe Zufferey, and Dario Floreano. 2.5 d infrared range and bearing system for collective robotics. In *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3659–3664. IEEE, 2009.

[14] G. Seminara and D. Fontanelli. First Responders Robotic Network for Disaster Management. In Jan Mazal, editor, *Modelling & Simulation for Autonomous Systems (MESAS)*, pages 350–373, Cham, 2018. Springer International Publishing. ISBN 978-3-319-76072-8.

[15] Ouarda Zedadra, Nicolas Jouandeau, Hamid Seridi, and Giancarlo Fortino. Multi-agent foraging: state-of-the-art and research challenges. *Complex Adaptive Systems Modeling*, 5(1):3, 2017.