Facilitating Real-time Collaboration and Learning in Search Environments for Multi-Robot Systems via Real-time Evolutionary Algorithm

Angel Sylvester
University of Minnesota
Twin Cities, United States
sylve057@umn.edu

Maria Gini
University of Minnesota
Twin Cities, United States
gini@umn.edu

ABSTRACT

This work draws inspiration from social insects exploring an unfamiliar environment to develop general search strategies. It specifically draws from discoveries in evolutionary and behavioral adaptation for foraging/exploration in ants. The solution seeks to encourage real-time adaptations that aid in search efficiency, removing the computationally intensive pre-training necessary in many learning architectures. Inter-agent communication is exploited to continuously adopt the parameter values that best complement the current exploration strategy in each robot. In contrast to the traditional genetic algorithm architecture, this form of unsupervised, heuristic-based online learning strives to optimize the controller parameters exclusively via local interactions, which are intended to offer fine-grain control to the low -level thresholds and constants employed for the exploration strategies expressed while reconciling the system's implicit (environment familiarity) and explicit goals (object acquisition).

KEYWORDS

Genetic algorithm, unsupervised learning, dynamic learning, swarm robotics, multi-agent systems, online learning

ACM Reference Format:
Angel Sylvester and Maria Gini. 2023. Facilitating Real-time Collaboration and Learning in Search Environments for Multi-Robot Systems via Real-time Evolutionary Algorithm. In *Proc. of the 22nd International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2023), London, United Kingdom, May 29 – June 2, 2023*, IFAAMAS, 10 pages.

1 INTRODUCTION

Swarm robotics is a field of robotics that derives insight from biological entities such as bees, beetles and foraging ants. At the agent level, the internal mechanisms are primarily primitive and based on current interactions with the environment. However, these behaviors fundamentally contribute to the emergence of complex macroscopic events. For instance, the behaviors can enable the synchronization of agents in some formations or in the swift acquisition of objects from the environment (i.e., foraging). These individual behaviors are embodied within the controller algorithm of each agent but the mechanisms may be based on heuristics or optimization methods inspired by real organism behavior (i.e., particle swarm optimization, ant colony optimization).

Proc. of the 22nd International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2023), A. Ricci, W. Yeoh, N. Agmon, B. An (eds.), May 29 – June 2, 2023, London, United Kingdom. © 2023 International Foundation for Autonomous Agents and Multiagent Systems (www.ifaamas.org). All rights reserved.

These algorithms are fine-tuned to the current environment, with no direct focus on creating generalizable search strategies that could be applied to similar situations. This work, however, strives to adopt the strategies utilized by biological organisms to develop dynamically changing behaviors in response to incentives and punishments in a 'food' and possibly obstacle-laden environment. Besides the strategies expressed, we also incorporate an evolutionary algorithm with parameters intended to guide how the strategies themselves occur. It has been shown that animals learn via Bayesian updating, which when applied to animal behavior, is based on the assumption that animals have an expectation of the world (i.e., patch quality, quality of mates) or "posterior opinion" which gets updated as new information is encountered [19]. Coupled with this intuition, we modify the controller parameters in each agent using its real-time performance. This is especially applicable when an agent needs to instantaneously update itself in response to unforeseen events or elements in the environment. The purpose of this approach is to adopt the framework of genetic algorithms to support real-time (and thus less training-intensive) adaptation to the environment.

2 RELEVANT WORK

Many solutions for improving aspects of emergent behaviors found in foraging-based scenarios draw inspiration from social or biological phenomena or via unsupervised learning models such as reinforcement learning [20]. However, the computational burden caused by memory-intensive learning frameworks or lack of generality across similar scenarios merit further exploration. In order to avoid these shortcomings, insights from heuristics applied in social insects are utilized [16, 18, 20, 25]. Algorithmic approaches inspired by social insects include the artificial bee colony algorithm (ABC) [9], artificial bacterial foraging [12], ant colony optimization [4], and particle swarm optimization [8, 16, 25], which have been used for optimization problems. This work addresses the use of genetic algorithms in online (embodied) evolution [11] and biology-inspired techniques [19, 31] to examine real-time learning for foraging tasks where prior knowledge of the environment is not available.

A genetic algorithm (GA) is a stochastic search technique intended to mimic natural evolution [15]. The potential solutions to a given problem are represented as chromosomes (mimicking genes in natural organisms) that are updated after each generation to improve the solutions as the problem space is explored more thoroughly. The solutions generated are ranked by a fitness function that retains the most promising solutions, which later undergo crossover and mutation to diversify the solution set. This process is repeated until an optimal or near-optimal solution is identified. The

process of finding a solution happens before assessing it. Online implementations, on the other hand, do this process in real-time as progress is made over time. Since the optimal search problem is NP-hard [27], computational methods such as GAs are a potential way to produce high-quality solutions with reduced time complexity.

GAs have been used in path planning, task coordination, and multi-objective control [29] In multi-robot settings, this approach allows for alternations in controller parameters [24], levels of interactivity with other robots [30], or collision avoidance [35] via the genetic encoding of these settings that are updated over many iterations. The genetic material is generally represented as a vector that is transformed via crossover or mutation to develop new solutions to be assessed [10]. However, our work specifically differentiates itself from others through its increased focus on existing biological exploration and learning strategies (i.e., using bayesian updating [31], optimal foraging theory [19]) as an alternative to explicit forms of long-term memory as applied to the foraging scenario. The intent is abstraction of the general environment in real-time rather than parameter fine-tuning for specific environmental configurations. This would be relevant in cases where complete visibility or prior knowledge of the environment itself is unavailable.

3 ONLINE PROGRESSIVE IMPROVEMENT GENETIC ALGORITHM (OPI-GA)

We propose a solution to the foraging problem based on making progressive strategic improvements through real-time contextual information and inter-robot interactions using a genetic algorithm (GA). GAs use a fitness function to drive the evolution of certain parameters toward an optimal solution for that objective. This is initially done in a stochastic fashion until convergence towards an optimal configuration. Many generations are needed to thoroughly explore the problem space. However, the real world may be dynamically changing, so an optimal solution is not always attainable. The intuition underlying our online GA is progressive improvement rather than optimality. This is done by allocating a certain amount of time to exploration (assigning agent parameters) and comparing that outcome to past results, to determine whether the current approach should be enforced or penalized. Specifically, we propose a solution that combines strategic behavior learning (behavioral component) with a GA dedicated to fine-tuning how those behaviors are expressed (GA component).

3.1 Genetic algorithm overview

The behavioral component includes a probability distribution of the exploration strategy (i.e., forward persistence, circular persistence, correlated random). The hard parameters for fine-tuning those behavioral strategies are represented as a chromosome $\theta^t_{P_i}$ with each component i associated with each hard parameter's value (which is initially stochastically generated).

The parameters we seek to fine-tune via the algorithm are listed in Table 1. The *speed* parameter was selected since it is analogous to the adapted morphology (longer legs = more straight line behavior, shorter legs = tighter turn radius) found in ants to optimize search behavior depending on the environmental terrain [23].

The *reward and penalty* are derived from optimal foraging theory, which predicts behavioral strategies that maximize net energy gain (from food collected) at the lowest penalty [26], and uses constraints on temporal, energetic, and cognitive aspects. We aim to explicitly account for the temporal and energetic component involved in determining this cost-benefit relationship through the energetic reward and initial energy of the agent as well as the penalty for exploration. Also, relative abundance (the cognitive factor) is indirectly accounted for by the agent's reaction in response to a lack of success in a local area and increased straight-line (forward-persistence) exploration when the energy of the agent reaches zero.

Table 1: Parameters in a chromosome

Parameter	*Range*
speed (revolutions/sec)	[0, 10]
reward	[0, 30]
exploration penalty (per sec)	[0,5]
object encounter threshold	[0,5]

The final component, *the object encounter threshold*, aims to integrate a naive Bayes classifier-like element when creating the discrete probability distribution for the available strategic exploration options: CRW (correlated random walk), circular, forward-persistent. For instance, after N collections, the discrete probability distribution for each strategy is altered to depend only on the number of successes for each strategy divided by the sum of the number of successes encountered. Bayes theorem calculates the conditional probability of the occurrence of an event based on prior knowledge. This threshold will determine how many successes are necessary before the discrete probability distribution is only dependent on its own personal experiences with each strategy. The benefit of this approach is its scalability, low training data requirements, and lack of sensitivity to potentially irrelevant features [1].

If a robot encounters another robot, they both exchange information regarding their chromosome and current fitness, using an LED and light sensor on each robot. Interactions between robots are used to determine the subsequent chromosome an agent would assume after a generation had passed. The chosen mate for this process is based on the best-encountered agent during that generation, not the global best. The best is determined by choosing the chromosome with the highest performance, using the following fitness equation:

$$F_{\theta^t_{P_i}} = \begin{cases} \alpha_1(c_g) + \alpha_2(\frac{1}{e_g}), & if e_g != 0 \\ \alpha_1(c_g), & otherwise. \end{cases}$$

where α_1 and α_2 are weight constants of 2 and 1 respectively (to encourage object collection over collision avoidance), while c_g is the total collected during that generation, and e_g is the total number of collisions encountered during that generation. If no fitter robots are encountered, the chosen mate is its own chromosome.

If the encountered agent has a higher fitness, the original chromosome for that agent is saved until the beginning of the next generation. Once that generation begins, that saved chromosome and current chromosome undergo crossover and mutation to create a new chromosome (that is randomly selected from the children) for that agent for that generation. If no agents are encountered

during that generation's duration, the chromosome does crossover and mutation with its original chromosome. The pseudo-code for the GA component can be found in Algorithm 1.

Algorithm 1 GA component pseudo-code

procedure OPI-GA(N) ▷ environment initialization
  Generate N robots
  Generate block distribution
  while $g < G$ **do**
    if *Generation* $g = 0$ **then** ▷ chromosome initialization
      for *Individual i in N* **do**
        Randomly generate chromosome $\theta^t_{P_i} \sim U(t_{min}, t_{max})$
    else
      for *Individual i in N-1* **do**
        Select Parent Index k
        if encounter **then**
          Select Parent Index k
          Select counterpart Index l and evaluate fitness
          Store chromosome if $F(F^t_l) > F(F^t_k)$
    Evaluate fitness F^t_i using $F(\theta^t_{P_i})$
    Select elite individual P^g_{elite}
    Cross-over and mutation P^g_{elite} with P^t_i
    g++

The process of crossover uses a binary representation of each parameter. It assumes parameter independence. Meaning, each parameter is represented as a binary string. The strings for each parameter may have different lengths but are the same for the same corresponding parameter. Cross-over assumes parameter independence such that each parameter pair has a different single point before being recombined into a new chromosome. The 'first' child from each recombination is the one chosen, this mirrors zygote formation in biology [17]. Also, practically speaking, exhaustive comparisons of each potential child generated would not be feasible using the existing architecture with this emphasis on real-time, dynamic changes (the robot gets only one chromosome). The overall intent is to ensure that beneficial trait parameters persist in general through this process. A potential future direction could include using forms of memory or association learning to develop ways to abstract potential fitness so all children can be considered in this real time scenario. This process is repeated for each parameter and combined to generate a new child chromosome for the subsequent generation. The process of mutation includes a scan of each binary component of a parameter, with a 0.2 probability of that element swapping (i.e., 0 to 1).

3.2 Behavioral Adaptation Component

In contrast, the exploration behaviors are updated instantly in response to events impacting individual agent performance, as described in Algorithm 2.

The system is initially set as an equally partitioned discrete probability mass function $\mathbb{P}(A)$ to sample a search strategy B_s from.

Algorithm 2 Behavior component pseudo-code

procedure OPI-GA(N) ▷ environment initialization
  Generate N robots
  Generate block distribution
  while $g < G$ **do**
    Initialize *E(0)* = reward, *penalty*, speed, and encounter threshold from $\theta^t_{P_i}$
    Initialize c = number successful collections for each respective strategy
    if *Generation* $g = 0$ **then**
      Sample $B_s \sim \mathbb{P}(A)$
    else
      for *t in g* **do**
        if *E(t)* = 0 **then**
          $\mathbb{P}(A) = \| \mathbb{P}(A) \text{ x } X_{B,1} \|$, where $X_i = 1.2$ if $A_{1,i}$ = "forward-persistent", $X_i = 1.2$, else $X_i = 0$
        if obj found **then**
          $E(t) = E(t) + E(0)$, $C(B_s) = C(B_s) + 1$
          if num obj found < threshold **then**
            $\mathbb{P}(A) = \| \mathbb{P}(A) \text{ x } X_{B,1} \|$, where $x_i = 1.2$ if $A_{1,i} = B_s$, $X_i = 1.2$, else $X_i = 0$
            resample $B_s \sim \mathbb{P}(A)$, reset *E(t)* to *E(0)*
          else ▷ naive bayes intuition
            $\mathbb{P}(A) = \| c \|$ and resample $B_s \sim \mathbb{P}(A)$
        $E(t) = E(t)$ - *penalty*
    g++

Throughout the simulation, however, the weight of each strategy is updated based on the current performance of the agent using that strategy. This weighting alters the probability function the agent samples from to determine its subsequent exploration strategy (i.e., forward persistence, circular persistence, correlated random) during that generation in response to success or failure. The scaling constant of 1.2 is intended to directly increase the probability of selecting that search strategy by 20% of what it was originally. This either occurs when the robot should enforce larger movement away from a current spot using the *forward-persistence* strategy or reinforce its current strategy. Forward persistence means that over the course of the strategy time's duration, the robot will proceed in the forward direction The overall probability distribution gets re-normalized after this manipulation.

On the other hand, *circular persistence* partitions the strategy time into four bearings, the subsequent bearing due right or left of the current heading. Finally, the *correlated random* is simply the usage of a correlated random walk, a movement that follows a Markov chain (with forward direction being most likely) and other headings equally likely with respect to each other.

Over the course of the entire trial, relevant robot status information includes the energy the agent has at that point in time (which gets updated after an object is found or deducted for every second spent exploring with no success), the current fitness, the number of objects collected so far, the strategy currently being used, whether the robot is homing or searching, and what robot's chromosome is best of those encountered so far. These values are represented

under the robot status. Examples of personal events include when the agent's energy reaches 0 or a successful collection occurs.

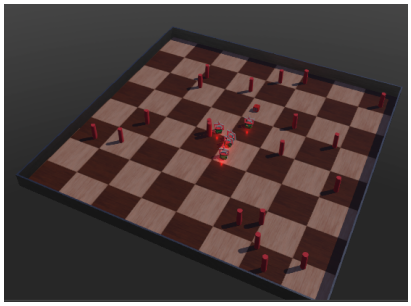
Instead of relying on centralized algorithms to determine an agent's subsequent destination (Particle Swarm Optimization) [6] or post-hoc fine-tuned parameters [28], we use a probability distribution of general strategies based on purely personal and local interactions. The behavioral probability distribution is accompanied by hard parameters such as speed, reward, penalty, and encounter threshold, which are listed in Table 1. These parameters come from insights derived from ant literature, optimal foraging theory, and elements of naive Bayes updating. Ants are an especially good analogy because of their success in foraging performance [8], which have real-world relevance (i.e., in search-and-rescue operations). It has been shown that the environment influences how a system chooses search strategies, prioritizes objectives, and coordinates local decisions [7], which makes real-time updating reasonable.

3.3 Determining the optimal learning time for a given environment

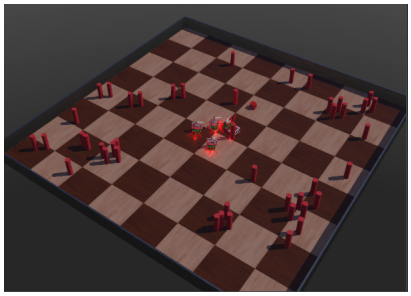
In order to determine an optimal learning time (i.e., generation duration) for each of the environment types, we assessed each for 30 trials (each trial 600 sec long) for each potential generation time. After the learning time t has elapsed, the robot controller does crossover and mutation to generate a new set of parameters as it continues the foraging task. Based on the environment, the level of success (number of objects collected, number of collisions occurred) during that duration is used to determine whether the robot will do crossover/mutation with the chromosome of another robot encountered, if it has a better fitness, or its own chromosome (reinforcing hyper-parameters already in place).

The environments we use include a 2 m × 2 m arena with objects distributed in either a random, (RN) power-law (PL), or urban-inspired (Urban) fashion. These environments were selected to simulate natural distributions (i.e., random individual seed distribution [14], food patches [19], static environment with obstacles [33]). The environment during each trial is re-generated to avoid randomly generating an environment that has a more favorable object distribution, especially given the small number of trials used to determine the learning time. During the exploration task, the learning time influences the frequency of update as each robot fine-tunes its hyper-parameters and develops a general strategy to traverse an unknown area. Ultimately the best average collection was selected (or equivalent based on p value with respect to best and selected learning time).

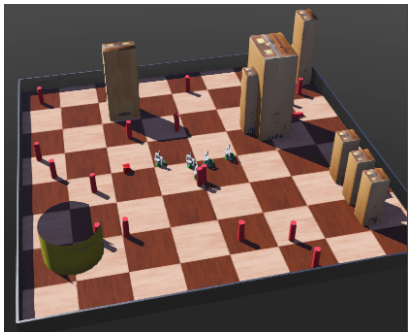
To determine the learning time for *the random distribution (RN)* environment, we counted the mean number of the objects collected. The learning time that was ultimately selected was 20 seconds for each generation because it had the largest mean of objects collected. The times attempted can be found in Table 2. An example of the distribution is in Fig. 1a. Based on the distribution of p-values done via two-sample statistical t-test, the performance for learning times of 30, 50, 90, and 110 are very similar (greater than 50 percent chance of null hypothesis being true). The mean number of objects collected decreases, being potentially indicative of a sub-optimal learning time for an environment with randomly distributed objects. The high degree of variability may also be indicative of lack of



(a) RN Scenario



(b) PL Scenario



(c) Urban Scenario

Figure 1: Snapshot of environments assessed

significance of a precise learning time per se, given how close the learning times are with respect to each other.

For the *power-law distribution (PL)*, we generated repeatedly the environment (with the same object distribution) of 20 clusters randomly distributed (each cluster ranging between 1 and 5). An example of the distribution can be found in Fig. 1b. The learning time selected was 30 sec, chosen from the best-performing candidate learning times. Based on the distribution of p-values done via two-sample statistical t-test, the other candidate solutions that are very similar (greater than 50 percent of null hypothesis being true) include 20, 50, and 60. Besides the 100 learning time, as the learning time increases, the average performance begins to decrease, with the difference growing more significant as a result.

The final environment assessed was *the urban environment*, which consists of buildings (static obstacles) of varying dimensions spread throughout the environment. This environment is of particular interest because it reflects the reality in human-centered environments. The manner in which the objects are collected is similar

Table 2: OPI-GA mean learning time for RN (random distribution of 20 objects over 30 trials)

Potential Time	*Mean (RN)*	*Stdev (RN)*	*p-val*
20	1.90	1.348	-
30	1.73	1.172	0.611
40	1.17	1.117	0.025
50	1.67	1.516	0.531
60	1.53	1.167	0.265
70	0.97	0.999	0.004
80	1.43	0.971	0.130
90	2.10	1.626	0.606
100	1.37	1.129	0.102
110	2.07	1.388	0.639

Table 3: OPI-GA learning time for PL (20 clusters distributed in the environment, over 30 trials)

Potential Time	*Mean (PL)*	*Stdev (PL)*	*p-val*
20	2.40	1.567	0.871
30	2.33	1.605	-
40	1.97	1.586	0.377
50	2.06	1.530	0.512
60	2.17	1.555	0.684
70	1.83	1.206	0.178
80	1.63	2.076	0.150
90	1.20	1.126	0.003
100	2.00	1.912	0.468
110	1.47	1.383	0.029

Table 4: OPI-GA learning time for Urban (distribution similar to RN, but with obstacles, over 30 trials)

Potential Time	*Mean (Urban)*	*Stdev (Urban)*	*p-val*
20	1.60	1.354	0.402
30	1.13	1.042	0.019
40	1.60	1.453	0.418
50	1.90	1.398	-
60	1.57	1.357	0.353
70	1.87	1.279	0.924
80	1.40	1.329	0.161
90	1.23	1.073	0.043
100	1.10	0.803	0.009
110	1.17	1.177	0.032

to the RN scenario. An example of the distribution can be found in Fig. 1c. Similar to the previous environmental distributions, the placement of each element is consistent across different trials and candidate learning times. The learning time selected was 50 seconds because it has the highest mean performance. Based on the p-values generated from the two-sample t-test, another similar candidate solution was 70. However, besides this learning time, the p-value appears to decrease, indicative of an increasing significance in the difference between the performance of these learning times with respect to the selected learning time. Furthermore, the number of objects collected on average for the same environmental conditions appear to roughly decrease at the potential time increases if you exclude the 70 candidate solution as well.

3.4 Insights on chromosome composition

The chromosome composition of each chromosome draws inspiration from optimal foraging theory, bayesian updating, and general morphological traits that influence social insect performance in different environmental conditions.

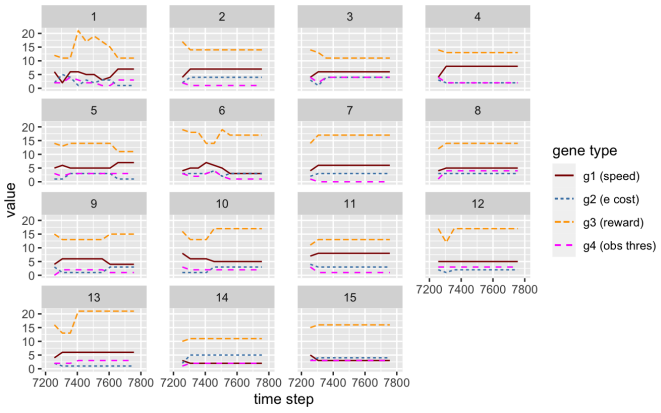


Figure 2: Snapshot of individual performance for urban environment trial (population size 15, number collected = 6)

In general, successive successes reinforce behaviors that improve performance. Furthermore, it is apparent that in contrast to the chromosomes that do not change, dynamic changes in the parameters as time progresses contribute to enhanced performance.

For instance, examining agent 5 and 9's (in Fig. 2) changes in the controller parameters contribute to the improved number of objects collected. This aligns with our understanding of the flexibility represented in ants as they make moment-to-moment updates to adapt to their every-changing surroundings [13]. Furthermore, it appears that some parameter types such as g1 (speed) and g3 (reward) are especially susceptible to increases when the fitness increases. However, as simulation progresses, the degree of observed variability in each of the parameters flattens towards generally the same value, which could be indicative of increased homogeneity.

4 RESULTS AND DISCUSSION

4.1 Experimental Set-up

As mentioned before, these parameter attributes are intended to guide the emergence of general search strategies for a given object distribution within a shared environment. In order to assess the efficacy of this approach, we used the Webots simulation software [32] with the parameters listed in Table 5.

Each robot was equipped with three distance sensors in the front for obstacle detection, a camera for object recognition, motors for navigation, and LED lights/light sensors for inter-robot recognition. The precise parameters used for the GA are listed in Table 6.

The population range was selected to assess the performance of this approach using a variety of different population densities. The other constants were fitted post-hoc and generalized to all scenarios

Table 5: Simulation Parameters in Webots

Parameter	*Constant*
Arena Size	2m *times* 2m
Robot type	Khepera
Maximum Speed	0.813 m/s
Robot Radius	0.275 mm
Turning Radius	0 90 180 270
Proximity Sensor Detection Range	0.1 m
Collision Distance	0.033 mm

Table 6: GA Parameters

Parameter	*Constant*
Time step	32 ms
Robot strategy duration	25.6 sec
Number of Trials	50
Trial Duration	600 sec
Population Sizes	5 10 15

after preliminary work determined reasonable time constraints to collect an object and gain information from the environment and other robots. The turning radius represents the available headings a robot can turn to, the time step represents the duration for each forward movement, and the robot strategy duration represents the duration for each strategy selected. If no object is found during this time, the probability of that strategy being selected decreases by 0.02, and the distribution is re-normalized. If success is observed, the probability increases by 0.02 and is re-normalized.

Once the learning time was approximated for each environment, we compared our algorithm with other bio-inspired algorithms on the same task. The algorithms included Correlated Random Walk [21], Levy [2], LAS [34], and SCAAM [18] algorithms which have demonstrated potential in foraging tasks without requiring memory-intensive computations [3, 5, 22].

The CRW and Levy algorithms are memoryless and are heuristics that determine subsequent robot orientation or step size. CRW tends to maintain the current heading but can also move in any other heading as well. The Levy walk is a random walk that samples from the Levy distribution to determine the size of a given robot's step before moving in that given direction. LAS and SCAAM, on the other hand, incorporate a learning automata-based strategy influenced by prior successes in the environment. The area is broken into cells and the LAS algorithm uses a probability distribution that gets updated with more successes to reinforce robot returns to those corresponding cells. The SCAAM algorithm uses an inverted ant-pheromone approach, enforcing areas that have not been traversed by agents yet.

4.2 Object collection performance

Once we found an acceptable learning time, we compared the performance of each algorithm to OPI-GA in three different environments (RN, PL, and urban) and different population sizes (5, 10, 15).

For the *RN scenario* (Fig. 3), 20 objects are randomly strewn throughout the environment. The OPI-GA algorithm performs the worst at the lowest population size but significantly improves as the population size (and thus perhaps the number of potential interactions) increase. With increased interactions, it is possible that agents with particularly high fitness values are better able to disseminate information via crossover and mutation regarding parameters that are conducive to exploration. The algorithm that performs particularly well in this scenario is the Levy flight. This may be due to the heterogeneous dissemination of objects which allows this algorithm to benefit due to the variable step sizes from the Levy long-tail distribution.

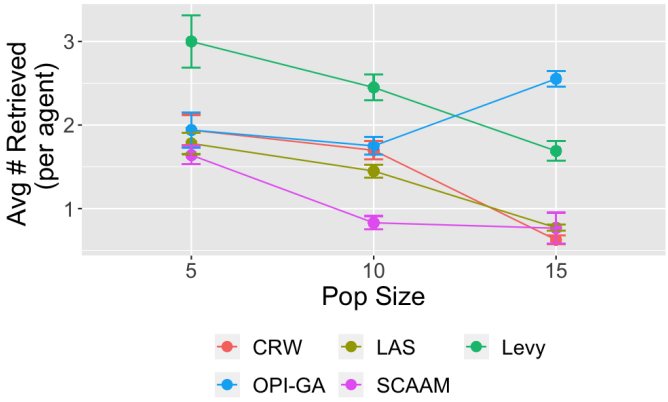


Figure 3: RN collection statistics, averaged over 50 trials

For the *PL scenario* (Fig. 4), the OPI-GA performance is almost precisely like the LAS algorithm. LAS uses an automata-based framework that reinforces local areas that have been successful in the past (there is a higher probability of returning to that location) [34]. This would make sense since there is a clustering of collectable items. The fact that OPI-GA behaves very similarly would imply the emergence of strategic behaviors, especially as the population sizes increase. Deeper analysis on the number of interactions encountered as well as their nature (i.e., locating an agent with a higher fitness value) would be necessary to further characterize these interactions and understand to what extent they can be beneficial vs impeding further productivity as the population size increases.

For the *Urban scenario* (Fig. 5), similarly to the RN scenario, 20 objects are randomly strewn throughout the environment. Additional static obstacles are included to mimic an urban environments via the inclusion of heterogeneous 'building' objects disseminated in a similar fashion to a real city. As the population increases, the performance of OPI-GA also improves with respect to the other algorithms compared. This could be due to the increased number of collisions, especially for the SCAAM and LAS algorithms as the pheromone based flocking behavior may contribute to congestion.

4.3 Generalization performance

In order to determine the extent to which the evolved strategies in OPI-GA can be generalized to new environments (same number of items, but distributed slightly differently), the fine-tuned parameters

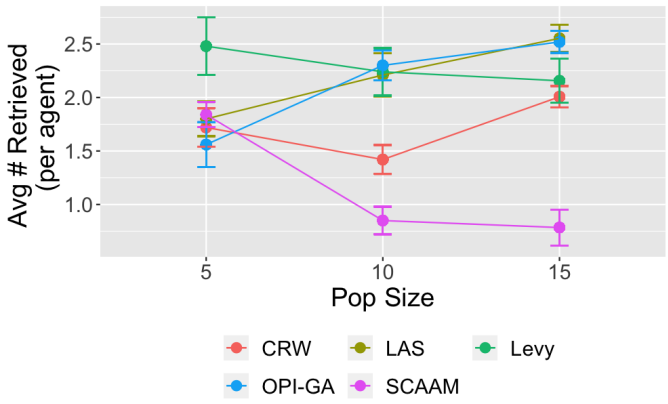


Figure 4: PL collection Statistics, averaged over 50 trials

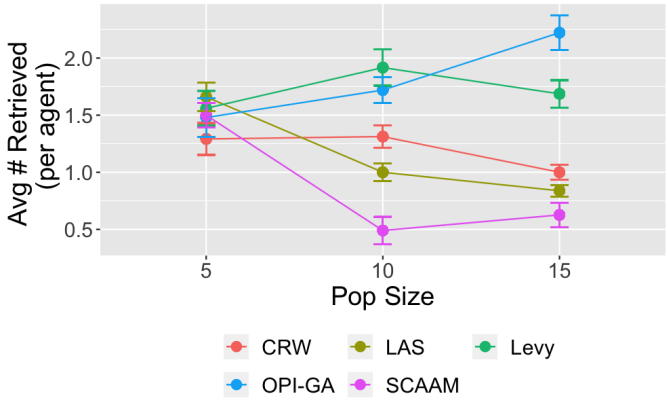


Figure 5: Urban environment collection statistics, averaged over 50 trials

from the initial environment (seed 11) were applied to the trial with the updated environment (seed 15) for 50 trials. The seed value represents the seed used to originally generate the position of the objects in the environment, to ensure that the distribution is the same across trials and algorithms. For algorithms that also utilize a form of memory or learning (i.e., LAS and SCAAM), those saved parameters were applied to the newly generated environment. Since the derived strategies are heuristics-based, we hope to see an improvement since explicit forms of memory (i.e., exact location of elements) aren't utilized in this case.

In the *RN environment* (Fig. 6), the GA algorithm performs better with respect to the other algorithms for both seed values/alternate object distributions. In general, all algorithms (even those that do not use memory) do improve their average collection performance in the second trial. This may be due to a generally more favorable distribution of objects with this particular seed. The LAS algorithm represents the best improvement between the two scenarios. The genetic algorithm, on the other hand, exhibits a slight improvement, which may mean that the fine-tuned chromosome and behavioral probability distribution from seed 11 presents only a marginal advantage in collection performance, if at all.

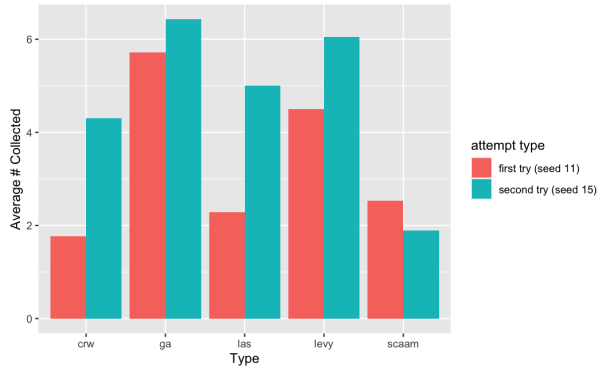


Figure 6: Overall average collection statistics in RN (population size 15) for two block distributions (seed 11 and seed 15)

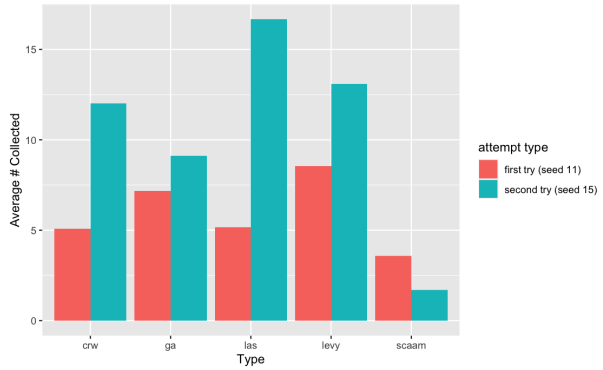


Figure 7: Overall average collection statistics in PL (population size 15) for two block distributions (seed 11 and seed 15)

As seen in the *power law environment* (Fig 7), the genetic algorithm does not perform as well across most of the other algorithms compared. This may be due to the inherent stochasticity and lack of recognition of patches in an unknown environment. Stochasticity is inherent in the behavioral strategies utilized by the agents in the genetic algorithm as it traverses over time. Repeated successes won't necessarily mean that an agent will explore the same spot, but rather, would use the same navigation strategy after successfully returning an item to the nest. For the LAS algorithm in particular, this appears to be the contrary, which makes sense this this algorithm reinforces returns to local areas with particular success.

For the scenario with 15 robots in the *urban environment* (Fig. 8), the genetic algorithm outperforms the other algorithms for seed 10 (with a randomized starting chromosome) and for seed 15 (with the fine-tuned chromosome). The fact that there is no significant decrease in performance across different environmental distributions is potentially indicative of prior learning that facilitated the emergence of fine-tuned parameters that encourage object collection for similarly distributed urban environments. A similar behavior is observable in the LAS algorithm where the previously beneficial locations were more likely to be selected OPI-GA during the seed 15 scenario as well). The general improvement across all algorithms also may indicate a preferable distribution of objects in general.

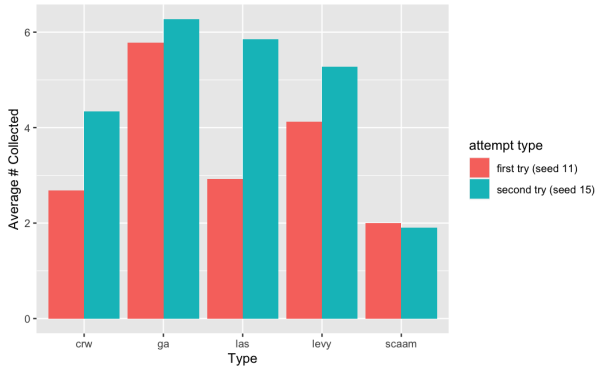


Figure 8: Overall average collection statistics in urban environment (population size 15) for two block distributions (seed 11 and seed 15)

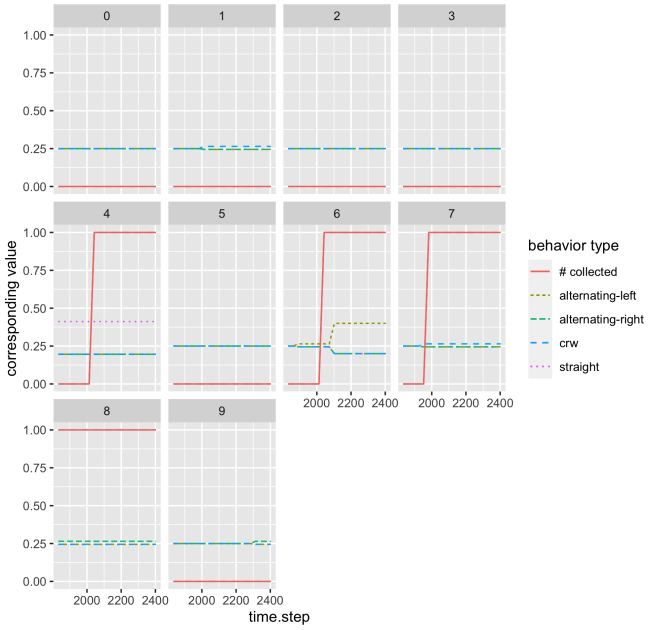
5 UNDERLYING INDIVIDUAL BEHAVIORS

We gathered information in order to better understand the transformations observed in each robot and in the environment. Based on Fig. 9, agents either have one search strategy which is chosen preferentially or an equi-partitioned distribution. The parameters also appears to stabilize towards similar corresponding values as the agents continue to explore the environment.

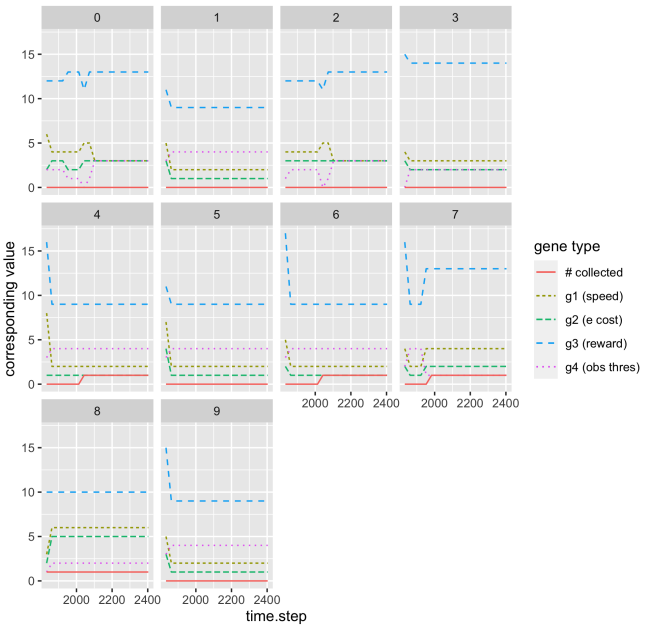
Fig. 10 (a) shows that the number of encounters of robots with a better fitness value during a given generation is highest towards the beginning of the simulation. This makes sense since all the agents begin in the center of the arena. However, the encounters remain at 0 for the majority of the time, which may imply fewer opportunities for interaction or convergence towards similar fitness values.

Fig. 10 (b), shows that the agents tended to distribute themselves in a manner where interactions were limited (towards the extremes of the arena). This would provide greater explainability of the encounter statistics observed. Despite the lack of encounters in this case, the use of distributed tactics proved fairly effective in limiting collisions between agents. Further work is needed to characterize the extent to which exclusionary vs collaborative tactics may be useful, especially in different environments.

In order to confirm the role of interaction frequency in performance, the size of the arena for the *RN scenario* was reduced by 30 percent to increase to probability of interaction. It was apparent that for certain trials that performed well, the number of interactions was markedly greater. Examples of this are referenced in Fig. 15. As the interactions between better performing robots increase, especially later on in the exploration phase, the overall performance of that respective robot improves. For the poor performing trial, the frequency of these interactions throughout the entire exploration phase is more isolated and not as consistent. It also indicates promise in the potential for inter-robot communication in propagating information that facilities effective exploration.



(a) Behavior summary

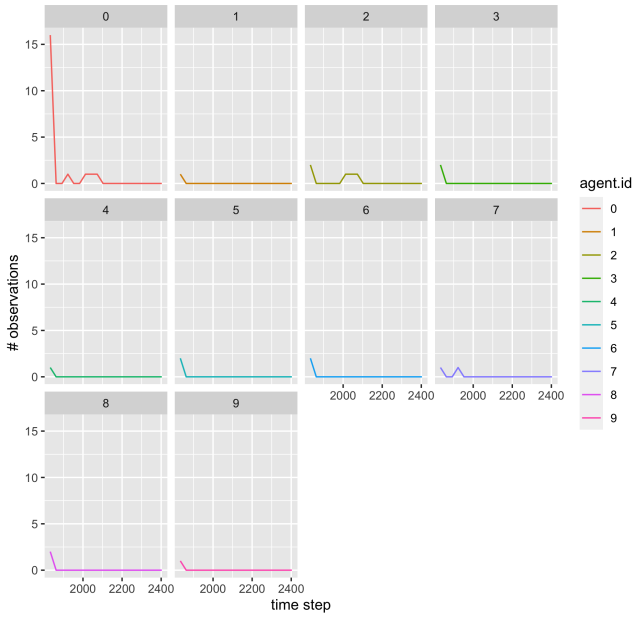


(b) Parameter progression

Figure 9: Snapshot of urban environment trial (population 10, number of blocks collected 4) showing behavioral and parameter changes

6 FUTURE WORK AND CONCLUSIONS

In general our findings indicate that our real-time genetic algorithm approach is a promising direction for abstraction-based coordination strategies across different agents in a shared environment. As the population size increases, the performance of our algorithm improves, compared to the other algorithms. Our approach was designed to garner past knowledge via changes in strategy sampling and the underlying fine-tuning parameters for those strategies,



(a) Number of more fit robots encountered

(b) Overview of each individual robot path taken, with each location recorded after the end of every generation

Figure 10: Example of navigational and social information in urban environment (population 10, number collected 4)

incorporating both morphological and behavioral shifts in a dynamically changing environment.

Further work will be done to further characterize the social interactions between agents (i.e., task allocation, hierarchy, practicing

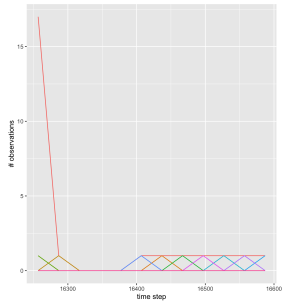


Figure 11: Encounter statistics for RN scaled-down arena (population 15) – good trial

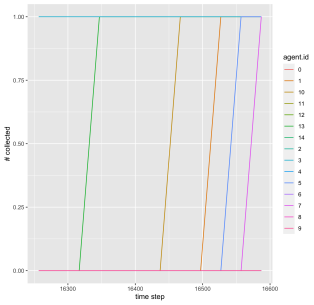


Figure 12: Collection statistics for RN scaled-down arena (population 15) – good trial

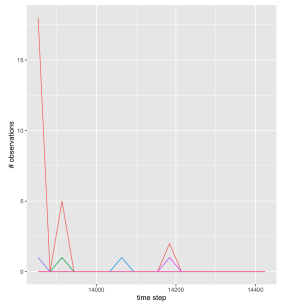


Figure 13: Encounter statistics for RN scaled-down arena (population 15) – poor trial

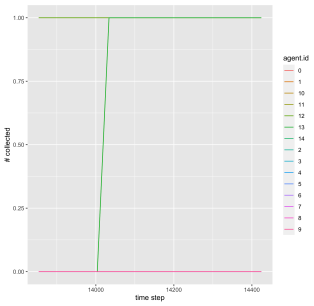


Figure 14: Encounter statistics for RN scaled-down arena (population 15) – poor trial

Figure 15: Comparison between trials with high and low encounters with better performing robots

restraint/foresight, a collective fitness vs individual fitness function, collaborative vs competition-based higher-level strategies). There will be a greater focus on the use of social dynamics to drive online learning from organisms in biology. Additionally, work will be done to explore more diverse urban environments with dynamic obstacles that tend to stay on predictable paths (e.g., cars on roads, people on sidewalks).

REFERENCES

[1] Mayank Banoula. 2023. Naive Bayes classifier - machine learning: Simplilearn. https://www.simplilearn.com/tutorials/machine-learning-tutorial/naive-bayes-classifier

[2] F. Bartumeus, J. Catalan, U. L. Fulco, M. L. Lyra, and G. M. Viswanathan. 2002. Optimizing the encounter rate in biological interactions: Lévy versus Brownian strategies. *Phys. Rev. Lett.* 88 (Feb 2002), 097901. Issue 9. https://doi.org/10.1103/PhysRevLett.88.097901

[3] Martina Dal Bello, Alfonso Pérez-Escudero, Frank C Schroeder, and Jeff Gore. 2021. Inversion of pheromone preference optimizes foraging in C. elegans. *eLife* 10 (July 2021). https://doi.org/10.7554/elife.58144

[4] E. Bonabeau, M. Dorigo, and G. Theraulaz. 2000. Inspiration for optimization from social insect behaviour. *Nature* 406, 6791 (July 2000), 39–42. https://doi.org/10.1038/35017500

[5] Arjun Chandrasekhar, James A. R. Marshall, Cortnea Austin, Saket Navlakha, and Deborah M. Gordon. 2021. Better tired than lost: Turtle ant trail networks favor coherence over short edges. *PLOS Computational Biology* 17, 10 (Oct. 2021), e1009523. https://doi.org/10.1371/journal.pcbi.1009523

[6] Cecilia Di Chio, Riccardo Poli, and Paolo Di Chio. 2006. Extending the particle swarm algorithm to model animal foraging behaviour. In *Ant Colony Optimization*

and Swarm Intelligence. Springer Berlin Heidelberg, Brussels, Belgium, 514–515. https://doi.org/10.1007/11839088_58

[7] Stefanie M. Countryman, Martin C. Stumpe, Sam P. Crow, Frederick R. Adler, Michael J. Greene, Merav Vonshak, and Deborah M. Gordon. 2015. Collective search by ants in microgravity. *Frontiers in Ecology and Evolution* 3 (March 2015). https://doi.org/10.3389/fevo.2015.00025

[8] M. Dorigo, V. Maniezzo, and A. Colorni. 1996. Ant system: optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)* 26, 1 (1996), 29–41. https://doi.org/10.1109/3477.484436

[9] Chengli Fan, , Qiang Fu, Guangzheng Long, and Qinghua Xing. 2018. Hybrid artificial bee colony algorithm with variable neighborhood search and memory mechanism. *Journal of Systems Engineering and Electronics* 29, 2 (April 2018), 405–414. https://doi.org/10.21629/jsee.2018.02.20

[10] Stephanie Forest. 1993. Principles of natural selection applied to computation. *Science* 261, 5123 (Aug. 1993), 872–878.

[11] Marco Galassi, Nicola Capodieci, Giacomo Cabri, and Letizia Leonardi. 2016. Evolutionary strategies for novelty-based online neuroevolution in swarm robotics. In *2016 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*. IEEE, Budapest, Hungary, 002026–002032. https://doi.org/10.1109/smc.2016.7844538

[12] Fei Gao, Hongrui Gao, Yibo Qi, and Qiang Yin. 2010. Bacterial foraging oriented by differential evolution strategy. In *2010 2nd International Conference on Information Engineering and Computer Science*. IEEE, Wuhan, China, 1–4. https://doi.org/10.1109/iciecs.2010.5677664

[13] Deborah M. Gordon. 2010. *Ant Encounters: Interaction Networks and Colony Behavior*. Princeton University Press, Princeton, NJ, USA. http://www.jstor.org/stable/j.ctt7rpzh

[14] Deborah M. Gordon. 2016. The Evolution of the Algorithms for Collective Behavior. *Cell Systems* 3, 6 (Dec. 2016), 514–520. https://doi.org/10.1016/j.cels.2016.10.013

[15] John H. Holland. 1975. *Adaption in Natural and Artificial Systems*. University of Michigan Press, Ann Arbor, Michigan.

[16] Hiroto Inahara and Naoki Motoi. 2021. Research on search slgorithm by PSO with virtual pheromone and dynamical niche for swarm robots. In *2021 IEEE 30th International Symposium on Industrial Electronics (ISIE)*. IEEE, Kyoto, Japan, 1–6. https://doi.org/10.1109/isie45552.2021.9576188

[17] Amy Lee and Ann A. Kiessling. 2016. Early human embryos are naturally aneuploid—can that be corrected? *Journal of Assisted Reproduction and Genetics* 34, 1 (Nov. 2016), 15–21. https://doi.org/10.1007/s10815-016-0845-7

[18] Danielli A. Lima and Gina M. B. Oliveira. 2019. Stochastic Cellular Automata Ant memory model for swarm robots performing efficiently the garbage collection task. In *2019 19th International Conference on Advanced Robotics (ICAR)*. IEEE, Belo, Horizonte, Brazil, 708–713. https://doi.org/10.1109/icar46387.2019.8981560

[19] John M. McNamara, Richard F. Green, and Ola Olsson. 2006. Bayes' theorem and its applications in animal behaviour. *Oikos* 112, 2 (Feb. 2006), 243–251. https://doi.org/10.1111/j.0030-1299.2006.14228.x

[20] Syed Irfan Ali Meerza, Moinul Islam, and Md. Mohiuddin Uzzal. 2019. Q-Learning based particle swarm optimization algorithm for optimal path planning of swarm of mobile robots. In *2019 1st International Conference on Advances in Science, Engineering and Robotics Technology (ICASERT)*. IEEE, Dhaka, Bangladesh, 1–5. https://doi.org/10.1109/icasert.2019.8934450

[21] Melanie E. Moses, Judy L. Cannon, Deborah M. Gordon, and Stephanie Forrest. 2019. Distributed adaptive search in T cells: lessons from ants. *Frontiers in Immunology* 10 (June 2019). https://doi.org/10.3389/fimmu.2019.01357

[22] Grégoire Pasquier and Christoph Grüter. 2016. Individual learning performance and exploratory activity are linked to colony foraging success in a mass-recruiting ant. *Behavioral Ecology* 27 (2016), 1702, 1709. https://doi.org/10.1093/beheco/arw079

[23] Jessica M.C. Pearce-Duvet, Coen P.H. Elemans, and Donald H. Feener. 2011. Walking the line: search behavior and foraging success in ant species. *Behavioral Ecology* 22, 3 (2011), 501–509. https://doi.org/10.1093/beheco/arr001

[24] Jim Pugh and Alcherio Martinoli. 2007. Parallel learning in heterogeneous multi-robot swarms. In *2007 IEEE Congress on Evolutionary Computation*. IEEE, Singapore, 3839–3846. https://doi.org/10.1109/cec.2007.4424971

[25] J. Pugh, A. Martinoli, and Yizhen Zhang. 2005. Particle swarm optimization for unsupervised robotic learning. In *Procs IEEE Swarm Intelligence Symposium, SIS 2005*. IEEE, Pasadena, CA, USA, 92–99. https://doi.org/10.1109/sis.2005.1501607

[26] G.H. Pyke. 2010. Optimal Foraging Theory: Introduction. In *Encyclopedia of Animal Behavior*. Elsevier, Sydney, Australia, 601–603. https://doi.org/10.1016/b978-0-08-045337-8.00210-2

[27] Ioannis Rekleitis, Ai Peng New, Edward Samuel Rankin, and Howie Choset. 2008. Efficient boustrophedon multi-robot coverage: an algorithmic approach. *Annals of Mathematics and Artificial Intelligence* 52, 2-4 (2008), 109–142. https://doi.org/10.1007/s10472-009-9120-2

[28] Nicolas D. Griffiths Sanchez, Patricia A. Vargas, and Micael S. Couceiro. 2018. A Darwinian swarm robotics strategy applied to underwater exploration. In *2018 IEEE Congress on Evolutionary Computation (CEC)*. IEEE, Rio de Janeiro, Brazil, 1–6. https://doi.org/10.1109/cec.2018.8477738

[29] Pu Shi and Yujie Cui. 2010. Dynamic path planning for mobile robot based on genetic algorithm in unknown environment. In *2010 Chinese Control and Decision Conference*. IEEE, Xuzhou, China, 4325–4329. https://doi.org/10.1109/ccdc.2010.5498349

[30] Woo sung Moon, Jin Won Jang, and Kwang Ryul Baek. 2008. Evolutional interactivity in a swarm of robots. In *2008 Int'l Conf. on Control, Automation and Systems*. IEEE, Seoul, South Korea, 118–122. https://doi.org/10.1109/iccas.2008.4694535

[31] Courtney Turrin, Nicholas A. Fagan, Olga Dal Monte, and Steve W. C. Chang. 2017. Social resource foraging is guided by the principles of the Marginal Value Theorem. *Scientific Reports* 7, 1 (Sept. 2017), 11274. https://doi.org/10.1038/s41598-017-11763-3

[32] Webots. [n.d.]. http://www.cyberbotics.com. Open-source Mobile Robot Simulation Software.

[33] Hao Wei, Jon Timmis, and Rob Alexander. 2017. Evolving test environments to identify faults in swarm robotics algorithms. In *2017 IEEE Congress on Evolutionary Computation (CEC)*. IEEE, Donostia, Spain, 929–935. https://doi.org/10.1109/cec.2017.7969408

[34] JunQi Zhang, Peng Zu, and Huan Liu. 2021. Learning automata-based multi-target search strategy using swarm robotics. In *2021 11th International Conference on Information Science and Technology (ICIST)*. IEEE, Chengdu, China, 416–421. https://doi.org/10.1109/icist52614.2021.9440567

[35] Yuxin Zhao and Wei Zu. 2009. Real-time obstacle avoidance method for mobile robots based on a modified particle swarm optimization. In *2009 International Joint Conference on Computational Sciences and Optimization*. IEEE, Sanya, China, 269–272. https://doi.org/10.1109/cso.2009.196