Flexibility through Autonomous Decision-making in Robot Swarms

Wayne A. Just[1], Melanie E. Moses[1,2]
[1]Department of Computer Science, University of New Mexico
Albuquerque, New Mexico, USA
[2]Santa Fe Institute, Santa Fe, New Mexico, USA
Email: {wjust, melaniem}@cs.unm.edu

***Abstract*— For robot swarms to succeed in a complex world, the algorithms that support their behavior must be flexible enough to adjust to unknown and dynamic resource distributions. This research focuses on developing an algorithm that adjusts collective swarm foraging behavior to any resource distribution. We combine different solutions through a decision-making process that allows the individual, as opposed to the swarm, to dynamically decide which solution to use in its current environment. We evolve swarm foraging solutions to provide effective foraging for two extremes of resource distributions. These solutions, which are different parameterizations of a foraging algorithm, become the behaviors of each individual robot of the swarm. The individual robots make choices through a simple decision process as to which behavior to follow based only on locally sensed information. We show that the individual decision-making process increases the flexibility of the swarm, enabling it to outperform homogenous swarms in multiple distributions without prior knowledge and time-consuming optimizations for specific resource distributions.**

Keywords—swarm robotics, swarm intelligence, task allocation, response threshold model

I. Introduction

In nature, animals must adapt their foraging strategies as they deplete their resources or move to new locations and when natural or man-made events alter the distribution of resources. The ability to adjust foraging strategies to respond to changes in the distribution of resources is a central issue in foraging behavior [1], [2]. As robotic swarms are developed for use in real world foraging applications, they too must have the flexibility to dynamically alter behavior to forage effectively in unknown or changing resource distributions.

Robots forage by finding resources (their target) and returning with the resource to a specified destination (a nest). Thus, the effectiveness of a swarm of robots to search for resources is highly dependent on how flexibly the swarm responds to different resource distributions. However, resource distributions are heterogeneous in nature [3] and different locations may contain different distributions dependent on the complexity of their natural surroundings [4]. Furthermore, resources may be subject to dynamic activity that significantly changes the resource distribution. For example, a swarm that collects litter may be impacted by humans both contributing to and reducing litter at irregular intervals. Natural events like wind and rain might also cause changes to the distribution of litter in real time. Since a resource distribution may be unknown and may be subject to dynamic repositioning, it is critical that a swarm's foraging algorithm be flexible enough to perform effectively in any constant or dynamic resource distribution.

Biologically inspired foraging algorithms can provide flexibility by imitating actions and strategies of the animals. Ant-inspired foraging algorithms replicate pheromone use, direct communication and site fidelity (repeatedly returning to a remembered resource location) [5]-[9]. Local communication between bees in their waggle dance for recruitment [10], [11] and division of search space by honey bees [12] represent other insect foraging behaviors that respond adaptively to the resource distribution. These foraging behaviors combined with searching, grabbing and transporting food, recruitment and coordination are the fundamental components of a foraging strategy [13]. However, while these strategies may be effective in a specific distribution, they may be less effective in others. For example, laying a chemical pheromone trail between a rich patch of resource and the nest can be advantageous when resources are clustered in space and persist for long periods of time, but a pheromone strategy is ineffective given sparse and non-clustered resources [9], [14].

This work takes a different approach to creating flexible foraging strategies. Rather than considering foraging as a pre-determined strategy of the swarm for a distinct resource distribution, we consider foraging as a complex behavior that emerges from the choices of individual robots in the swarm. Each robot has two complex behaviors (or strategies) to choose from. These behaviors are evolved for effective foraging from different resource distributions. This establishes the two principle concepts of this work.

1. We evolve foraging behavior for two specific resource distributions and we combine these two different behaviors through a decision-making process to create an effective emergent foraging strategy for any resource distribution.

978-1-5386-2726-6/17/$31.00 ©2017 IEEE

2. Each robot selects which solution to use based only on its local information. The decision is independent of the actions of the rest of the swarm. This produces a dynamic distribution of solutions in use at any point in time across the swarm. Thus, the overall solution emerges from the robots' choices over time and enables the swarm to adapt to unknown and dynamic resource distributions.

This method resembles task allocation where each robot or group of robots perform different tasks. There are two methods of task allocation, orchestrated and self-organized [15]. In orchestrated task allocation, there is a plan for allocating robots to specific tasks and this plan is communicated to all robots. Self-organized task allocation emerges from decisions individual robots make from information local to the robot.

Self-organized task allocation commonly uses a response threshold model. This is a stochastic model where thresholds identify different behaviors. If a threshold is exceeded a robot performs the specified behavior. These models may be fixed (the threshold is fixed) or dynamic. A response threshold model has been shown to simulate the behavior of a worker [16], [17] in an insect swarm. Response threshold models have been used for energy management of the swarm by limiting the number of foragers at one time [18]-[20], proportioning of robots to task [21], [22] and supports coordination between simple robots with no required communication [23].

We combine the different foraging strategies with the response threshold model which produces a simple decision-making algorithm allowing individual robots to autonomously and dynamically determine which foraging strategy they will use based on their current local sensor information. We show that our Autonomous Decision-making Algorithm (ADMA) increases flexibility of the swarm which allows the swarm to maintain a high foraging performance. We find that the ADMA is statistically equivalent in most cases compared to optimal foraging strategies for each resource distribution. We also find that in a dynamic distribution, where resources do not remain in the same location throughout the experiment, the ADMA performance is also statistically equivalent to the best performance of the optimized solution.

II. Background

A. Central Place Foraging Algorithn

The Central Placed Foraging Algorithm (CPFA) was designed to mimic the resource gathering habits of seed-harvester ants [9]. It has been fully implemented and tested in the iAnt robot for different resource distributions [24]. The CPFA behavior is a probabilistic state machine with transitions between states governed by the 7 parameters described below. All robots use the same parameters and associated settings in the CPFA.

1) Probability of switching to searching (p_s): When a robot leaves the nest following a randomly selected heading (not specified by site fidelity or pheromones), the probability it switches to a search state is specified by this parameter.

2) Probability of returning to nest (p_r): A robot switches from any search state to return to nest state and heads back to the central nest.

3) Uninformed search variation (ω): A robot without information about the location of resources uses a correlated random walk (a random walk with angles correlated with the previous direction of movement). This parameter determines how correlated successive steps are and therefore how straight the random walk is.

4) Rate of informed search decay(λ_{id}): When a robot searches with information that resources have previously been found at its location it performs an informed search with more random, and therefore more thorough, local search. As this angle decays the search becomes broader which mimics the uniform search variation.

5) Rate of site fidelity(λ_{sf}): Determines the probability of using site fidelity as a function of the number of resources detected in the local environment. Site fidelity is remembering and returning to the last location visited where it found a resource.

6) Rate of laying a pheromone(λ_{lp}): Determines the probability that the robot will leave a pheromone trail from the resource location back to the nest.

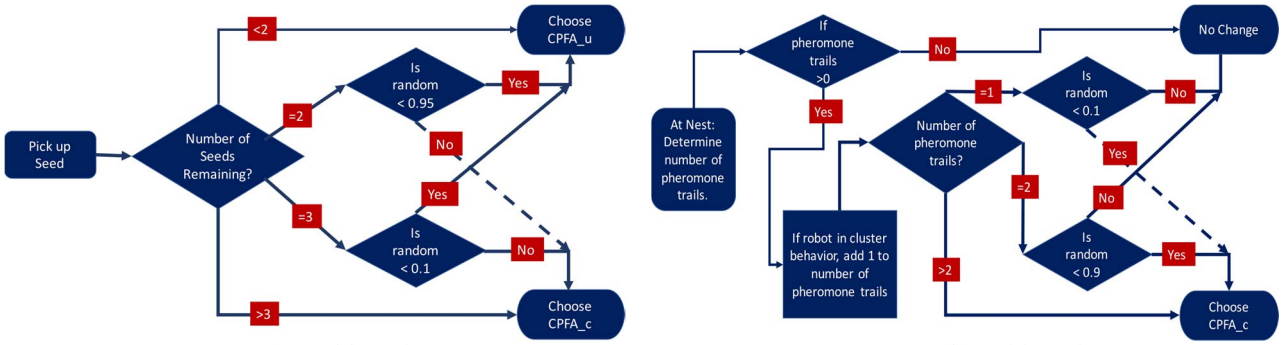
7) Rate of pheromone decay(λ_{pd}): *Determines the rate at which a pheromone trail will decay. This rate is set at the time the pheromone is placed.*

In prior work, the CPFA parameters for a particular resource distribution were evolved using a genetic algorithm (GA) [9]. Fitness is defined as the number of resources collected in a fixed period. The GA uses elitism to save the most fit parameter settings and uses a tournament selection to select parents for the next generation. Crossover and mutation are used and the process is repeated for 100 generations. The parameter settings for individuals that result in the largest number of resources collected by the swarm are then considered optimized for the distribution.

The GA is used to optimize parameter settings for specific settings of the environment: the arena size, number of initial resources, the statistical properties of the distribution of resources, and number of robots are held constant for each set of GA runs.

When a robot finds a resource, the CPFA performs a survey of the number of resources remaining within its limited sensor range (it can detect resources next to it). The robot uses this information to determine the probability of laying a pheromone or remembering the location using site fidelity.

Since the robots are autonomous and their sensors only detect objects next to them, collisions with other robots may occur. To handle these collisions, a simple collision detection algorithm is used in which robots rotate until they can move freely again. Once clear of the obstacle, they will adjust their heading towards their goal location.



(a) Decision Point 1 (b) Decision Point 2

Fig. 1. The decision point flows used by the ADMA to determine which foraging strategy to use. (a) Decision point 1 occurs when a robot picks up a resource. Depending on the resources remaining the robot selects a strategy. (b) Decision point 2 occurs when a robot returns to the nest. Depending on its current strategy and the number of pheromone trails it determines whether to switch to the CPFA_c strategy.

B. *Distributed Deterministic Spiral Algorithm*

We compare results to the distributed deterministic spiral algorithm (DDSA) [25]. This algorithm supports guarantees that it will find all resources within a specified area in the first pass provided that resources are static. Since the DDSA is guaranteed to find all resources regardless of distribution (in a noiseless environment with non-dynamic resources), it makes an ideal algorithm for comparing the flexibility of the ADMA across all resource distributions.

The DDSA identifies a unique square spiral path for each robot starting from the nest and extending to the limit of the arena size. The spiral for each robot is spaced such that robotic sensors for detecting resources will be able to find all resources as the spiral widens leaving no resources behind. The deterministic nature of the algorithm ensures that regardless of resource distribution, all resources will be found if enough time is allowed for completion of the algorithm.

C. *Autonomous Decision-making Algorithm*

1) Foraging strategies: The ADMA uses two foraging strategies in this work. Both strategies are the same optimized strategies (i.e. paramter settings of the CPFA) evolved for particular environments. Any individual robot may use either strategy at any time. Thus both strategies may be in use at the same time in different robots. No robot will use both strategies simultaneously and all robots will use either Strategy 1 or 2.

a) Strategy 1: The CPFA_u foraging strategy is developed for the uniform resource distribution. The distribution is shown in Fig 3a.

b) Strategy 2: The CPFA_c foraging strategy is developed for the clustered resource distribution. This distribution is shown in Fig 3b.

2) Decision Points: Decision points are events where the ADMA has identified that the robot needs to make a choice on which foraging strategy to use. The decision is determined by a response threshold based on the decision point and local information and the current strategy being used.

a) Decision Point 1: When a robot finds a resource, it surveys the immediate region. The remaining resources left in the region are the input to the decision flow shown in Fig 1a.

b) Decision Point 2: When the robot returns to the nest (with or without a collected resource) it checks its current strategy and the number of pheromone trails leaving the nest. Based on these two variables it may select to use the CPFA_c strategy or keep with its current strategy (which could also be the CPFA_c strategy). This decision flow is shown in Fig 1b.

c) Response Thresholds: In Fig 1, once the sensors determine the remaining number of seeds (Fig 1a) or the number of pheromone trails (Fig 1b) a random number is compared to a response threshold. If the random number is less than the threshold the robot chooses the strategy along the 'yes' path; otherwise it chooses the other strategy. These thresholds are set so that the robots will primarily pick the CPFA_u in the uniform distribution and the CPFA_c strategy in the clustered distribution. Fig 2 shows the number of times all robots chose the CPFA_u vs. CPFA_c averaged over all the runs for each of the 3 basic distributions. This percentage is compared to the expected percentage of recources not in a cluster for a given distributon. In the uniform distribution the robots chose the CPFA_u strategy 99.7% of the time which is near optimal. In

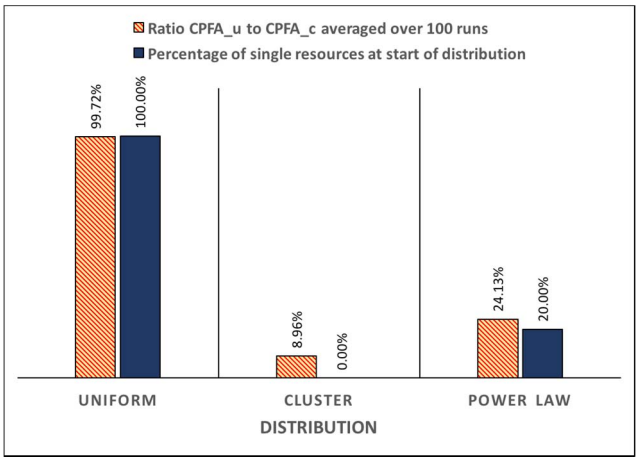
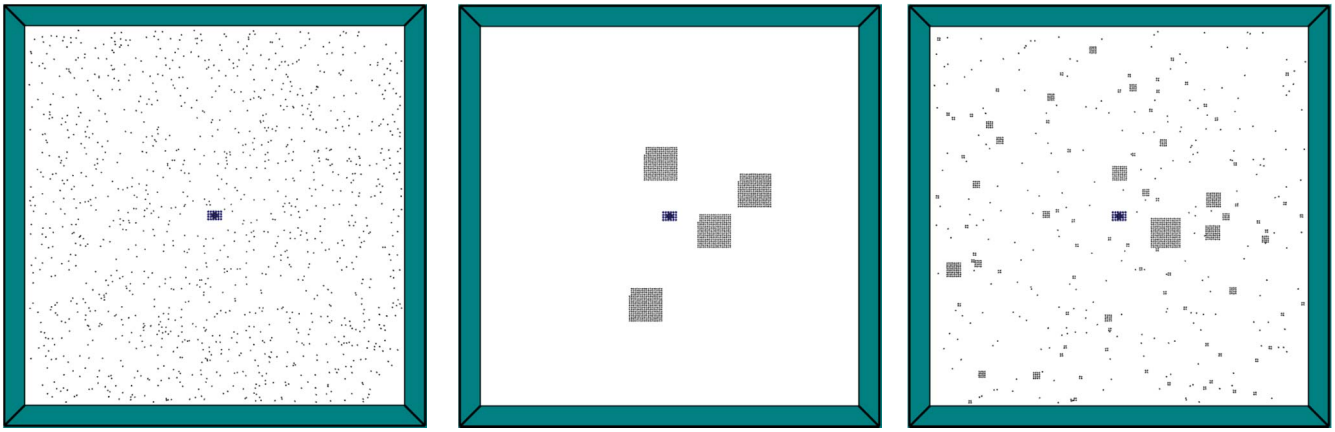


Fig. 2. Relationship between robot strategy choices and initial resource distribution. The ratio is the number of times the CPFA_u was selected by all robots dividing by the number of times the CPFA_c was selected then averaged over all runs. The percentage of single resources at start is the expected number of non-clustered resources (resources are not next to one another) when the simulation is initialized.



(a) Uniform Distribution (b) Clustered Distribution (c) Power Law Distribution

Fig. 3. Examples of the 3 base resource distributions. (a) Uniform distribution: resources are placed uniformly at random across the arena. (b) Cluster distribution: resources are clustered into 4 groups (c) Power law distribution: resources are distributed in 5 groups of varying clusters sizes (see section III B)

the clustered distribution the robots choose the CPFA_u strategy 8.96% (and the CPFA_c strategy 91.04%) of the time. The larger difference from optimum in the clustered distribution is due to resource depletion as foraging occurs (e.g. robots only sense individual resources in the local area as the seeds in the cluster become more sparse). We also compared to the power law distribution where 20% of the seeds are not clustered initially and we observe a similar performance where the CPFA_u strategy is selected 24.13% of the time.

III. Simulation and Experiment

A. Simulator

All algorithms (CPFA, DDSA and ADMA) are implemented in the ARGoS swarm simulator [26]. The simulator uses a physics engine to support robot motion and supports collision detection.

B. Resource Distributions

Algorithms to create 3 basic resource distributions and 2 combined distributions are implemented in the ARGoS simulator. All resources are randomly placed with no resources stacked on another resource. Each time ARGoS is run, resources are randomly placed so that each run is likely to have a unique placement of resources.

Each distribution has an associated optimal CPFA algorithm as determined by the CPFA GA (see section II A). Table I shows each optimal CPFA algorithm and its associated resource distribution.

TABLE I. CPFA Optimal Algorithm and Resource Distribution

Algorithm	***Distribution***
CPFA_u	Uniform Distribution (see Fig 3a)
CPFA_c	Clustered Distribution (see Fig 3b)
CPFA_p	Power Law Distribution (see Fig 3c)
CPFA_j	Jumble Distribution
CPFA_m	Mystery Distribution

1) Uniform Distribution: Fig 3a illustrates the uniform distribution. Resources are uniformly at random placed across the arena.

2) Clustered Distribution: Fig 3b illustrates the clustered distribtuion. It is made of 4 square clusters with equal numbers of resources in each cluster. Clusters are randomly positioned.

3) Power Law Distribution: Fig 3c illustrates this distribution. There is a power law distribution of pile sizes such that the placement of 1280 resources contains 1 cluster of 256 resources, 4 clusters of 64 resources, 16 clusters of 16 resources, 64 clusters of 4 resources and 256 individual resources. For other numbers of resources, the resources are divided in a similar fashion where the large cluster size is reduced to accommodate numbers of resources that do not allow all pile sizes to be powers of 2.

4) Jumble Distribution: The jumble distribution is a combination of the 3 other distributions. This distribution uniformily at random selects one of the 3 basic distribution as an initial distribution. Additional, every 20 mins of the 1 hour run, the jumble distribution uniformly at random selects one of the 3 distributions (it may select the one it is currently using). All uncollected resources are then re-distributed across the arena in accordance with the new selected distribution. This provides a dynamic resource distribution within a run.

5) Mystery Distribution: This distribution selects one of the 3 basic distributions uniformly at random and that distribution is kept throughout the hour-long run. Thus, during the CPFA optimization the GA does not know which distribution it will be using.

C. ARGoS Settings

The ARGoS environment was designed to simulate the real-world environment of the iAnt robot [9], [24]. Settings were chosen to ensure that any differences between algorithms and distributions were observable while still allowing the GA to solve the CPFA evolutionary runs in reasonable time. (The evolutionary runs for each distribution had a runtime of

approximately 2 days). The settings used by ARGoS for both CPFA optimization runs and the experiment runs that compare the algorithms are shown in Table II and Fig 4.

TABLE II. ARGOS ENVIRONMENT SETTINGS

Item	*Setting*
Robot Type	iAnt [24] (visual is a footbot)
Ticks/sec	16
Simulation Time	3600 sec (1 hour)
Initial Resource Count	1280
Initial Number of Clusters	4 (Clustered distribution always has 4 clusters initially)
Number of Robots	24
Arena Size	30m x 30m
DDSA Nest Angle Tolerance	0.05
DDSA Target Angle Tolerance	0.05
Initial Robot Positions	6 across x 4 deep over nest (see Fig 4)
Number of Experiment runs	100 runs were taken for all algorithms.

D. CPFA Distribution Parameter Settings

Table III provides the settings for all CPFA optimized distributions. The ADMA uses both the CPFA_u and CPFA_c settings for its foraging strategies and begins the simulation using the CPFA_u strategy for all robots. Refer to section II A for parameter explanations.

TABLE III. OPTIMIZED CPFA EXPERIMENT PARAMETER SETTINGS

Parameter	**CPFA Algorithm by Resource Distribution**				
	CPFA_u	***CPFA_c***	***CPFA_p***	***CPFA_j***	***CPFA_m***
p_s	1	1	1	1	0.1539
p_r	0	0	0	0	0
ω	0.0395	0.2401	0.0453	0.02032	0
λ_{id}	0.06925	0.2062	1.1314	0.1694	2.5484
λ_{sf}	12.6571	3.9013	4.3540	17.8361	13.068
λ_{lp}	16.3573	1.1025	18.165	16.5574	6.8329
λ_{pd}	0.30182	0.2460	1.9824	0.4946	1.1251

IV. RESULTS

Results are provided in a box and whisker plot of all 100 samples for a specific resource distribution. Each algorithm is noted above the chart color coded and in order of appearance left to right. The mean of the samples is shown as an "X" in the chart for each algorithm. Below the chart is a table of statistics that show the mean, standard deviation (SD), a 95% confidence interval (CI) and p value for an alpha of 0.05. The p value is calculated as a two-tailed t-test normal distribution with unequal variances comparing the ADMA to the listed algorithm. Very small p values are noted with a <0.0001 rather than the actual value. The order left to right is identical to the order in the charts for each figure.

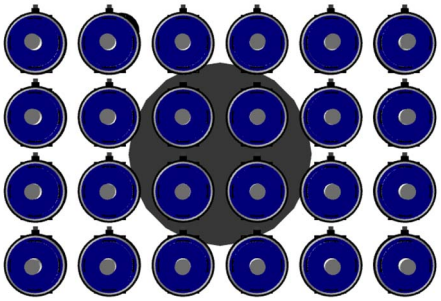


Fig. 4. ARGoS run robot start formation. The large black circle in the center is the nest. The blue and silver circles are the 24 robots.

A. Uniform Distribution

Fig 5a shows the comparisons for the uniform distribution. The ADMA mean value is significantly different than the mean of the optimal CPFA_u algorithm. However, the average performance of the ADMA is 99.2% of the CPFA_u algorithm collecting only 4 fewer resources on average. The CPFA_c performance is 68.9% of the CPFA_u and 69.4% of the ADMA making it the worst algorithm in this distribution. The DDSA also has a low performance at 81.8% of the CPFA_u.

B. Clustered Distribution

Fig 5b shows the comparisons for the clustered distribution. The ADMA mean performance is not statistically different than the CPFA_c, but the DDSA outperforms both the ADMA and the CPFA_c.

C. Power Law Distribution

Fig 5c shows the comparisons for the power law distribution. The ADMA mean performance is not significantly different than the optimal CPFA_p mean performance. The DDSA does show a significant increase compared to the ADMA, slightly outperforming it on average (ADMA performs at 97.7% the DDSA). The remaining algorithms perform slightly worse than the ADMA (1 -4 % worse).

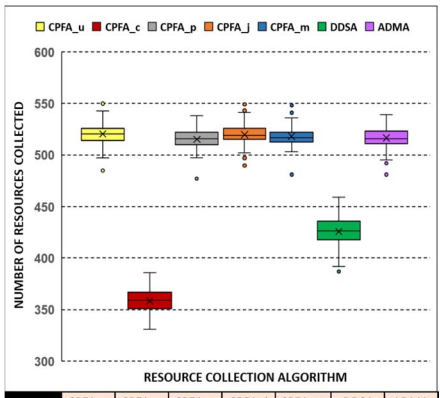
D. Jumble Distribution

Fig 6a shows the comparisons for the jumble distribution. The ADMA mean performance is not significantly different from the optimal CPFA_j or the CPFA_p mean performance. The ADMA outperforms all other algorithms including the DDSA.

Foraging performance on this distribution has a very high variance. This high variance is because all three basic distributions are represented in the data and the clustered distribution underperforms the other two by a large margin.

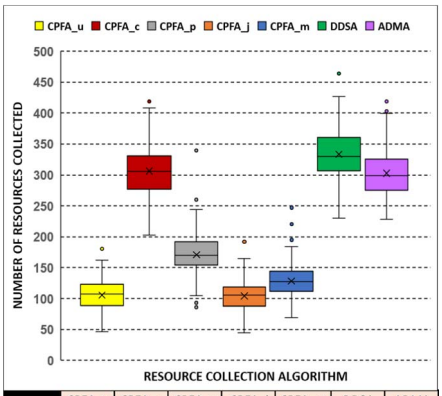
E. Mystery Distribution

Fig 6b shows the comparisons for the mystery distribution. The ADMA mean performance is better than the optimized CPFA_m mean performance. The ADMA is not significantly different than the DDSA which is the only algorithm that



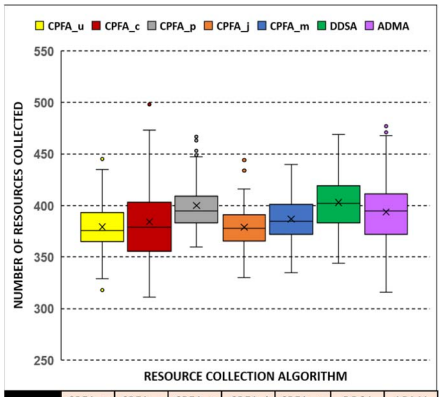
	CPFA_u	CPFA_c	CPFA_p	CPFA_j	CPFA_m	DDSA	ADMA
Mean	520.5	358.4	515.4	519.8	518.1	425.6	516.5
SD	10.6	11.4	9.5	10.3	9.6	16.3	10.4
CI	2.1	2.2	1.9	2.0	1.9	3.2	2.0
p value	0.0082	<0.0001	0.4201	0.0268	0.2772	<0.0001	

(a) Uniform Distribution



	CPFA_u	CPFA_c	CPFA_p	CPFA_j	CPFA_m	DDSA	ADMA
Mean	105.3	306.2	170.5	103.8	128.2	333.3	302.8
SD	27.7	38.1	34.7	26.9	28.7	47.8	38.1
CI	5.4	7.5	6.8	5.3	5.6	9.4	7.5
p value	<0.0001	0.5300	<0.0001	<0.0001	<0.0001	<0.0001	

(b) Clustered Distribution



	CPFA_u	CPFA_c	CPFA_p	CPFA_j	CPFA_m	DDSA	ADMA
Mean	379.3	384.1	400.0	378.9	386.9	402.9	393.7
SD	21.8	35.8	24.1	19.3	21.5	28.4	32.7
CI	4.3	7.0	4.7	3.8	4.2	5.6	6.4
p value	0.0003	0.0471	0.1266	0.0001	0.0806	0.0354	

(c) Power Law Distribution

Fig. 5. Box and Whisker plots of the performance in number of resources collected vs collection algorithms for the (a) Uniform distribution, (b) Clustered distribution, (c) Power law distribution. The 'x' indicates the mean of the data.

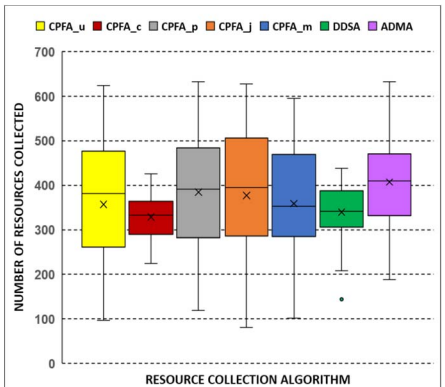
performs as well as the ADMA. The CPFA_p underperforms the ADMA by about 9% on average with the remaining algorithms below that.

This distribution also has a high variance for the same reasons as the jumble distribution. It may also be thought of as the merged version of the basic 3 distributions and gives a view of the ADMA flexibility across those distributions.

V. Discussion

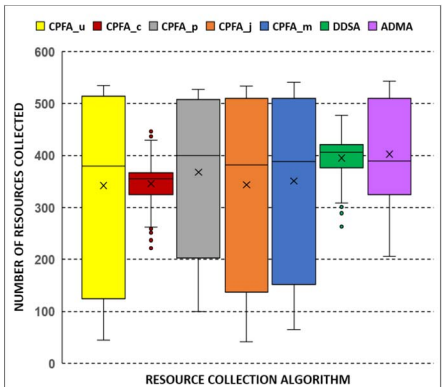
The ADMA foraging algorithm combines two foraging parameterizations of the CPFA, each of which is evolved for different extremes of resource distributions: The CPFA_u algorithm for the uniform distribution, and the CPFA_c algorithm for the highly-clustered distribution. As shown in Fig 5a, when the CPFA_c is run in the uniform distribution it performs far worse than all other algorithms. Similarly, in Fig 5b the CPFA_u performs far worse in the clustered distribution compared to CPFA_c (by a factor of over 50%). However, the ADMA efficiently uses both algorithms showing < 1% difference between it and the CPFA_u in the uniform distribution and showing no statistical difference with the CPFA_c in the clustered distribution.

The big advantage of the ADMA is that it does not need to know what the resource distribution is *a priori*. Prior studies on the CPFA effectively used an online GA to tune parameters to be optimized for a pre-specified resource distribution [27]. Here, we show that this tuning can be done once on extremes of the distribution and then reused by ADMA to provide a nearly equivalent foraging strategy. Additionally, the ADMA outperforms both the CPFA and DDSA when the resource distribution changes dynamically. Because the ADMA allows individual agents to respond to their current environment, it is not necessary for a global observer to ensure robots are properly proportioned for the task at hand. The ADMA is also tolerant



	CPFA_u	CPFA_c	CPFA_p	CPFA_j	CPFA_m	DDSA	ADMA
Mean	357.0	328.2	384.3	376.7	358.5	339.5	406.8
SD	141.3	47.2	128.7	147.9	129.1	60.7	103.9
CI	27.7	9.3	25.2	29.0	25.3	11.9	20.4
p value	0.0050	<0.0001	0.1754	0.0969	0.0040	<0.0001	

(a) Jumble Distribution



	CPFA_u	CPFA_c	CPFA_p	CPFA_j	CPFA_m	DDSA	ADMA
Mean	342.2	345.6	367.7	343.6	351.2	395.1	402.3
SD	171.5	48.3	143.8	169.9	161.4	44.0	98.5
CI	33.6	9.5	28.2	33.3	31.6	8.6	19.3
p value	0.0028	<0.0001	0.0488	0.0033	0.0076	0.5056	

(b) Mystery Distribution

Fig. 6. Box and Whisker plots of the performance in number of resources collected vs collection algorithms for the (a) Jumble distribution, (b) Mystery Distribution. The 'x' indicates the mean of the data.

of individual robot errors in decision making. For example, an individual robot may incorrectly select a distribution while being in the opposite distribution (e.g. a robot that detects 3 remaining resources in a uniform distribution has a 90% chance to select the CPFA_c algorithm even though the distribution is uniform). Individual robot errors only have a small impact on the overall swarm performance, and those decisions can be reversed as the robot continues to sample.

The ADMA performed as well as any other algorithm in the dynamic (jumble) and unknown (mystery) algorithms without time-consuming offline evolution. Comparing the ADMA to itself across the jumble and mystery distribution there is no significant difference in means (p value = 0.7502). This implies the ADMA is not affected by the changing dynamic environment.

All these factors indicate that the ADMA achieves the desired flexibility to operate in all distributions tested at a level that is beyond the other CPFA algorithms.

While the DDSA is very effective for static distributions, The ADMA outperforms it in the dynamic case because the DDSA does not revisit areas it has already checked while the ADMA (and the CPFA) will. In the case of the uniform distribution where the DDSA performed unusually poorly this is because of the simple collision detection which causes bigger delays for the deterministic robot behaviors in the DDSA compared to the ADMA and CPFA in which robots are not required to follow specific paths. Improving collision detection or using collision avoidance will likely have a larger initial impact on the DDSA than on the other algorithms.

VI. Conclusion

We show that we can combine foraging strategies using a task allocation mechanism such that the global behavior of the swarm emerges from individual robots that choose their behaviors based on locally sensed conditions in the environment. By allowing individual agents to choose behaviors based on a simple response threshold model, we increase the flexibility of the swarm allowing it to collect many resources in all resource distributions including unknown and dynamic distributions.

The chosen strategies are dependent on local observations of an individual robot rather than a global observer or communication among robots. This maintains the distributed nature of the swarm and allows the method to scale to larger swarms.

Our ADMA demonstrates this by either outperforming the optimized CPFA algorithm or showing no significant difference in 4 out of 5 cases. In the one case where performance of the ADMA was statistically significantly lower than the optimal parameterization of the CPFA (in the uniform distribution), the difference was < 1% lower.

The ADMA also performed consistently across all distributions, demonstrating that it is more flexible than the CPFA and DDSA.

Our goal in this work was to demonstrate the emergence of effective and flexible swarm behavior from individual robot choices based on their local observations of the environment. The emergent swarm behavior is flexible enough to effectively forage in previously unseen and dynamic resource distributions.

Additional strategies may be added to the ADMA by finding appropriate decision points for those strategies and applying response threshold models as a method of decision-making. This would allow more complex behaviors to emerge to perform other tasks such as cooperative foraging, swarm resource management and other division of labor tasks.

Acknowledgment

The authors would like to thank all the members of the Moses Biological Computation lab who have provided support, insite and valuable critiques of the work. This work is supported by funding from James S. McDonnell Foundation Complex Systems Scholar Award and the NASA Education (MUREP) Swarmathon award #NNX15AM14A.

References

[1] M. P. Hassel and T. R. E. Southwood, “Foraging strategies of insects,” Annual Review of Ecology and Systematics 9:75–98, November 1978. Available: https://doi.org/10.1146/annurev.es.09.110178.000451.

[2] A. C. Kamil and T. D. Sargent, “Foraging behavior: ecological, ethological, and psycho-logical approaches,” Garland STPM Press, 1981.

[3] M. E.Ritchie, “Scale, heterogeneity, and the structure and diversity of ecological communities,” Princeton University Press, Princeton, 2010.

[4] J. A. Wiens, N. C. Stenseth, B. V. Horne, and R. A. Ims, “Ecological mechanisms and landscape ecology,” Oikos, 66 no. 3, 369–380, 1993 Available: http://doi.org/10.2307/3544931.

[5] Liviu Panait and Sean Luke, “A pheromone-based utility model for collaborative foraging,” In Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems - Volume 1 (AAMAS '04), Vol. 1. IEEE Computer Society, Washington, DC, USA, pp. 36-43, 2004.

[6] E. J. Robinson, F.L. Ratnieks, and M. Holcombe, “An agent-based model to investigate the roles of attractive and repellent pheromones in ant decision making during foraging,” Journal of Theoretical Biology, 255(2), pp. 250-258, August 22, 2008 Available: http://doi.org/10.1016/j.jtbi.2008.08.015.

[7] Z. Meng, B. Zou, Y. Zeng Y, “Considering direct interaction of artificial ant colony foraging simulation and animation,” Journal of Experimental & Theoretical Artificial Intelligence 24(1), pp.95–107, 2012. Available: http://dx.doi.org/10.1080/0952813X.2010.545999.

[8] N.R. Hoff, A. Sagoff, R.J. Wood, R. Nagpal, “Two foraging algorithms for robot swarms using only local communication,” in: IEEE International Conference on Robotics and Biomimetics (ROBIO), Piscataway, pp. 123–130, December 2010. Available: http://dx.doi.org/10.1109/ROBIO.2010.5723314.

[9] J. P. Hecker and M. E. Moses, “Beyond pheromones: evolving error-tolerant, flexible, and scalable ant-inspired robot swarms,” Swarm Intelligence, vol. 9, no. 1, pp. 43–70, March 2015. Available: http://dx.doi.org/10.1007/s11721-015-0104-z.

[10] Sjriek Alers, Daan Bloembergen, Daniel Hennes, Steven de Jong, Michael Kaisers, Nyree Lemmens, Karl Tuyls, and Gerhard Weiss, “Bee-inspired foraging in an embodied swarm,” In The 10th International Conference on Autonomous Agents and Multiagent Systems (AAMAS '11), vol.3, International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC, pp. 1311-1312, 2011.

[11] Sjriek Alers, Daniel Claes, Karl Tuyls, and Gerhard Weiss, “Biologically inspired multi-robot foraging,” In Proceedings of the 2014 international conference on Autonomous agents and multi-agent systems (AAMAS '14). International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC, pp. 1683-1684, 2014.

[12] Jong-Hyun Lee, Hyun-Tae Kim, and Chang Wook Ahn, “Foraging swarm robots system adopting honey bee swarm for improving energy

efficiency," In Proceedings of the 6th International Conference on Ubiquitous Information Management and Communication (ICUIMC '12). ACM, New York, NY, USA, Article 100, 4 pages, 2012. Available: http://dx.doi.org/10.1145/2184751.2184867.

[13] A.F.T. Winfield, "Foraging robots," In Encyclopedia of complexity and systems science, New York, NY: Springer. pp. 3682–3700, 2009.

[14] D. Levin, J. P. Hecker, M. E. Moses, and S. Forrest. "Volatility and spatial distribution of resources determine ant foraging strategies," In Proceedings of the European Conference on Artificial Life 2015 (ECAL 2015), Cambridge, MA, MIT Press, pp. 256–263, 2015. Available: http://dx.doi.org/10.7551/978-0-262-33027-5-ch050.

[15] T. H. Labella, M. Dorigo and J-L. Deneubourg, "Efficiency and task allocation in prey retrieval", Biologically Inspired Approaches to Advanced Information Technology Lecture Notes in Computer Science, Vol. 3141, pp. 274-289, 2004.

[16] E. Bonabeau, E., G. Theraulaz, & JL . Deneubourg, "Quantitative study of the fixed threshold model for the regulation of division of labour in insect societies," Proc. Roy. Soc. Lond. B263, 1565–1569, 1996.

[17] E. Bonabeau, E., G. Theraulaz, & JL . Deneubourg, "Fixed response thresholds and the regulation of division of labor in insect societies," Bulletin of Mathematical Biology, Vol. 60 pp. 753-807, 1998. Available: http://dx.doi.org/10.1006/bulm.1998.0041.

[18] M. J. B. Krieger and J-B. Billeter, "The call of duty: self-organised task allocation in a population of up to twelve mobile robots", Robotics and Autonomous Systems, Vol. 30, pp. 65-84, 2000.

[19] Y. Yang, C. Zhou, Y. Tian, "Swarm robots task allocation based on response threshold model," International Conference on Autonomous Robots and Agents. IEEE, pp. 171-176, 2009.

[20] H. Ishiguro, "Foraging optimization in swarm robotic systems based on an adaptive response threshold model". Advanced Robotics, 28(20):1343-1356, 2014.

[21] W. Lee and D. Kim, "Adaptive division of labor in multi-robot system with minimum task switching," In ALIFE 14: Proceedings of the fourteenth international conference on the synthesis and simulation of living systems, MIT Press, pp. 750–756, 2014. Available: http://dx.doi.org/10.7551/978-0-262-32621-6-ch120

[22] A. Kanakia and N. Correll, "A response threshold sigmoid function model for swarm robot collaboration," Distributed and autonomous robotic systems (DARS), volume 112 of the series springer tracts in advanced robotics, Heidelberg: Springer, pp. 193–206, 2016

[23] A. Kanakia, B. Touri, and N. Correll, "Modeling multi-robot task allocation with limited information as global game," Swarm Intell, vol 10, pp. 147-160, 2016 Available: http://dx.doi.org/10.1007/s11721-016-0123-4.

[24] J. P. Hecker, K. Stolleis, B. Swenson, K. Letendre, and M. E. Moses, "Evolving error tolerance in biologically inspired iant robots," in InbvECAL 2013.

[25] G. M. Fricke, J. P. Hecker, A. D. Griego, L. T. Tran and M. E. Moses, "A distributed deterministic spiral search algorithm for swarms," 2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Daejeon, pp. 4430-4436, October, 2016 Available: http://dx.doi.org/10.1109/IROS.2016.7759652.

[26] C. Pinciroli, V. Trianni, R. OGrady, G. Pini, A. Brutschy, M. Brambilla, N. Mathews, E. Ferrante, G. Di Caro, and F. Ducatelle, "ARGoS: a modular, parallel, multi-engine simulator for multi-robot systems,"Swarm intelligence, vol. 6, no. 4, pp. 271–295, 2012.

[27] J. P. Hecker., & M. E. Moses, "Real-time evolution of iAnt robot foraging strategies," In ALIFE 14: Proceedings of the Fourteenth International Conference on the Synthesis and Simulation of Living Systems. Cambridge, MA: MIT Press, pp. 835–836, 2014. Available: http://doi.org/10.7551/978-0-262-32621-6-ch136.