

EVOLVING AGGREGATION BEHAVIORS FOR SWARM ROBOTIC SYSTEMS: A SYSTEMATIC CASE STUDY

Erkin Bahçeci

KOVAN Research Lab.,
Department of Computer Engineering
Middle East Technical University,
06531, Ankara, Turkey
E-mail: erkinb@ceng.metu.edu.tr

Erol Şahin

KOVAN Research Lab.,
Department of Computer Engineering
Middle East Technical University,
06531, Ankara, Turkey
E-mail: erol@ceng.metu.edu.tr

ABSTRACT

When one attempts to use artificial evolution to develop behaviors for a swarm robotic system, he is faced with decisions to be made regarding the parameters of the evolution. In this paper, we chose the aggregation behavior as a case, and systematically studied the performance and the scalability of aggregation behaviors of perceptron controllers evolved for a simulated swarm robotic system with different parameter settings. We have conducted four experiments, varying some of the parameters and derived rules of thumb which can be of guidance to the use of evolutionary methods to generate other swarm robotic behaviors.

1. INTRODUCTION

Swarm robotics[1, 2] is a new approach to the coordination of large numbers of relatively simple robots. The approach takes its inspiration from the system-level functioning of social insects which demonstrate three desired characteristics for multi-robot systems: robustness, flexibility and scalability. Most of the studies[3, 4] on swarm robotics focus on developing behaviors with these desired characteristics.

Evolutionary methods are becoming promising candidates to develop behaviors for swarm robotic systems. However, when one attempts to use evolution to develop behaviors for a swarm robotic system, he is faced with decisions to be made regarding the parameters of the evolution. In this paper, we chose the aggregation behavior, which can be considered a pre-cursor for most swarm behaviors, as a case, and conducted systematic experiments varying some of the parameters and derived rules of thumb which can be of guidance to the use of evolutionary methods to generate other swarm robotic behaviors.

In the next section, we review earlier studies that set the stage for our paper.

2. RELATED WORK

Early studies on evolving behaviors for swarm robotic systems reported limited success and expressed pessimistic conclusions. In one of the earliest studies, Zaera *et al.*[5] used evolution to develop behaviors for dispersal, aggregation, and schooling in fish. Although they had evolved successful controllers for dispersal and aggregation; the performance of the evolved behaviors for schooling was considered disappointing, and they concluded that for complex actions like schooling, manual design of a controller would require less time and effort than evolving one, mainly due to the difficulty of determining a useful evaluation function for the specific task.

Mataric *et al.*[6] have made a comprehensive review of the studies until 1996 on evolving controllers to be used in physical robots and they have discussed the key challenges. They addressed approaches and problems such as evolving morphology and/or controller, evolving in simulation or with real robots, fitness function design, co-evolution, and genetic encodings. They emphasized that for an evolved controller to be beneficial, the effort to produce it in evolution should be less than the effort needed to manually design a controller for the same robotic task. They stated that it has not been the case, yet; but when the challenges and problems are handled, they may become a practical alternative to controllers designed by hand.

In [7], Lund *et al.* used evolution to develop minimal controllers for the task considered. They evolved controllers for the Khepera robot (K-Team, Switzerland) for exploration and homing task where the robot was desired to leave a light source, i.e., home, explore the surrounding for some time, and then return back home where it is virtually recharged. To obtain this periodic behavior, they used sampled sensory input and a minimal network architecture without recurrent connections, which can be used to obtain the notion of *return period*. Instead their evolution exploited the geomet-

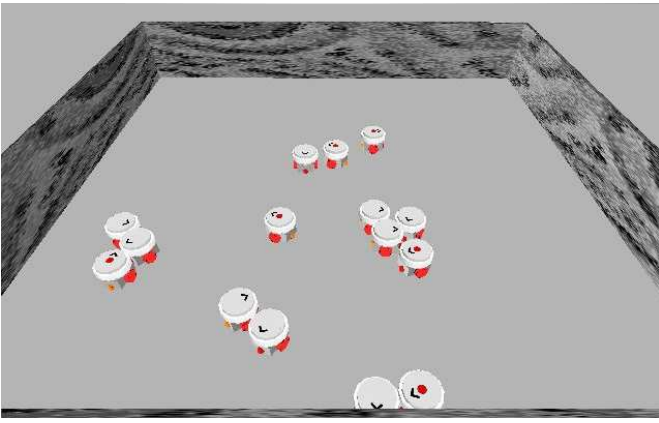


Figure 1. A screenshot of the simulator.

rical shape as perceived by robot and produced a suitable controller.

In contrast to some of these pessimistic conclusions, during the recent years optimistic results are being reported on the evolution of swarm behaviors. In the Swarm-bots Project[8], Baldassarre *et al.*[9] successfully evolved controllers for a swarm of robots to aggregate and move towards a light source in a clustered formation. Moreover, for this specific task, several distinct movement types emerged: constant formation, amoeba (extending and sliding), and rose (circling around each other). In [8], Trianni *et al.* also evolved successful controllers for a swarm of robots that can grip each other, called a swarm-bot, to fulfill tasks such as aggregation, coordinated motion in a common direction, cooperative transport of heavy loads (as in ants), and all-terrain navigation to avoid holes (connected in swarm-bot formation). Their evolved controllers made use of sound sensors, traction sensors, and flexible links. Trianni *et al.*[10] has also identified two types aggregation behaviors emerged from evolution: a dynamic and a static clustering behavior. In static clustering, robots move in circles until they are attracted to a sound source. Then they *bounce* against each other until an aggregate is formed. The clusters are tight and static with the robots involved turning on the spot, whereas in dynamic aggregation, the clusters are loose and they flock around. This study is a good example of evolution of different strategies, or behaviors, for a specific task. Furthermore, in [11] Dorigo *et al.* evolved aggregation behaviors for a swarm robotic system. They analyzed two of the evolved behaviors and showed that evolution was able to discover rather scalable behaviors.

Ward *et al.* [12] have evolved neural network controllers for such a survival scenario where there are two populations of animals, predators and preys, that co-evolve to produce a schooling behavior. They have also studied on the connection of physiology with behavior and they claim that prey

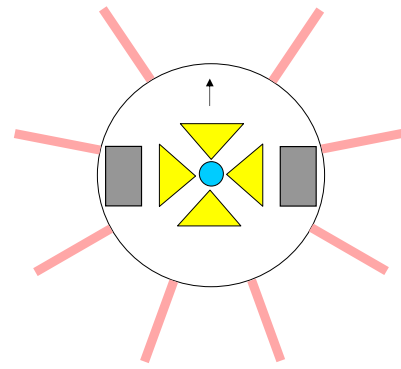


Figure 2. A schematic view of the robot model. The robot has a diameter of 5.8 units. The 8 bars emanating from the body of the robot indicate the IR sensor direction and range. The 4 triangles are placed at the center represent microphones, 2 rectangles at the sides represent wheels, and the circle at the center represents an omni-directional speaker.

need a wide-range low-resolution visual sensors whereas predators are better off with visual input concentrated in the front.

Despite these studies, the use of evolution to generate swarm robotic behaviors for a desired task is a rather unexplored field of study. When one attempts to use evolution to evolve a behavior for swarm robotic systems he is faced with decisions to be made regarding the use of evolution. To the best of our knowledge, no systematic study has been made to investigate effects of parameters to help such decisions.

For this paper, we chose aggregation behavior as the case for our study on evolution. Aggregation behavior is observed in almost all social animals. Animals either use aggregation to increase their chances of survival, or they use aggregation as a pre-cursor of other behaviors. For example, self-assembly and pulling heavy objects require prior aggregation at the site of interest.

In this paper, we systematically studied the performance and the scalability of aggregation behaviors evolved for a simulated swarm robotic system. The control architecture evolved for this task was a perceptron connecting sound and IR inputs to wheel and speaker actuators. The effect of different parameter choices on the performance and the scalability of the aggregation behaviors are analyzed.

3. EXPERIMENTAL FRAMEWORK

We used a port¹ of the Swarmbot3D simulator [13], a physics-based simulator developed within the Swarm-bots project

¹We ported the Swarmbot3D simulator to the Open Dynamics Engine, a free physics-based simulation library.

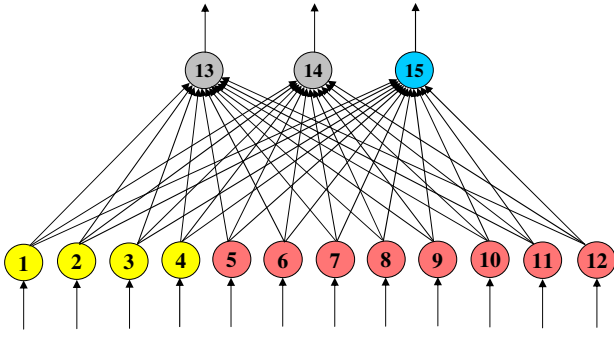


Figure 3. Neural network controller used as the controller for robots. Neurons match as follows to Fig. 2: 1-4: microphones, 5-12: IR sensors, 13-14: wheel actuators, and 15: speaker.

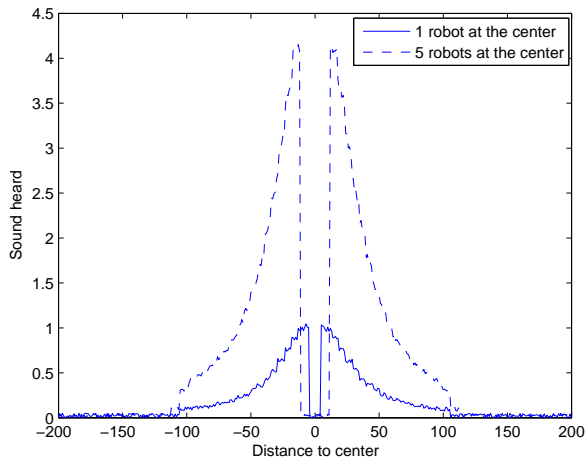


Figure 4. Sound heard at certain distances with one and five robots emitting sound at the center. The audibility values shown are the maximum of sensory input values recorded by the four microphones of a virtual robot which is placed at different distances from one wall to the opposite on a line intersecting the center of a 400×400 unit arena. Higher values indicate higher audibility. The region with 0 values near the center is occupied by the sound emitting robots. Arenas that are used in the experiments have sizes 110×110 , 140×140 , 200×200 , 282×282 and 400×400 units.

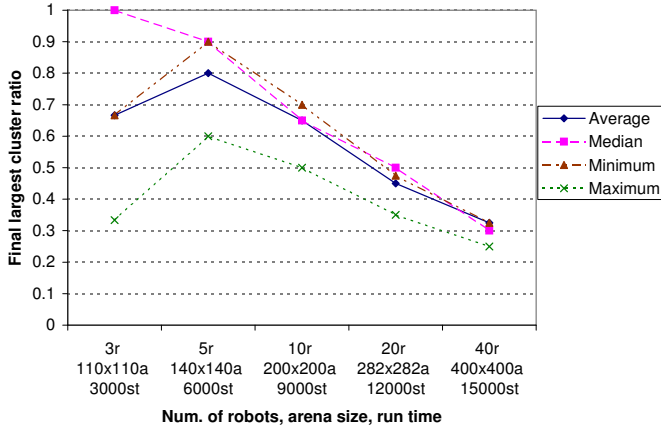
that modeled the s-bots (mobile robots with the ability to connect to each other). The Swarmlab3D simulator included simulation models of the s-bot at different levels, all obtained from and verified against the actual s-bot. We used the minimal s-bot model, with which evolution of aggregation behavior was first studied by Dorigo *et al.* in [11]. A snapshot of the simulator is shown in Fig. 1.

A schematic view of the robot indicating the sensor and signal source configuration used in our experiments is shown in Fig. 2. The robot is modeled as a differential drive robot with two wheels. The model has 8 infrared range sensors around the robot, and one omni-directional speaker and 4 directional microphones placed at the center of the robot. The details of these models were described in detail in [13]. Both the infrared and the sound sensors are modeled using sampling data obtained from the real robot with the addition of white noise as described in [13] and [9]. Figure 4 shows the characteristics of the sound sensor model which drives the long-range interactions among the robots. As it is, the sound sensor model can be regarded as unrealistic due to its simplicity. However, using a proper placement of microphones robust sound source localization can be done as in [14], where Valin *et al.* has localized sound sources with a precision of 3 degrees in 3 meters range using an array of 8 sound sensors placed at the corners of a rectangular prism.

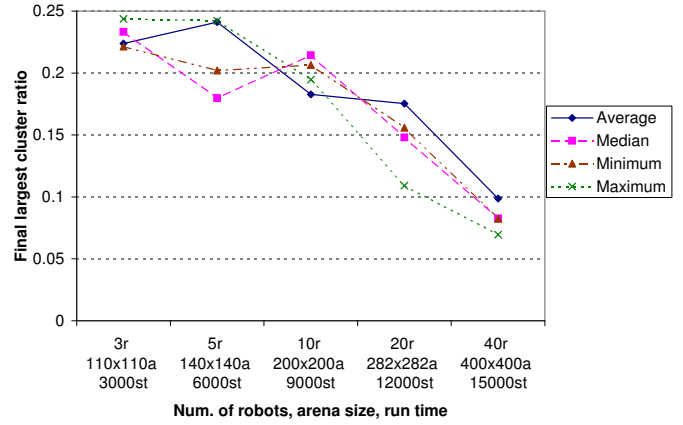
It should be noted, however, that our simulator was neither verified against the original Swarmlab3D simulator, nor against the physical robots. Therefore, we make no claims about the portability of the evolved controllers onto the physical robots. Yet, for the purpose of this study, we believe that the sensor and signaling models which were taken from the Swarmlab3D simulator are sufficient since our study aims to deduce general rules of thumb for evolving behaviors in swarm robotic systems.

Our goal is to evolve controllers that would aggregate a swarm of robots that are initially dispersed in an environment. To this end, the robots are initially placed in an empty square arena, shown in Fig. 1, at random position. Then the swarm robotic system is simulated, with each robot being controlled by the same controller.

The controller is chosen to be a single-layer perceptron which has 12 input neurons (4 connected to microphones and 8 connected to infrareds), 3 output neurons (1 to control the omni-directional speaker and 2 to control the wheels) as seen in Fig. 3. The connection weights of the perceptron are encoded as 39 floating point numbers on a chromosome, or population member. Here, for a chromosome, mutation is defined as choosing one weight out of all 39 weights on the chromosome, and adding a random value in ± 1.0 range. Each chromosome is subjected to this type of mutation with a probability of 0.5 and to a crossover with a probability of 0.8. This means that each network weight in the population has a mutation probability less than $\frac{0.5}{39}$. The genetic



(a) Results



(b) Standard deviation

Figure 5. Results of experiment 1: Different integrations of fitness values of the same controller (evolved with 10 robots, 6000 steps, and 5 runs/controller). The y -axis shows the median and standard deviation of fitness values of 50 *evaluation* runs for each set-up and the x -axis designates 5 different set-ups used to evaluate scalability performance of produced controllers by the evolutions depicted on the legend. The evaluation set-ups increase in number of robots, size of arena, and number of simulation steps from left to right.

algorithm is run with a population of 50 chromosomes. At each generation, depending on their fitness, the best 10% of the population is copied unchanged to the next generation, i.e., elitism, together with the rest which is mutated and subjected to crossover.

In order to evaluate the fitness of a chromosome, the perceptron defined by that particular chromosome is replicated as the controller for all the robots in the swarm, and the swarm robotic system is simulated for a certain number of steps.

In our study, robots i and j are referred to as neighbors if $Neighbor(i, j)$, defined in Equation 1, is true; and they are in the same cluster, or aggregate or group, if $Connected(i, j)$, defined in Equation 2, is true.

$$Neighbor(i, j) = \begin{cases} true & \text{if distance between} \\ & i \text{ and } j \leq 10 \\ false & \text{otherwise} \end{cases} \quad (1)$$

$$Connected(i, j) = \begin{cases} true & \text{if there is a path from} \\ & i \text{ to } j \text{ over the} \\ & \text{relationship } Neighbor \\ false & \text{otherwise} \end{cases} \quad (2)$$

At the end of a simulation run, sizes of clusters are computed. The aggregation performance, or *fitness*, of a single evaluation run is defined as the ratio of the number of robots forming the largest cluster to the total number of robots. The fitness of a chromosome is defined as in Equation 3.

$$Fitness = F(fitness_1, \dots, fitness_{n_{runs}}) \quad (3)$$

where F , fitness combining function is one of *average*, *median*, *minimum*, and *maximum*. to join the fitness values of n_{runs} simulation runs done for a single chromosome. These runs differ in their randomization seed. This seed determines the initial placement of robots. $fitness_i$ in this equation refers to the fitness value of a simulation run with the i^{th} random seed.

Evolution of controllers is done using PES (Parallelized Evolution System) on a Beowulf cluster with 128 nodes. PES [15], a software platform implemented using the PVM library, parallelizes the fitness evaluations of evolutionary methods over multiple computers connected via a network.

4. EXPERIMENTS

We conducted four different experiments to investigate the effect of different parameters on performance and scalability of evolved behaviors. Parameters altered in the evolution experiments are **(a)** the fitness combining method $F(\cdot)$, **(b)** the number of runs per controller (n_{runs}), **(c)** the number of simulation steps, and **(d)** the set-ups. In experiments 1, 2, and 3, the total number of simulation steps in a run remained the same for different parameter choices.

For each given set of parameters, one evolution is run and the resulting aggregation behaviors are analyzed. During the analysis, each behavior is tested with 50 seeds on

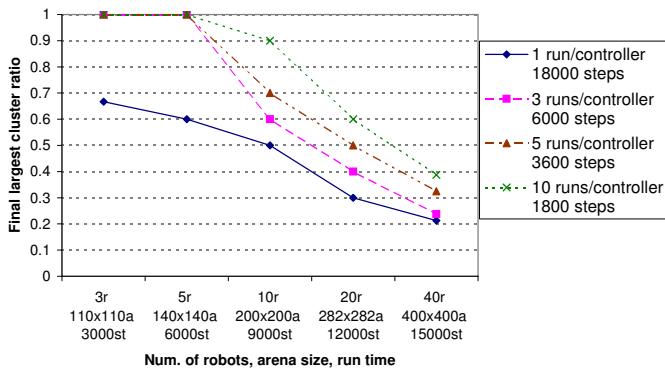


Figure 6. Results of experiment 2: Different number of runs for the same controller are varied while keeping total number of steps for a controller constant. Evolution is done with 10 robots.

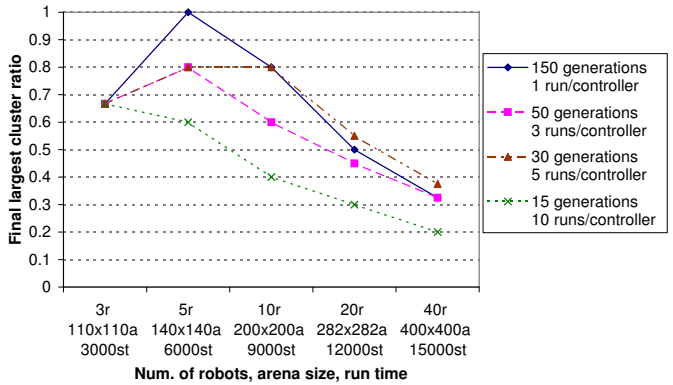


Figure 7. Results of experiment 3: The number of generations and the number of runs for the same controller are varied while keeping total number of steps constant. In this experiment, evolution is done with 10 robots in 6000-step simulations.

5 different set-ups, that are shown in Table 1. In all these set-ups the robot density over the arena is kept the same. The number of simulation steps is increased in larger arenas to allow more time for aggregation. In these tests, both the performance and the scalability of the evolved behaviors are evaluated.

set-up	# Robots	Arena Size	# Simulation Steps
1	3	110 × 110	3000
2	5	140 × 140	6000
3	10	200 × 200	9000
4	20	282 × 282	12000
5	40	400 × 400	15000

Table 1. Set-ups used for evaluation.

The first experiment is motivated by the question of how the different performance evaluations (each obtained from different initial conditions) of a controller should be combined to obtain the best result. This is an important issue, since initial placement of robots in the arena creates a large bias for the resulting performance and therefore a fair evaluation of different controllers requires multiple performance evaluations each starting from a different initial placement.

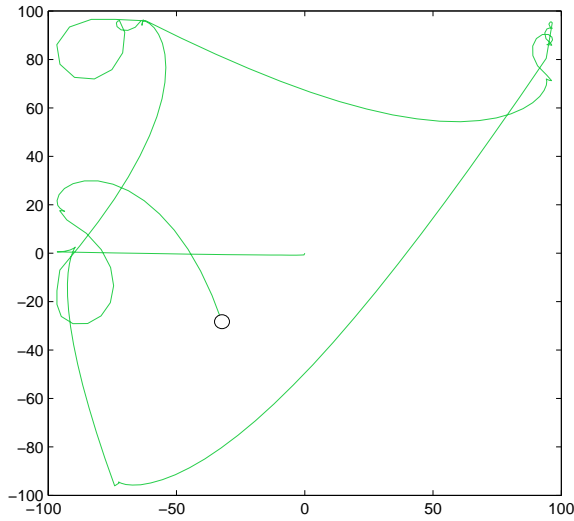
We examined the effect of fitness combining method, i.e., to combine the fitness values of n_{runs} simulations done for a controller. The results, shown in Fig. 5(a), indicate that among the four functions used (*average*, *median*, *minimum*, and *maximum*), *minimum* and *median* should be preferred for better results. Use of the *maximum* function, which corresponds to optimistic evaluation, is clearly the worst of the four. One possible reason may be the high variance inherent in all of the fitness evaluations, which can be seen in the standard deviation plot in Fig. 5(b).

The behavior of one of the best controllers evolved in this experiment can be seen in Figures 8(a) and 8(b). The evolved strategy can be described as “go straight until a sound is heard while avoiding walls, approach the loudest sound source, and then rotate, i.e. do not change position”. The emergent behavior of formed groups is to move slowly toward the loudest sound source. This can be seen in paths of groups in Fig. 8(b), which are slowly going towards each other.

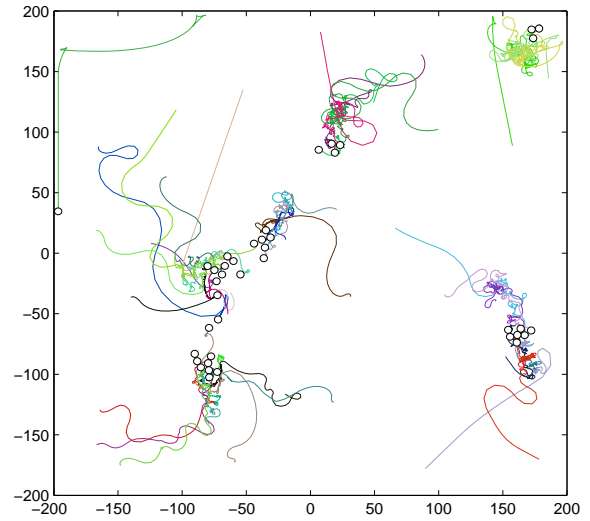
The second experiment investigates the trade-off between n_{runs} and simulation duration while keeping the number of total simulation steps executed for a specific controller constant. Figure 6 shows a significant monotonous increase in performance as n_{runs} increases although duration of simulation decreases. This implies that number of runs for a controller is clearly more important than simulation duration.

The third experiment (Fig. 7) investigates the trade-off between n_{runs} and number of generations while keeping the number of total simulation steps in the whole evolution constant. In this trade-off, the change in performance is not monotonous as in the previous experiment. The results show that a controller is not yet mature at 15 generations, hence the low performance of that case. They also show that the $n_{gens}=30 - n_{runs}=5$ pair seems to be the best point of this trade-off. While $n_{gens}=150$ case seems to perform as good as that on the average, doing one run per controller may impose problems due to possible bias for a lucky initial random placement.

Experiment 4 (Fig. 9) investigates the effect of set-up size on performance and scalability. Unlike the first three experiments, which were conducted to find out how to use total processing time most effectively, this experiment does



(a) A single robot in an arena of size 200×200 after 10000 time steps.



(b) 40 robots in an arena of size 400×400 after 15000 time steps.

Figure 8. The behavior of an evolved controller. Final positions of robots are shown as circles together with the paths they followed during the whole simulation run.

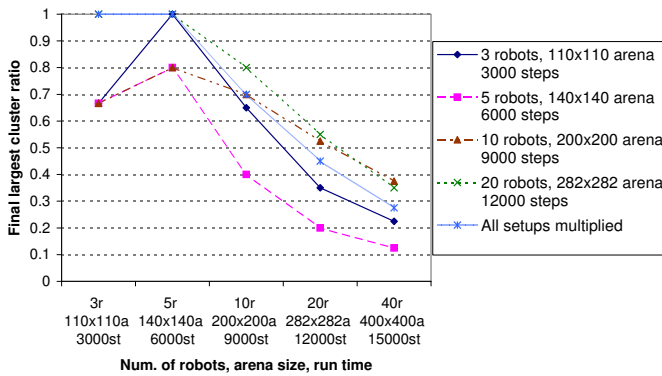


Figure 9. Results of experiment 4: Varying evolution set-up size, i.e. number of robots, arena size, and number of simulation steps. In this experiment, evolution is done with 3 runs/controller for the first four evolutions and $4 \times 3 = 12$ runs/controller for the multiplied set-up evolution.

not keep total number of simulation steps in the whole evolution constant. Instead it analyzes how good controllers evolved with different set-ups perform on smaller/larger set-ups. It tries to find out which evolution set-up size leads to the best scalability and also whether using all set-ups in one evolution (where calculating fitness for example by multiplying their results) improves overall scalability.

In the experiment, four evolutions are executed with single set-ups and one with all set-ups, multiplying the fitness values obtained from each set-up. The results show that 5-robot evolution generally produced the worst controller, even when evaluated on its own set-up. Again we do not see a monotonous change in performance with set-up size used in evolutions, the controller evolved with 3-robots seems better than the one with 5-robots. In fact, except the 20-robot evolution, none of the evolutions produced the best controller on their own set-ups. 10-robot evolution seems to have produced a controller suitable to larger set-ups, which can be observed from its line that has significantly different slopes than the other four controllers, tending to score higher than others as set-up size increases. The slopes of the curves hint us that 10-robot evolution may score higher than others for set-ups with more than 40 robots. For the evaluation set-ups considered, overall the best controller seems to be produced by the 20-robot evolution.

It is interesting to note that the multiplied set-up evo-

lution, which was especially designed for better scalability results, performs close but not as good as the 20-robot evolution. We believe that this unexpected result is most probably relevant to the high variance in performance which can be seen in the result graphs.

5. CONCLUSIONS

We studied how different parameters of evolutionary methods affect the performance and the scalability of behaviors in swarm robotic systems. We chose the aggregation behavior as our case and made four systematic experiments. These experiments investigated trade-offs among number of runs per controller, number of generations in the genetic algorithm, and number of simulation steps to find out the most beneficial resource to dedicate processing time to. Furthermore, this study examined how to best merge fitness results obtained from simulation runs of the same controller with different seeds. Finally, one more experiment was done to better understand how evolution set-up size affects the scalability and performance on set-ups of different size.

Based on the results obtained from the experiments conducted, we conclude the following rules of thumb:

- The use of optimistic functions (like *maximum* to combine performance values obtained from different runs of a controller should be avoided. Instead *median* or *minimum* should be preferred.
- When faced with the trade-off between the number of simulation steps for each run and the number of different runs per controller, one should choose the minimum sufficient number of simulation steps while maximizing the number of runs per controller. This will considerably eliminate negative effects of the high variance observed in robotics applications when initial positions are random.
- The optimum value of the number of runs per controller and the number of generations (which is as important as number of runs) is not easy to obtain. Number of generations in evolution needed for emergence of a controller with acceptable performance, depends on architectural complexity of the controller and difficulty of the task. It is best to let the evolution run once initially for many generations to see about when the performance reaches a reasonable level.
- In fitness evaluation, running simulation in multiple set-ups of different scale and multiplying the results does not necessarily improve scalability. Multiple-setup strategy should be more successful if more simulation runs per controller can be done, which would reduce drawbacks caused by outliers.

We believe that these results obtained through the systematic experiments have a high chance of being relevant both for evolving other swarm robotic behaviors in simulation, and for evolving behaviors for physical robotic systems.

This study can be extended by considering tasks other than aggregation and verifying the results on them. Also, more experiments can be carried out to investigate effects of other parameters which do not influence total run-time such as mutation-crossover rates, and different fitness measures. Moreover, experiments can be conducted to further explore optimal regions in trade-offs among parameters that affect total run-time, such as population size in the genetic algorithm, simulation run-time, and number of simulations for each chromosome. Finally, the results can be strengthened more by applying the evolved controllers to physical robots and evaluating the performance and scalability with different number of robots and with arenas of different size or shape.

Acknowledgements

This work was partially funded by the SWARM-BOTS project, a Future and Emerging Technologies project (IST-FET) of the European Community, under grant IST-2000-31010, and by the “Kontrol Edilebilir Robot Oğulları” Career Project awarded to Erol Şahin (104E066) by TÜBİTAK (Turkish Scientific and Technical Council).

The computational resources used in this work are provided by the TÜBİTAK ULAKBİM High Performance Computing Center.

6. REFERENCES

- [1] M. Dorigo and E. Şahin, *Swarm Robotics - Special Issue Editorial*. Autonomous Robots, vol. 17, no. 2, 2004.
- [2] E. Şahin, *Swarm Robotics: From Sources of Inspiration to Domains of Application*. Swarm Robotics Workshop: State-of-the-art Survey, Erol Şahin and William Spears, editors, Lecture Notes in Computer Science 3342, pp. 10-20, Springer-Verlag, 2005.
- [3] M. Dorigo and E. Şahin, *Swarm Robotics - Special Issue*. Autonomous Robots, vol. 17, pp. 111-113, 2004.
- [4] E. Şahin and W. M. Spears, *Swarm Robotics Workshop: State-of-the-art Survey*. Erol Şahin and William Spears, editors, Lecture Notes in Computer Science 3342, Springer-Verlag, 2005, ISBN 3-540-24296-1.
- [5] N. Zaera, C. Cliff, and J. Bruten, *(Not) Evolving Collective Behaviours in Synthetic Fish*. In From Ani-

- mals to *Animats 4: Proceedings of the Fourth International Conference on Simulation of Adaptive Behaviour*, Cambridge, MA: MIT Press, pp. 635-6, 1996.
- [6] M. Mataric and D. Cliff, *Challenges in Evolving Controllers for Physical Robots*. Robotics and Autonomous Systems, vol. 19, pp. 67-83, 1996.
- [7] H. H. Lund and J. Hallam, *Evolving Sufficient Robot Controllers*. In Proceedings of the Fourth IEEE International Conference on Evolutionary Computation, pp. 495-499. IEEE Press, Piscataway, NJ, 1997.
- [8] M. Dorigo, E. Tuci, R. Gross, V. Trianni, H. Labella, S. Nouyan, J.-L. Deneubourg, G. Baldassarre, S. Nolfi, F. Mondada, D. Floreano, and L. M. Gambardella, *The SWARM-BOTS Project*. In Swarm Robotics: SAB 2004 International Workshop, Santa Monica, CA, USA, E. Şahin and W. M. Spears editors, Lecture Notes in Computer Science, Volume 3342, Springer Verlag, p.31, July 17, 2004.
- [9] G. Baldassarre, S. Nolfi, and D. Parisi, *Evolving Mobile Robots Able to Display Collective Behavior*, Artificial Life, vol. 9, no. 3, pp. 255-267(13), 1 August 2003.
- [10] V. Trianni, R. Groß, T.H. Labella, E. Şahin, and M. Dorigo, *Evolving Aggregation Behaviors in a Swarm of Robots*. In W. Banzhaf, T. Christaller, P. Dittrich, J. T. Kim, and J. Ziegler, editors, Lecture Notes in Artificial Intelligence 2801, pp. 865-874, Springer Verlag, Heidelberg, Germany, 2003.
- [11] M. Dorigo, V. Trianni, E. Şahin, R. Groß, T. H. Labella, G. Baldassarre, S. Nolfi, F. Mondada, J.-L. Deneubourg, F. Mondada, D. Floreano, L. Gambardella, *Evolving Self-Organization Behaviors for a Swarm-bot*. Autonomous Robots 17, 223-245, 2004.
- [12] C. R. Ward, F. Gobet, and G. Kendall, *Evolving Collective Behavior in an Artificial Ecology*. Special issue on Evolution of Sensors in Nature, Hardware and Simulation, Artificial Life 7(2), 191-209, 2001.
- [13] F. Mondada, G. C. Pettinaro, A. Guignard, I. W. Kwee, D. Floreano, J.-L. Deneubourg, S. Nolfi, L. M. Gambardella, and M. Dorigo, *Swarm-bot: A New Distributed Robotics Concept*. Autonomous Robots, vol. 17, no. 2-3, 193-221, September 2004.
- [14] J.-M. Valin, F. Michaud, J. Rouat, and D. Letourneau, *Robust Sound Source Localization Using a Microphone Array on a Mobile Robot*. In Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 1228-1233, vol.2, 2003.
- [15] O. Soysal, E. Bahçeci, and E. Şahin *PES: A System For Parallelized Fitness Evaluation of Evolutionary Methods*. In A. Yazici and C. Sener, editors, Lecture Notes in Computer Science 2869, pp. 889-896. Springer-Verlag, Heidelberg, Germany, 2003.