# Evolving Distributed Control for an Object Clustering Task

**Timothy D. Barfoot**[*]
*Controls and Analysis,*
*MDA Space Missions,*
*9445 Airport Road,*
*Brampton, Ontario L6S 4J3, Canada*

**Gabriele M. T. D'Eleuterio**
*Institute for Aerospace Studies,*
*University of Toronto,*
*4925 Dufferin Street,*
*Toronto, Ontario M3H 5T6, Canada*

Motivated by social insects, the possibility of evolving distributed control for a task requiring global coordination is investigated. The task is object clustering. A key aspect of this work is that a population of robot-like agents is allowed to select the cluster location. A detailed examination of how solutions evolved by a genetic algorithm are able to scale as key parameters are varied is presented, allowing commentary on the sensitivity of the evolved solution to changes in the environment. In most of the scaling experiments, the solution degrades gracefully about the evolutionary design point. However, in the case of constant-density scaling, the solution maintains its effectiveness as the problem is made larger.

## 1. Introduction

The phrase *multiagent system* encompasses large bodies of work from engineering, computer science, and mathematics. Examples include networks of mobile robots [1], software agents [2], and cellular automata (CA) [3]. A common thread in all multiagent systems is the issue of *coordination*. How are a large number of sparsely coupled agents able to produce a coherent global behavior using simple rules? Answering this question will not only permit the construction of interesting and useful artificial systems but may allow us to understand more about the natural world. Ants and the other social insects are examples of local interaction producing a coherent global behavior. It is possible for millions of ants to act as a superorganism through local pheromone communication [4].

[*]Work carried out while at the University of Toronto Institute for Aerospace Studies.

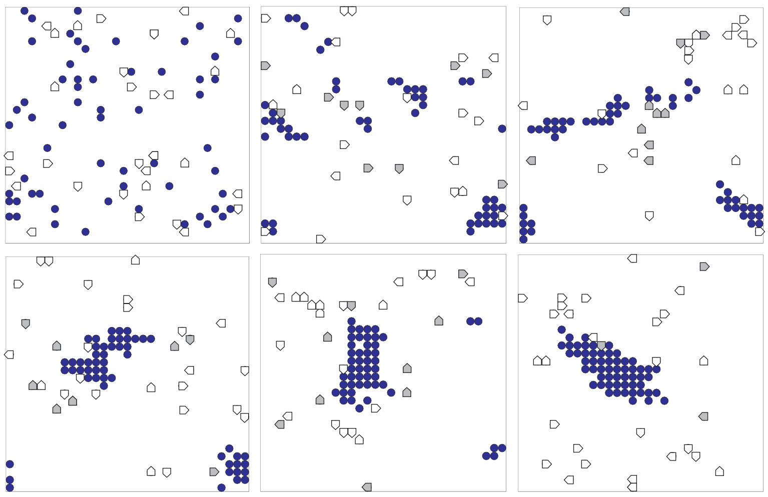


**Figure 1.** Preview of the object clustering task. Agents (pentagons) must cluster objects (dark circles) into a single heap (bottom right) from a random initial distribution (top left). The location of the heap is not predetermined in this self-organizing system.

Lewis Thomas [5] perhaps describes this phenomenon best:

> *A solitary ant, afield, cannot be considered to have much of anything on his mind. Four ants together, or ten, encircling a dead moth on a path, begin to look more like an idea. But it is only when you watch the dense mass of thousands of ants, blackening the ground that you begin to see the whole beast, and now you observe it thinking, planning, calculating. It is an intelligence, a kind of live computer, with crawling bits for its wits.*

We seek to reproduce this ability on a fundamental level in order to coordinate artificial systems.

It can be argued that CA are the simplest example of a multiagent system. Originally studied by von Neumann [6], CA were used to describe systems of sparsely coupled difference equations. Despite their simple mechanics, some extremely interesting behaviors have been cataloged, Conway's "The Game of Life" is one example. The term *self-organization* is used in many contexts when discussing multiagent systems, which can lead to confusion. Here we use it to mean "multiagent coordination in the face of more than one alternative." For example, in our task of object clustering (see Figure 1) we do not specify where the heap of objects should form but instead rely on self-organization.

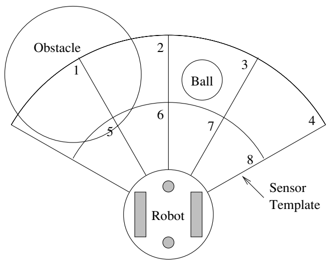
Researchers in collective robotics often use social insects to explain the motivation behind their work [7–10]. In the absence of a central controlling agent, colonies of ants are able to work together to the benefit of the society as a whole [4]. Each ant behaves according to its local situation yet interesting and coherent global behaviors result. No particular ant is essential to the overall dynamics. We hope to reproduce basic aspects of social insect behavior in a simple artificial system. This may be a key step to developing control methods for colonies of robots, for example. It is hoped that the benefits of such work indeed may be twofold in that it may guide the engineer in the design of distributed systems while also furthering our knowledge of how biological collectives achieve coordination.

In this paper, we investigate a problem in which a large collective of agents must come to a common decision. The goal is to have the collective choose a location in a grid world to create a single large cluster of objects [11]. We maintain that rules able to succeed at this task are self-organizing because the agents are not told where to form the pile, yet they must all coordinate their choices to produce a globally coherent decision. If we told the agents where to create the cluster, the task would be easier and no communication between the agents would be necessary. This is an example of *centralized organization* and is in stark contrast to *self- or decentralized organization*. We believe that coordination in the face of more than one alternative is a key aspect of multiagent systems [12].

Das [13] showed that genetic algorithms (GAs) are able to evolve CA that perform prescribed tasks requiring the type of global coordination in which we are interested. Motivated by this CA work, we used an evolutionary approach to learn rules that were successful at the object clustering task [11]. Our agents are different from CA as they are mobile and live in a grid-type environment. Once successful rules were found, we carried out extensive experiments to determine whether our solution would scale up (and down) to larger (and smaller) problem sizes. The results of this sensitivity analysis suggest that the densities of the agents and objects are key parameters of the system. The effectiveness of the solution is found to degrade when the densities are varied too far from the evolutionary design point.

## 2. Agent description

This section describes the nature of our robot-like agents. Each agent has a number of sensors and actuators. To relate this to real robots, it will be assumed that some transformation may be performed on raw sensor data so as to achieve a set of independent (sometimes called orthogonal [9]) virtual sensors that output a discrete value. Orthogonality can be divided into two types: spatial and modal. Spatially orthogonal sensors do not look at the same region of space. Modally orthogonal sensors do not look at the same type of datum (e.g., imagine two sensors, one only detects other agents and another which only detects obstacles),

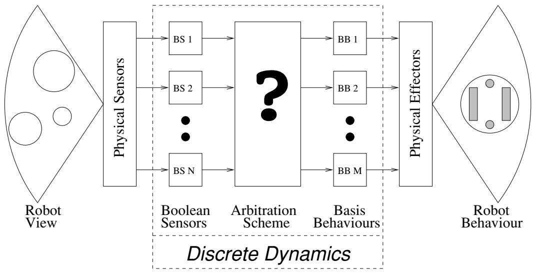


| Boolean Sensor Inputs | | |
|---|---|---|
| # | Obstacle | Ball |
| 1 | 1 | 0 |
| 2 | 1 | 0 |
| 3 | 0 | 1 |
| 4 | 0 | 0 |
| 5 | 0 | 0 |
| 6 | 0 | 0 |
| 7 | 0 | 0 |
| 8 | 0 | 0 |

Spatially Orthogonal

Modally Orthogonal

(a)



(b)

**Figure 2.** (a) A robot-like sensor array. (b) Schematic of control scheme. Internal dynamics are discrete.

Figure 2(a) provides a visualization. The transformation is essentially a preprocessing step that reduces the raw sensor data to more readily usable discretized inputs.

Let us further assume that the output of our control system may be discrete. This may be done by way of a set of *basis behaviors* [14]. Rather than specify the actuator positions (or velocities), we assume that we may select a simple behavior from a finite predefined palette. This may be considered a postprocessing step that takes a discretized output and converts it to the actual actuator control. Neither the preprocessing nor the postprocessing steps will be allowed to change once set. The actual construction of these transformations requires careful consideration but is also somewhat arbitrary, as will be seen. Figure 2(b) shows what the control system in each robot-like agent is beginning to look like. The internal dynamics may now be considered as entirely discrete.

Once all the pre/postprocessing has been set up, the challenge remains to find an appropriate arbitration scheme that takes in a discrete input sequence (size $N$) and outputs the appropriate discrete output (one of $M$ basis behaviors). There are several candidates for this role but the one affording the most general decision surfaces between input and output is a straightforward lookup table similar to CA. In various papers [13, 15–17], it has been shown that GAs are able to evolve CA that perform prescribed tasks requiring global coordination. This is essentially what we wish to achieve, but have the added difficulty of dealing with the environment of our robot-like agents.

This type of lookup table control in autonomous robots is often called *reactive*. For every possible input sequence, the CA scheme stores a discrete output value. In other words, for every possible input view there is an output corresponding to one of the basis behaviors. At each time step, the agent looks up the action that corresponds to its current view and executes it. The size of the lookup table (for binary sensors) will be $2^N$ so if there are too many input sensors, the CA lookup table will be very large; the approach is therefore usually rendered feasible for only modest numbers. Although the number of basis behaviors $M$ does not directly affect the size of the CA lookup table, it does affect the number of all such possible tables, which is $M^{2^N}$ for binary sensors. Again, modest numbers of basis behaviors keep the size of the search space reasonable.

The crucial step in this whole approach is the discovery of particular CA lookup tables that cause a collection of identical agents to succeed at a task requiring global coordination. The obvious first method to attempt is to design the local rules by hand. How hard can it be? It turns out to be very difficult to do for all but the most trivial examples. Even when working with simple one-dimensional CA models (a far cry from a fleet of robot-like agents) there can be millions if not billions of sets of local rules from which to choose. We are faced with a combinatorial explosion. To combat this, we employed an evolutionary optimization process to search for good solutions.

A GA is a global-optimization technique loosely based on biological evolution [18]. A random initial population of $P$ CA lookup tables is evolved over $G$ generations. Each CA lookup table $\phi$ has a *chromosome* that consists of a sequence of all the discrete values taken from the table. In our implementation, a fitness is assigned to each CA lookup table at each generation (based on how well a collection of agents, each containing the CA lookup table, conforms to our prescribed behavior [19]). A CA lookup table's fitness determines its representation in the next generation. Genetic crossovers and mutations introduce new CA lookup tables into the population. The best $K \leq P$ CA lookup tables are copied exactly from one generation to the next. The remaining $(P - K)$ CA lookup tables are subjected to a single site crossover at a random location with probability $p_c$. Furthermore, they are subjected to random site mutations with probability $p_m$ per site.

## 3. Object clustering task

Sometimes called the shepherding or heap-formation task, *object clustering* has an established history in the literature and is directly comparable to the behavior of some insect societies [7, 8, 20]. It is believed that this task requires global coordination for a group of agents, existing in a two-dimensional space, to move some initially randomly distributed objects into a single large cluster. However, there is no central controlling agent that says where to put the cluster. The agents must come to a common decision among themselves without any external help (analogous to the global partitioning task in CA work [16]). The absence of any central controlling agent or goal beacon makes this a difficult computation to be performed by this spatially-extended system.

Traditional GAs require a fitness function to be defined (on which selection is based). For the heap-formation problem, the physical space in which the agents exist is broken into $J$ cells $A_j$ as depicted in Figure 3(b) and the fitness function is defined to be

$$f_{\text{total}} = \frac{\sum_{i=1}^{I} f_i}{I} \tag{1}$$

where $I$ is the number of random initial conditions over which $f_i$ is averaged. $f_i$ is the fitness on one initial condition only, given by

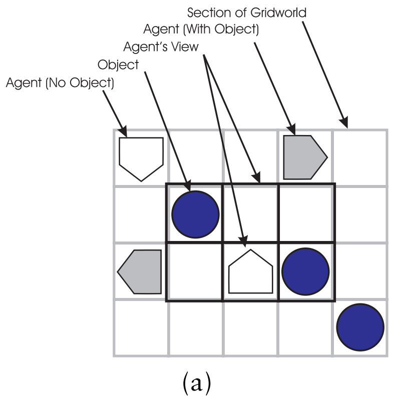
$$f_i = 1.0 + \frac{\sum_{j=1}^{J} p_j \ln p_j}{\ln J} \tag{2}$$

where $p_j = n(A_j)/\sum_{j=1}^{J} n(A_j)$ and $n(A_j)$ is the number of objects in cell $A_j$. This is a modified Shannon entropy [21] function that is 0 when the objects are equally distributed over all cells, and 1 when all the objects are in a single cell.

To summarize, fitness is assigned to a CA lookup table by equipping each agent in a collective with that CA lookup table. The collective is allowed to roam around in a two-dimensional space that has a random initial distribution of objects. At the end of $T$ time steps, $f_i$ is calculated, which indicates how well the objects are clustered. This is all repeated $I$ times to allow some statistical averaging and $f_{\text{total}}$, the fitness of the CA lookup table, is determined.

## 4. Simulation results

The space in which the agents roam will be a two-dimensional lattice with square cells and periodic boundary conditions in both spatial directions (i.e., the surface of a torus). Note, if we did not use periodic boundaries the object clustering problem is much easier. For example, the agents could learn to simply cluster objects in a corner. This would



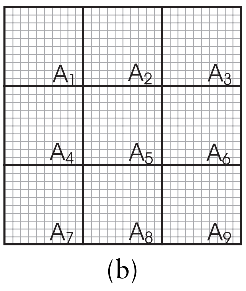


**Figure 3.** (a) Typical view of a software agent. Each agent sees only six squares (local information). Dark circles are objects. The pentagons (with the point indicating orientation) is the agent itself. (b) Partition of grid world into bins for fitness calculation.

not demonstrate the kind of global coordination we are after. Also note, with periodic boundary conditions there is no "edge" to the world. To ensure that all of the objects will be in a single fitness cell if they are well clustered (as opposed to being divided across two cells), we move the fitness grid in Figure 3(b) about on the surface of the torus, in order to determine the best fitness value for a particular configuration of objects. This is important as we are allowing the agents to select where to cluster the objects and do not want to penalize them for placing the cluster on the boundary of two fitness cells.

Each agent will be able to see only six squares as depicted in Figure 3(a). Each agent is able to carry one object at a time and can tell whether or not it possesses an object. In five of the squares an agent sees there are three possibilities (nothing, object, agent); in the sixth square, the one occupied by the agent, there are two possibilities (carrying or not carrying an object). This means there are $3^5 \times 2 = 486$ entries in the CA lookup table. Only two basis behaviors will be defined for this task.

- **Move self.** The agent moves forward if the cell in front is empty; otherwise turns left if the cell to the left is empty; otherwise turns right.
- **Manipulate object.** The agent picks up/puts down an object from directly ahead (if possible); otherwise picks up/puts down an object left of center (if possible); otherwise picks up/puts down an object right of center (if possible); otherwise activates the **Move self** basis behavior. (Whether the agent is picking up or putting down an object depends on whether it already has one in its possession or not.)

The choice of these behavior modules is arbitrary yet natural. With two basis behaviors and a CA lookup table of size 486 there are $2^{486} \approx 10^{146}$

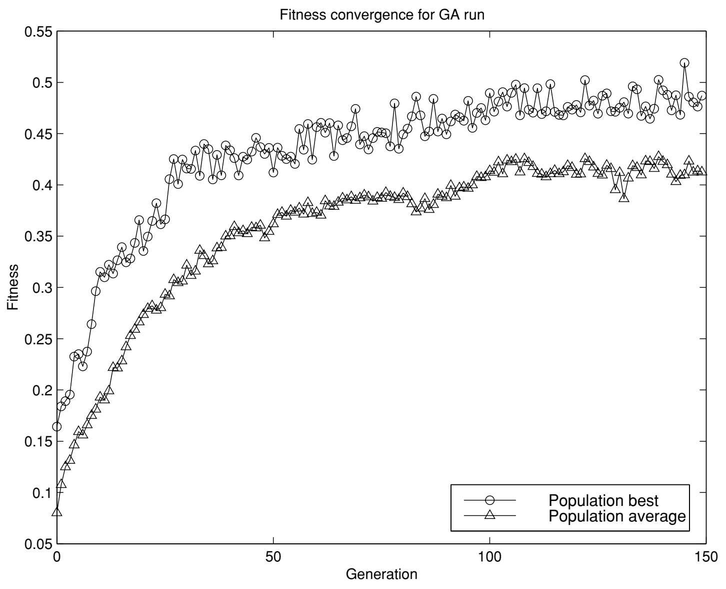


**Figure 4**. Convergence history ($f_{\text{total}}$ vs. generation) of a typical GA run. The population size was 50 for this case.

possible CA lookup tables. This number is quite large but, as we will see, good solutions can still be found. It should be pointed out that our agents will be functioning in a completely deterministic manner. From a particular initial condition, the system will always unfold in the same particular way.

In our experiments, we have used a GA population size of $P = 50$, number of generations $G = 150$, keepsize $K = 5$, crossover probability $p_c = 0.6$, and mutation probability $p_m = 0.005$. For the purposes of finding good rules we have used a two-dimensional world of size $31 \times 30$, 30 agents, 60 objects, a training time of $T = 2000$ time steps, number of areas involved in the fitness calculation $J = 9$, and number of random initial conditions per fitness evaluation $I = 30$. Figure 4 shows a typical convergence history of the GA.

At the end of 150 generations, we take the best CA lookup table in the GA population to be our solution. Figure 1 shows some snapshots of one solution, dubbed $\phi_{\text{heap}}$, in action. Starting from an initial random distribution of objects, the agents start by forming little piles that they eventually merge into one large cluster. This strategy is similar, for example, to that employed by *Pheidole pallidula* (ants) in the clustering of corpses or *Leptothorax unifasciatus* (also ants) in the clustering of larvae [8].

If one plots the fitness time series of the system starting from a random initial condition, a curve like the plot in Figure 5(a) is the result. However, if one averages the fitness time series over say 1000 initial conditions, the much smoother curve in the plot of Figure 5(b) results. Based on this average time series, an emergence time [22] may be calculated which for us represents how long the system takes, on average, to go to a steady state fitness (0.95 of maximum fitness). For the parameters used to evolve the $\phi_{\text{heap}}$ solution ($31 \times 30$ world, 30 agents, 60 objects), it takes 5200 time steps for the system to self-organize.

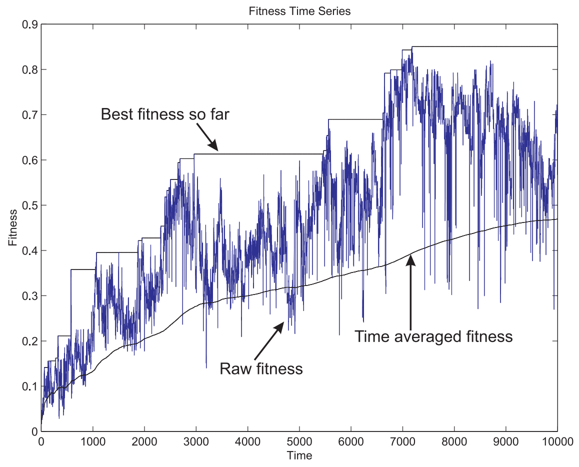
## 5. Scaling the solution

A very important issue for multiagent systems is that of *scaling*. We evolved the solution $\phi_{\text{heap}}$ with one particular set of parameters ($31 \times 30$ world, 30 agents, 60 objects). Scaling refers to changing the size of the problem under investigation while keeping the CA lookup table the same. We can change the size of this problem in the following ways.

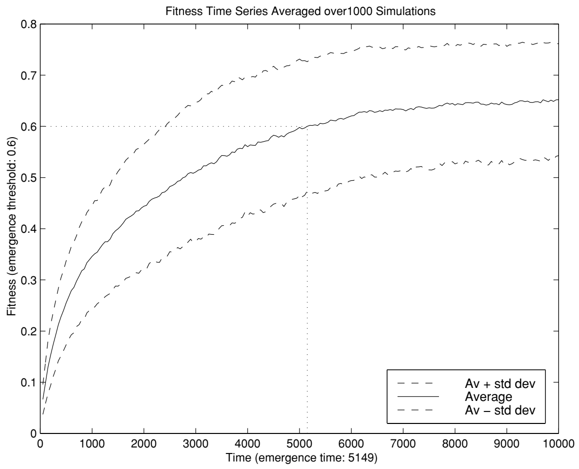
- Vary the number of agents in the simulation while keeping all other parameters the same.
- Vary the number of objects in the simulation while keeping all other parameters the same.
- Vary the size of the world while keeping all other parameters the same.
- Vary the number of agents, number of objects, and size of the world in unison so that agent density and object density remain constant.

Figure 6 shows some snapshots of the $\phi_{\text{heap}}$ solution in action with a $91 \times 90$ world, 270 agents, and 540 objects, which is an example of constant density scaling. Again, smaller piles of objects are formed first and eventually merged into a single large heap. Obviously it takes much longer for this larger system to form a single heap. This is not unreasonable especially if one takes the view that the collective is performing a computation; more complicated computations should take longer. A more detailed study of constant density scaling may be found below.

In the following subsections, a systematic investigation of scaling is presented. Each of these scaling experiments can tell us important things about the robustness of the solution and aid in the application of such methods to real problems. For example, we may want to evolve good solutions on a reduced problem size (in order to save time) and then predict how well our solution will perform once it is "scaled up" to the actual problem size. In another scenario we might have a very hazardous work environment, which means we would like to be able to know how many agents can cease to function such that the problem is still solved in a timely manner (if at all).



(a)



(b)

**Figure 5.** (a) Typical fitness time series ($f_i$ vs. time) for a $31 \times 30$ world with 30 agents and 60 objects. Best so far and time averaged fitness are shown for interest only and are not used in any calculations. (b) Time series resulting from an average over $I = 1000$ simulations with different random initial conditions. The value of the bottom plot at the maximum time (e.g., 10000) is the $f_{\text{total}}$ that is used as the fitness in the GA optimization. An emergence time is calculated.

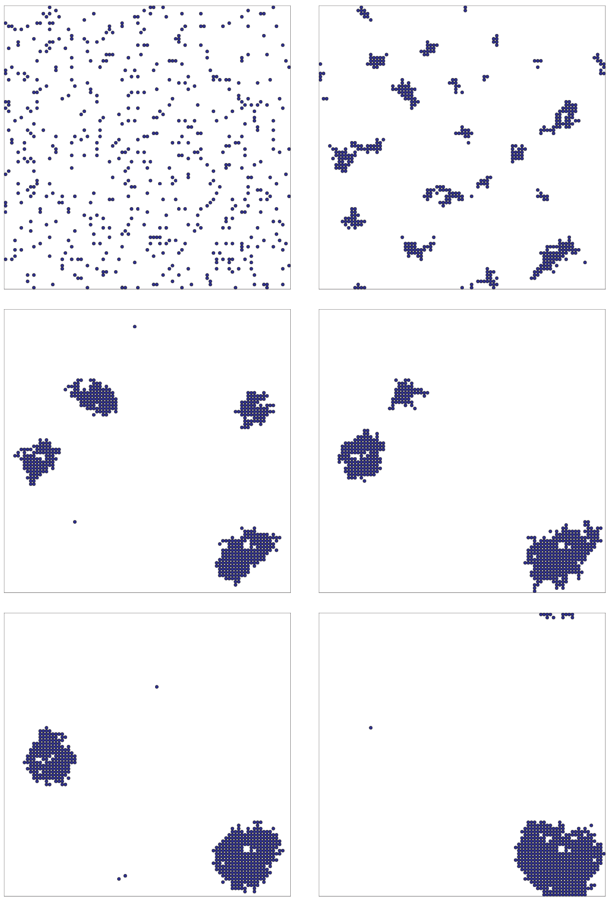
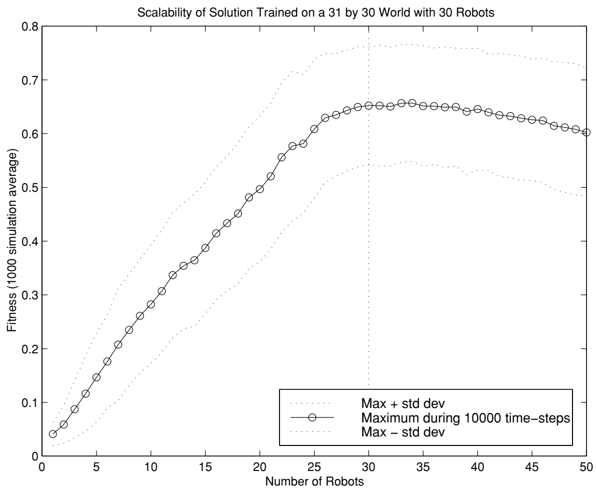


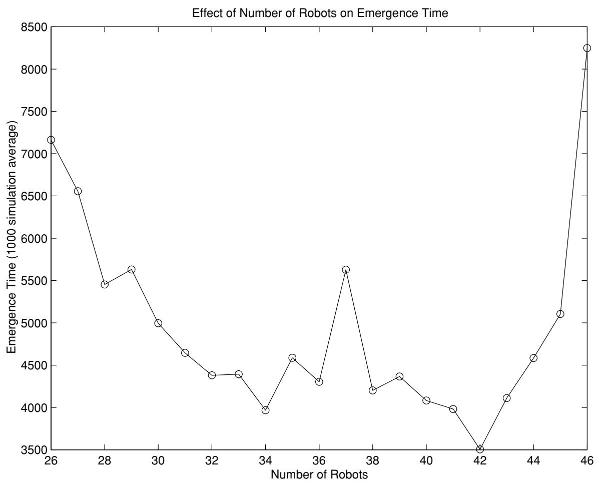
**Figure 6.** Typical snapshots of system at various times (0, 1010, 6778, 14924, 20153, 58006). The world size is $91 \times 90$, there are 270 agents, and 540 objects. Only the objects (dark circles) are shown for clarity. The gradual merging of objects into larger and larger piles is similar to techniques observed in some species of ant.

## 5.1 Scaling the number of agents

For a fixed world size ($31 \times 30$) and number of objects (60), what is the effect on varying the number of agents in a simulation? This is an important question as we would like to know how many agents to use for a certain task. Figure 7 shows some maximum fitness values and emergence times for different numbers of agents. The datum at 37 agents in Figure 7(b) deviates from the general trend owing to a finite sample size (i.e., only 1000 simulations from random initial conditions were averaged). It should be stressed that the solution under investigation



(a)



(b)

**Figure 7.** (a) The effect on the maximum fitness and (b) emergence time when the number of agents is changed (the rest of the problem stays the same).

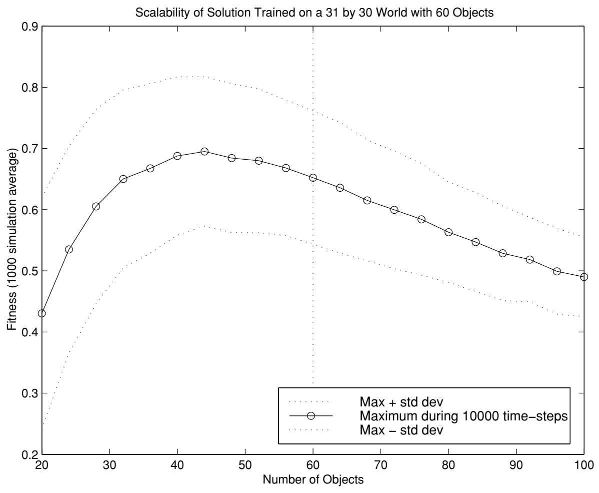
here was evolved based on 30 agents exactly. It is not surprising then that the highest fitness values correspond to numbers of agents near 30. Too few agents and the problem does not get adequately solved (underpopulated). Too many and the agents begin to hinder the progress of one another (sometimes called *antagonism* [7]). Clearly, antagonism is not as strong an effect as underpopulation in this system.

## 5.2 Scaling the number of objects

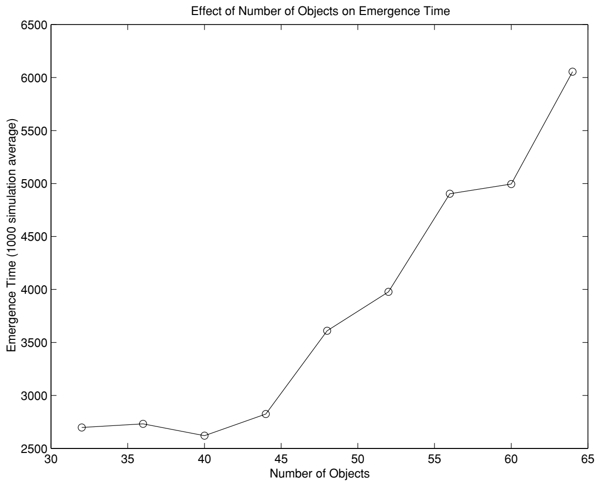
For a fixed world size ($31 \times 30$) and number of agents (30), what is the effect on varying the number of objects in a simulation? We would like to answer this in order to know how many objects the agents are able to successfully handle. Figure 8 shows some maximum fitness values and emergence times for different numbers of objects. It is again important to state that the solution under investigation here was evolved using exactly 60 objects. Here we are seeing some interesting results. If there are too few objects in the system, the performance of the agents drops off rather quickly as seen in Figure 8. This can be explained by the fact that to move objects, the agents activate the **Manipulate object** module which means they must "pick up" and carry them. If there are too few objects compared to the number of agents it is conceivable that all the objects are being carried which makes it difficult for piles to get started. Presumably one would not use more agents than the number of objects in this system. Less surprisingly, as the number of objects becomes larger it becomes difficult for the agents to adequately organize the system (in 10000 time steps). Intuitively this makes a great deal of sense, especially if one believes that computation and complex systems are intimately entangled. It should take longer to perform a more complicated computation (see Figure 8(b) for the effect on emergence time).

## 5.3 Scaling the size of the world

For a fixed number of objects (60) and number of agents (30), what is the effect on varying the size of the world? Figure 9 shows what happens to the maximum fitness during a 10000 time-step simulation as world size varies (all these graphs are based on 1000 simulation averages). Note that the solution under investigation was trained on a $31 \times 30$ world so it is not surprising that, at first glance, the agents did best on this size of world while doing poorly on both smaller and larger ones. However, Figure 9 is not perhaps as bad as it looks. When the world is smaller than the training size of 30, each area $A_i$ is also smaller (with $J$ fixed) so that in fact with the same number of objects, the maximum fitness achievable is not as high. What really should be done is to allow the number of areas involved in the fitness calculation $J$ to change with the world size. This would give a better comparison. When the world is larger, there are two effects to consider. First, in a larger world one might expect the agents (that move at finite speed) to take longer to move between piles of objects. Here we are only allowing them to work for 10000 time steps so we should expect to see some drop in their performance as the world size increases. One might then think that if given enough time, the agents should ultimately do as well as on smaller worlds. Second, however, the density of agents decreases as the world



(a)



(b)

**Figure 8.** (a) The effect on the maximum fitness and (b) emergence time when the number of objects is changed (the rest of the problem stays the same).

size increases. This means that there should be fewer collisions between agents. This lack of collisions has a tendency to allow agents to fall into fixed loop patterns in which they remain stuck for all subsequent times. Because a short simulation time was used, it is difficult to distinguish between the former and latter effects. In the next experiment, longer times will be allowed in order to make such a distinction. A plot of emergence time was not available for this experiment.

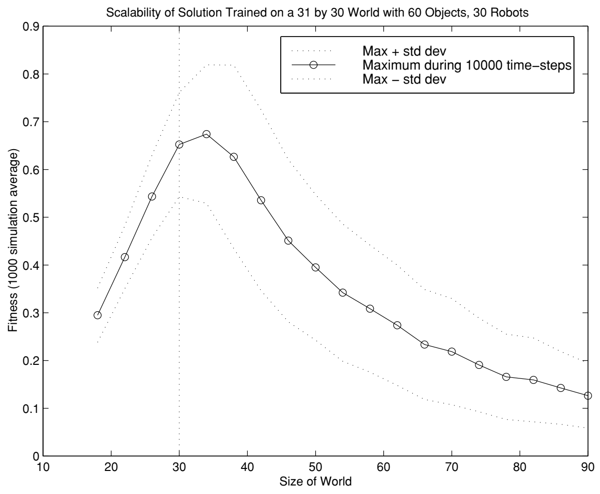
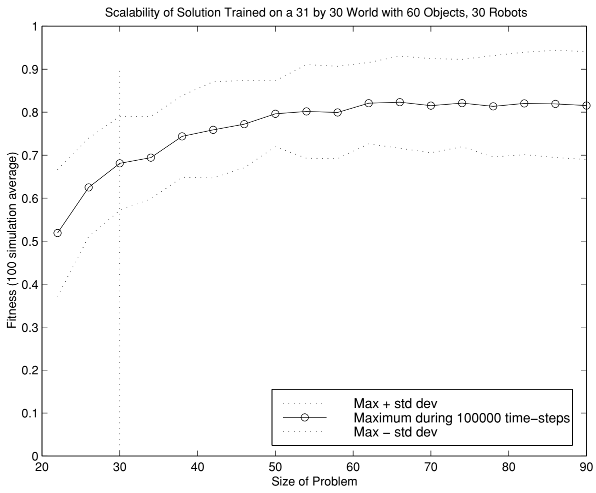


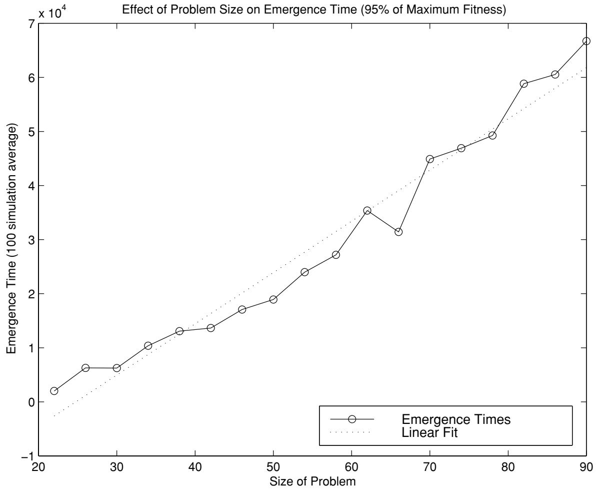
**Figure 9**. The effect on maximum fitness when the size of the world is changed (the rest of the problem stays the same). Note that the "Size of World" axis label refers to the $y$ world dimension and the $x$ dimension is this number augmented by 1 to avoid cyclic behaviors.

## 5.4 Constant density scaling

Perhaps the most natural way to scale such problems would be to change the world size, number of agents, and number of objects simultaneously such that the densities of agents and objects in the world remain constant. We will use the training densities, which were 1 agent per 30 squares and 1 object per 15 squares. It also turns out that the issue with the fitness function and number of areas $J$ disappears for constant density scaling ($J = 9$ areas will be used throughout) since the density of agents and objects within any such area should also remain constant. Figure 10 shows how the maximum fitness and emergence time fare over an extended 100000 time-step run. The "problem size" will be taken as the size of the $y$ dimension of the world (as in the previous subsection but this time the agent and object densities are constant). The problem size may be thought of as a characteristic length. The maximum fitness curve remains very high even for the largest case ($91 \times 90$) which is almost an order of magnitude larger than the training case in terms of number of agents and objects. It is interesting to note that there is a fairly linear relationship between the problem size and the emergence time.



(a)



(b)

**Figure 10.** (a) Maximum fitness and (b) emergence time during an extended 100000 time-step set of 100 simulations.

## 6. Conclusions

It is quite encouraging that our method of evolving global behaviors in a group of robot-like agents has been successful at the object clustering task. The approach is perhaps not immediately applicable to real-world engineering projects but there is a growing belief that decentralized, self-organizing systems will have their place in future technologies. They might be used in environmental cleanups, surface explorations, or mining. Decentralized control could be very useful in nanotechnology. At the very least this experiment helps further the notion that these types of systems must be designed with the whole system (agents plus environment) in mind. Each agent has no idea that it is helping to cluster objects, it just follows the rules laid out for it. In fact, it would be very difficult to predict just what would happen to the objects by analyzing a single agent. However, it has been shown here that it is possible to evolve successful behaviors on a reduced problem size and then scale up the solution to the desired full problem size. From our detailed study of scaling for this system we can see that the $\phi_{\text{heap}}$ solution degrades gracefully around the design point of 30 agents, 60 objects, and the $31 \times 30$ grid size. It is also important that by keeping the density of agents and objects constant as the grid size is increased did not cause the solution to degrade. It naturally took longer for the system to self-organize for larger problems but this is to be expected if we interpret object clustering as a computation [23]. It is also quite intriguing that the solutions we found to the object clustering task exhibit behavior that is quite similar to clustering behavior of ants. First small piles are formed, then larger ones, and this continues until there is just one pile. It is not difficult to see that this type of solution should scale nicely as the problem size is increased.

Although the grid-world model presented here is more realistic than cellular automata as an approach to collective robotics, the added complexity of the system makes its analysis quite difficult. We have shown that, on average, our solution $\phi_{\text{heap}}$ produces the desired system behavior after a certain emergence time. This is nice from an engineering perspective but does not further our general understanding of multiagent systems. Are there some basic mechanisms of self-organizing systems [12]? Even though the system presented here is barely complex enough to do anything useful, it is almost too complex to help us answer this question. We are certainly in need of a mathematical framework in which to analyze these types of systems. This may help us to give something back to our understanding of natural and biological systems whence our inspiration came. Computational mechanics [22, 24], statistical mechanics, and information theoretic approaches to multiagent systems are all candidates for this framework.

## Acknowledgments

The research was funded in part by the Centre for Research in Earth and Space Technology and the Natural Sciences and Engineering Research Council of Canada.

## References

[1] T. D. Barfoot, E. J. P. Earon, and G. M. T. D'Eleuterio, "A New Breed: Development of a Network of Mobile Robots for Space Exploration," in *Proceedings of the Sixth International Symposium on Artificial Intelligence, Robotics and Automation in Space (iSAIRAS)*, Montréal, Canada, June 19–21, 2001.

[2] Eric Bonabeau, Guy Theraulaz, Eric Arpin, and Emmanual Sardet, "The Building Behaviour of Lattice Swarms," in *Artificial Life IV: Proceedings of the Fourth International Workshop on the Synthesis and Simulation of Living Systems*, edited by Rodney A. Brooks and Pattie Maes (MIT Press, 1994).

[3] Stephen Wolfram, "Universality and Complexity in Cellular Automata," *Physica D*, **10** (1984) 1–35.

[4] Erick Hoyt, *The Earth Dwellers: Adventures in the Land of Ants* (Simon and Schuster, New York, 1996).

[5] Lewis Thomas, *The Lives of a Cell* (Viking Press, New York, 1974).

[6] Jon von Neumann, *Theory of Self-Reproducing Automata* (University of Illinois Press, Urbana and London, 1966).

[7] Thierry Dagaeff, Fabrice Chantemargue, and Beat Hirsbrunner, "Emergence-based Cooperation in a Multi-agent System," Technical report, University of Fribourg, Computer Science Department, PAI Group, 1997.

[8] J. L. Deneubourg, S. Goss, N. Franks, A. Sendova-Franks, C. Detrain, and L. Chrétien, "The Dynamics of Collective Sorting: Robot-like Ants and Ant-like Robots," in *Simulation of Adaptive Behaviour: From Animals to Animats*, edited by J. A. Meyer and S. Wilson (MIT Press, 1990).

[9] Ronald C. Kube and Hong Zhang, "The Use of Perceptual Cues in Multi-robot Box-pushing," in *Proceedings IEEE International Conference on Robotics and Automation*, 1996.

[10] G. Theraulaz, S. Goss, J. Gervet, and J. L. Deneubourg, "Task Differentiation in Polistes Wasp Colonies: A Model for Self-organizing Groups of Robots," in *Simulation of Adaptive Behaviour: From Animals to Animats*, edited by J. A. Meyer and S. Wilson (MIT Press, 1990).

[11] T. D. Barfoot and G. M. T. D'Eleuterio, "An Evolutionary Approach to Multiagent Heap Formation," in *Proceedings of the Congress on Evolutionary Computation*, Washington DC, USA, July 6–9 1999.

[12] T. D. Barfoot and G. M. T. D'Eleuterio, "Stochastic Self-organization," *Submitted to Complex Systems*, 2004.

[13] Rajarshi Das, James P. Crutchfield, Melanie Mitchell, and James E. Hanson, "Evolving Globally Synchronized Cellular Automata," in *Proceedings of the Sixth International Conference on Genetic Algorithms*, edited by L. J. Eshelman, San Fransisco, CA, April, 1995.

[14] Maja J. Matarić, "Behaviour-based Control: Examples from Navigation, Learning, and Group Behaviour," *Journal of Experimental and Theoretical Artificial Intelligence*, **9**(2) (1997) 232–336. Special Issue on Software Architectures for Physical Agents, edited by H. Hexmoor, I. Horswill, and D. Kortenkamp.

[15] Melanie Mitchell, Peter T. Hraber, and James P. Crutchfield, "Revisiting the Edge of Chaos: Evolving Cellular Automata to Perform Computations," *Complex Systems*, 7 (1993) 89–130. Santa Fe Institute Working Paper 93-03-014.

[16] Melanie Mitchell, James P. Crutchfield, and Rajarshi Das, "Evolving Cellular Automata with Genetic Algorithms: A Review of Recent Work," in *Proceedings of the First International Conference on Evolutionary Computation and Its Applications*, Moscow, Russia, 1996. Russian Academy of Sciences.

[17] Win Hordijk, James P. Crutchfield, and Melanie Mitchell, "Mechanisms of Emergent Computation in Cellular Automata," Working Paper 98-04-034, Santa Fe Institute, 1998. In *Fifth International Conference on Parallel Problem Solving from Nature PPSN-V*, edited by A. E. Eiben (Springer, New York).

[18] David E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning* (Addison-Wesley, Reading, MA, 1989).

[19] Melanie Mitchell and Stephanie Forrest, "Genetic Algorithms and Artificial Life," in *Artificial Life: An Overview*, edited by Chris G. Langton (MIT Press, 1995). Santa Fe Institute Working Paper 93-11-072.

[20] Marinus Maris and René te Boekhorst, "Exploiting Physical Constraints: Heap Formation Through Behavioural Error in a Group of Robots," in *Proceedings IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Senri Life Science Center, Osaka, Japan, November 4–8, 1996.

[21] C. E. Shannon, "A Mathematical Theory of Communication," *The Bell System Technical Journal*, **27** (1948) 379–423.

[22] James E. Hanson and James P. Crutchfield, "Computational Mechanics of Cellular Automata: An Example," Working Paper 95-10-095, Santa Fe Institute, 1995. *Physica D*, **103** (1997) 169–189; Proceedings of the International Workshop on Lattice Dynamics.

[23] Stephen Wolfram, *A New Kind of Science* (Wolfram Media, Inc., Champaign, IL, 2002).

[24] David P. Feldman and James P. Crutchfield, "Discovering Noncritical Organization: Statistical Mechanical, Information Theoretic, and Computational Views of Patterns in One-dimensional Spin Systems," Working Paper 98-04-026, Santa Fe Institute, 1998.