3

Evolutionary-Based Control Approaches for Multirobot Systems

Jekanthan Thangavelautham, Timothy D. Barfoot and Gabriele M.T. D'Eleuterio
Institute for Aerospace Studies, University of Toronto
Canada

1. Introduction

In this chapter, we summarize and synthesize our investigations of the use of evolutionary algorithms to automatically program robots, particularly for application to space exploration. In the Space Robotics Group at the University of Toronto Institute for Aerospace Studies, we were motivated to begin work in this area a decade ago when the concept of *network science* became popular in the space exploration community. Network science commonly refers to science that requires a distribution of possibly simultaneous measurement devices or a distribution of platforms on a planetary body. Consider, for example, seismology studies of an alien body that will require sending a signal from one point on the surface to be read at several other points in order to analyze the material characteristics of the body. Or, consider the development of a very-low frequency array (VLFA) on the Moon to allow for heretofore unattainable astrophysical observations using radio astronomy. Such an observatory will require a number of dipole units deployed over a region of a few hundred square kilometres.

Our original thoughts were that these and other network science experiments could be implemented using a network of small mobile robots, similar to a colony of ants. It is possible for millions of ants to act as a superorganism through local pheromone communication. Thomas (1974) 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 set out to reproduce this type of behaviour in a *multirobot system* (i.e., a network of mobile robots) for application to space exploration.

As we began investigating how to devise control schemes for the network science tasks, additional applications came to mind including deployment of solar cell arrays on a planetary surface, site preparation for a lunar base, and gathering objects of interest for analysis or in-situ resource utilization. Although these additional tasks did not necessitate

Source: Frontiers in Evolutionary Robotics, Book edited by: Hitoshi Iba, ISBN 978-3-902613-19-6, pp. 596, April 2008, I-Tech Education and Publishing, Vienna, Austria

the use of a group of robots, there are certain advantages offered by this choice. Redundancy and fault tolerance are fundamental attributes of any reliable space system. By using a group of robots, we might afford to lose a small number of individuals and yet still accomplish our desired task. The flip side to redundancy is taking risks. By making the system modular and thus redundant, we could be willing to accept more risk in the design of a single robot because it no longer has the potential to produce a single point failure.

In many of our early experiments, we tried designing controllers for our groups of robots by hand (Earon et al., 2001). This was possible for some simple tasks, such as having the robots position themselves in a particular geometric formation. As we became interested in the resource collection and array deployment tasks, the burden of manual programming became higher and we turned to the use of evolutionary algorithms.

Moreover, we wondered if it would be possible to specify the required task at the group level and have the evolutionary algorithm find the best way to coordinate the robots to accomplish the overall goal. This notion of top-down performance specification is very much in keeping with the formal approach to space engineering, in which mission-level goals are provided and then broken down to manageable pieces by a designer. Accordingly, this notion of *task decomposition* is at the heart of our discussion throughout this chapter. We will advocate for an approach that does not explicitly break a task down into subtasks for individual robots, but rather facilities this through careful selection of the evaluation criteria used to gauge group behaviour on a particular task (i.e., the *fitness function*). By using an evolutionary algorithm with this judiciously chosen fitness function, task decomposition occurs through *emergence* (self-organization).

The remainder of this chapter is organized as follows. First, we review the literature on the use of evolutionary algorithms for task decomposition and development of multirobot controllers. Next we report on a number of approaches we have investigated to control and coordinate groups of robots. Our discussion is framed in the context of four tasks motivated by space exploration: *heap formation* (Barfoot & D'Eleuterio, 2005), *tiling pattern formation* (Thangavelautham & D'Eleuterio, 2004), *a walking robot* (Barfoot et al., 2006) (wherein each leg can be thought of a single robot), and *resource gathering (*Thangavelautham et al., 2007). This is followed by a discussion of the common findings across these experiments and finally we make some concluding remarks.

2. Background

Task decomposition involves partitioning/segmenting of a complex task into subtasks. The partitioning is expected to facilitate solving the simpler subtasks which in turn are combined to yield a solution to the overall task. One approach to role assignment and execution of the subtasks is through use of multirobot systems. Multirobot systems offer the security of redundancy, fault tolerance and, depending on the task, scalability. They furthermore allow for the parallelization of operations. With the use of multiple agents or robots, their control and coordination are critical.

In nature, multiagent systems such as social insects use a number of mechanisms for control and coordination. These include the use of *templates*, *stigmergy*, and *self-organization*. *Templates* are environmental features perceptible to the individuals within the collective (Bonabeau et al., 1999). *Stigmergy* is a form of indirect communication mediated through the environment (Grassé, 1959). In insect colonies, templates may be a natural phenomenon or they may be created by the colonies themselves. They may include temperature,

humidity, chemical, or light gradients. In the natural world, one way in which ants and termites exploit stigmergy is through the use of pheromone trails. *Self-organization* describes how local or microscopic behaviours give rise to a macroscopic structure in systems (Bonabeau et al., 1997). However, many existing approaches suffer from another emergent feature called *antagonism* (Chantemargue et al., 1996). This describes the effect that arises when multiple agents that are trying to perform the same task interfere and reduce the overall efficiency of the group.

Within the field of robotics, many have sought to develop multirobot control and coordination behaviours based on one or more of the prescribed mechanisms used in nature. These solutions have been developed using user-defined deterministic 'if-then' or preprogrammed stochastic behaviours. Such techniques in robotics include template-based approaches that exploit light fields to direct the creation of circular walls (Stewart and Russell, 2003), linear walls (Wawerla et al., 2002) and planar annulus structures (Wilson et al., 2004). Stigmergy has been used extensively in collective-robotic construction tasks, including blind bull dozing (Parker et al., 2003), box pushing (Matarić et al., 1995) and heap formation (Beckers et al., 1994).

Inspired by insect societies the robot controllers are often designed to be reactive and have access only to local information. They are nevertheless able to self-organize, through cooperation to achieve an overall objective. This is difficult to do by hand, since the global effect of these local interactions is often hard to predict. The simplest hand-coded techniques have involved designing a controller for a single robot and scaling to multiple units by treating other units as obstacles to be avoided (Parker et al., 2003) (Stewart & Russell, 2003), (Beckers et al., 1994). Other more sophisticated techniques involve use of explicit communication or designing an extra set of coordination rules to handle graceful agent-to-agent interactions (Wawerla et al., 2002). These approaches are largely heuristic and rely on ad hoc assumptions that often require knowledge of the task domain.

In contrast, machine learning techniques (particularly artificial evolution) exploit self-organization and relieve the designer of the need to determine a suitable control strategy. The controllers in turn are designed from the start with cooperation and interaction, as a product of emergent interactions with the environment. It is more difficult to design controllers by hand with cooperation in mind because it is difficult to predict or control the global behaviours that will result from local interactions. Designing successful controllers by hand can devolve into a process of trial and error.

A means of reducing the effort required in designing controllers by hand is to encode controllers as Cellular Automata (CA) look-up tables and allow a genetic algorithm to evolve the table entries (Das et al., 1995). The assumption is that each combination of discretized sensory inputs will result in an independent choice of discretized output behaviours. This approach is an instance of a 'tabula rasa' technique, whereby a control system starts off with a blank slate with limited assumptions regarding control architecture and is guided through training by a fitness function (system goal function).

As we show in this chapter, this approach is used successfully to solve a multiagent heap formation task (Barfoot & D'Eleuterio, 1999) and a 2 × 2 tiling formation task (Thangavelautham et al., 2003). Robust decentralized controllers that exploit stigmergy and self-organization are found to be scalable to 'world size' and to agent density. Look-up table approaches are also beneficial for hardware experiments where there is minimal computational overhead incurred as a result of sensory processing.

We also wish to analyze the scalability of evolutionary techniques to bigger problem spaces. One of the limitations with a look-up table approach is that the table size grows exponentially to the number of inputs. For the 3 × 3 tiling formation task, a monolithic look-up table architecture is found to be intractable due to premature search stagnation. To address this limitation, the controller is modularized into 'subsystems' that have the ability to explicitly communicate and coordinate actions with other agents (Thangavelautham et al., 2003). This act of dividing the agent functionality into subsystems is a form of user-assisted task decomposition through modularization. Although the technique uses a global fitness function, such design intervention requires domain knowledge of the task and ad hoc design choices to facilitate searching for a solution.

Alternatively, CA lookup tables could be networked to exploit inherent modularity in a physical system during evolution, such as series of locally coupled leg controllers for a hexapod robot (Earon et al., 2000). This is in contrast to some predefined recurrent neural network solutions such as by (Beer & Gallagher, 1992), (Parker & Li, 2003) that are used to evolve 'leg cycles' and gait coordination in two separate stages. This act of performing staged evolution involves a human supervisor decomposing a walking gait task between local cyclic leg activity and global, gait coordination. In addition, use of recurrent neural networks for walking gaits requires fairly heavy online computations to be performed in real time, in contrast to the much simpler network of CA lookup tables.

Use of neural networks is also another form of modularization, where each neuron can communicate, perform some form of sensory information processing and can acquire specialized functionality through training. The added advantage of neural network architectures is that the neurons can also generalize unlike a CA lookup table architecture, by exploiting correlations between a combination of sensory inputs thus effectively shrinking the search space. Fixed topology neural networks architectures have been extensively used for multirobot tasks, including building walls, corridors and briar patches (Crabbe & Dyer, 1999) and cooperative transport (Groß & Dorigo, 2003).

However, fixed topology monolithic neural network architectures are also faced with scalability issues. With increased numbers of hidden neurons, one is faced with the effects of *spatial crosstalk* where noisy neurons interfere and drown out signals from feature-detecting neurons (Jacob et al., 1991). Crosstalk in combination with limited supervision (through use of a global fitness function) can lead to the 'bootstrap problem' (Nolfi & Floreano, 2000), where evolutionary algorithms are unable to pick out incrementally better solutions for crossover and mutation resulting in premature stagnation of the evolutionary run. Thus, choosing the wrong network topology may lead to a situation that is either unable to solve the problem or difficult to train (Thangavelautham & D'Eleuterio, 2005).

A critical element of applying neural networks to robotic tasks is how best to design and organize the neural network architecture to facilitate self-organized task decomposition and overcome the 'bootstrap problem'. For these tasks, we may use a global fitness function that doesn't explicitly bias towards a particular task decomposition strategy.

For example, the tiling formation task could be arbitrarily divided into a number of subtasks, including foraging for objects, redistributing object piles, arranging objects in the desired tiling structure locally, merging local lattice structures, reaching a collective consensus and finding/correcting mistakes in the lattice structure. Instead, with less supervision, we rely on the robot controller themselves to determine how best to decompose and solve the task through an artificial Darwinian process.

This is in contrast to other task decomposition techniques that require more supervision including *shaping* (Dorigo & Colombetti, 1998) and *layered learning* (Stone & Veloso, 2000). *Shaping* involves controllers learning on a simplified task with the task difficulty being progressively increased through modification of learning function until a desired set of behaviours emerge. *Layered learning* involves a supervisor partitioning a task into a set of simpler goal functions (corresponding to subtasks). These subtasks are learned sequentially until the controller can solve the corresponding task. Both of these traditional task decomposition strategies rely on supervisor intervention and domain knowledge of a task at hand. For multirobot applications, the necessary local and global behaviours need to be known *a priori* to make decomposition steps meaningful. We believe that for a multirobotic system, it is often easier to identify and quantify the system goal, but determining the necessary cooperative behaviours is often counterintuitive. Limiting the need for supervision also provides numerous advantages including the ability to discover novel solutions that would otherwise be overlooked by a human supervisor.

Fixed-topology ensemble network architectures such as the Mixture of Experts (Jacob et al., 1991), Emergent Modular architecture (Nolfi, 1997) and Binary Relative Lookup (Thangavelautham & D'Eleuterio, 2004) in evolutionary robotics use a gating mechanism to preprocess sensor input and assign modular 'expert' networks to handle specific subtasks. Assigning expert networks to handle aspects of a task is a form of task decomposition. Ensemble networks consist of a hierarchical modularization scheme where networks of neurons are modularized into experts and the gating mechanism used to arbitrate and perform selection amongst the experts. Mixture of Experts uses pre-assigned gating functions that facilitate cooperation amongst the 'expert networks' while Nolfi's emergent modular architecture uses gating neurons to select between two output neurons. The BRL architecture is less constrained, as both the gating mechanism and expert networks are evolved simultaneously and it is scalable to a large number of expert networks.

The limitation with fixed-topology ensemble architectures is the need for supervisor intervention in determining the required topology and number of expert networks. In contrast, with variable length topologies, the intention is to evolve both network architecture and neuronal weights simultaneously. Variable length topologies such as such as Neuro-Evolution of Augmenting Topologies (NEAT) (Stanley & Miikkulainen, 2002) use a one-to-one mapping from genotype to the phenotype. Other techniques use recursive rewriting of the genotype contents to a produce a phenotype such as Cellular Encoding (Gruau, 1994), L-systems (Sims, 1994), Matrix Rewriting (Kitano, 1990), or exploit artificial *ontogeny* (Dellaert & Beer, 1994). *Ontogeny* (morphogenesis) models developmental biology and includes a growth program in the genome that starts from a single egg and subdivides into specialized daughter cells. Other morphogenetic systems include (Bongard & Pfeifer, 2001) and Developmental Embryonal Stages (DES) (Federici & Downing, 2006).

The growth program within many of these morphogenetic systems is controlled through artificial gene regulation. Artificial gene regulation is a process in which gene activation/inhibition regulate (and is regulated by) the expression of other genes. Once the growth program has been completed, there is no further use for gene regulation within the artificial system, which is in stark contrast to biological systems where gene regulation is

always present. In addition, these architectures lack any explicit mechanism to facilitate network modularization evident with the ensemble approaches and are merely variable representations of standard neural network architectures. These variable-length topologies also have to be grown incrementally starting from a single cell in order to minimize the dimensional search space since the size of the network architecture may inadvertently make training difficult (Stanley & Miikkulainen, 2001). With recursive rewriting of the phenotype, limited mutations can result in substantial changes to the growth program. Such techniques also introduce a *deceptive fitness landscape* where limited fitness sampling of a phenotype may not correspond well to the genotype resulting in premature search stagnation (Roggen & Federici, 2004).

Artificial Neural Tissues (Thangavelautham and D'Eleuterio, 2005) address limitations evident with existing variable length topologies through modelling of a number of biologically-plausible mechanisms. Artificial Neural Tissues (ANT) includes a coarse-coding-based neural regulatory system that is similar to the network modularity evident in fixed-topology ensemble approaches. ANT also uses a nonrecursive genotype-to-phenotype mapping avoiding deceptive fitness landscapes, and includes gene duplication similar to DES. Gene duplication involves making redundant copies of a master gene and facilitates *neutral complexification*, where the copied gene undergoes mutational drift and results in expression of incremental innovations (Federici & Downing, 2006). In addition, both gene and neural-regulatory functionality limits the need to grow the architecture incrementally, as there exist mechanisms to selectively activate and inhibit parts of a tissue even after completion of the growth program.

A review of past work highlights the possibility of training multirobot controllers with limited supervision using only a global fitness function to perform self-organized task decomposition. These techniques also show that by exploiting hierarchical modularity and regulatory functionality, controllers can overcome tractability concerns. In the following sections, we explore a number of techniques we have used in greater detail.

3. Tasks

3.1 Heap-Formation

The heap-formation task or object-clustering has been extensively studied and is analogous to the behaviour in some social insects (Deneubourg et al., 1991). In the space exploration context, this is relevant to gathering rocks or other materials of interest. It is believed that this task requires global coordination for a group of decentralized agents, existing in a two-dimensional space, to move some randomly distributed objects into a single large cluster (Fig. 1). Owing to the distributed nature of the agents, there is no central controlling agent to determine where to put the cluster and the agents must come to a common decision among themselves without any external supervision (analogous to the global partitioning task in cellular automata work (Mitchell et al., 1996)). The use of distributed, homogenous sets of agents exploits both redundancy and parallelism. Each agent within the collective has limited sensory range and lacks a global blueprint of the task at hand but cooperative coordination amongst agents can, as we show here, make up for these limitations (Barfoot & D'Eleuterio, 1999); (Barfoot & D'Eleuterio, 2005).

To make use of Evolutionary Algorithms (EAs), a fitness function needs to be defined for the task. Herein we define a fitness function that can facilitate selection of controllers for the task at hand without explicitly biasing for a particular task decomposition strategy or set of

behaviours. In contrast to this idea, the task could be manually decomposed into a number of potential subtasks, including foraging for objects, piling objects found into small transitionary piles, merging small piles into larger ones and reaching a collective consensus in site selection for merging all the piles. Traditional fitness functions such as (Dorigo & Colombetti, 1998) involve summing separate behaviour shaping functions that explicitly tune the controllers towards predetermined set of desired behaviours. With multiagent systems, it is not always evident how best to systematically determine these behaviours. It is often easier to identify the global goals within the system but not the coordination behaviours necessary to accomplish the global goals. Thus, the fitness functions we present here perform an overall global fitness measure of the system and lack explicit shaping for a particular set of behaviours. The intention is for the multiagent system to self-organize into cooperatively solving the task.

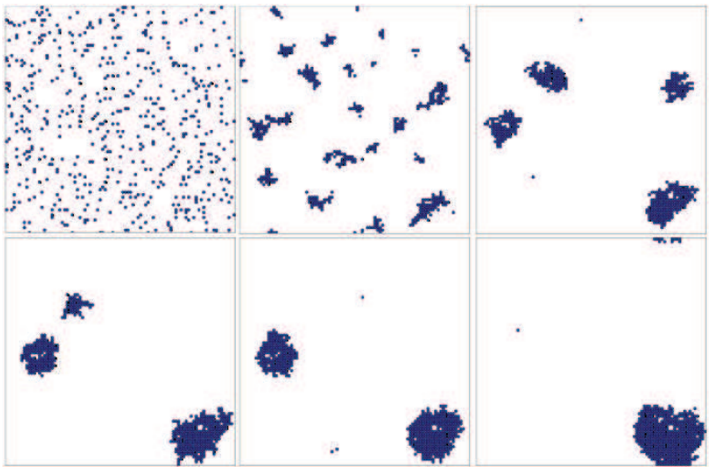


Figure 1. Typical snapshots of system at various time steps (0; 1010; 6778; 14924; 20153; 58006). The world size is 91 × 90; there are 270 agents and 540 objects. Only the objects (dark circles) are shown for clarity

For the heap formation task, the two dimensional grid world in which the agents exist is broken into J bins, A_j, of size $l \times l$. We use a fitness function based on Shannon's entropy as defined below:

$$f_i = 1 + \frac{\sum_{j=1}^{J} q_j \ln q_j}{\ln J} \tag{1}$$

where q_j is defined as follows:

$$q_j = \frac{n(A_j)}{\sum_{j=1}^{J} n(A_j)} \tag{2}$$

$n(A_j)$ is the number of objects in bin A_j so that $0 \leq f_i \leq 1$. To summarize, fitness is assigned to a controller by equipping each agent in a collective with the same controller. 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 (varying initial conditions) to determine the average fitness.

3.1.1 Cellular Automata

To perform the task, each robot-like agent is equipped with 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 orthogonal (Kube & Zhang, 1996) virtual sensors that output a discrete value. This 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 behaviours* (Matarić, 1997). Rather than specify the actuator positions (or velocities), we assume that we may select a simple behaviour from a finite predefined palette. This may be considered a post-processing step that takes a discretized output and converts it to the actual actuator control. The actual construction of these transformations requires careful consideration but is also somewhat arbitrary.

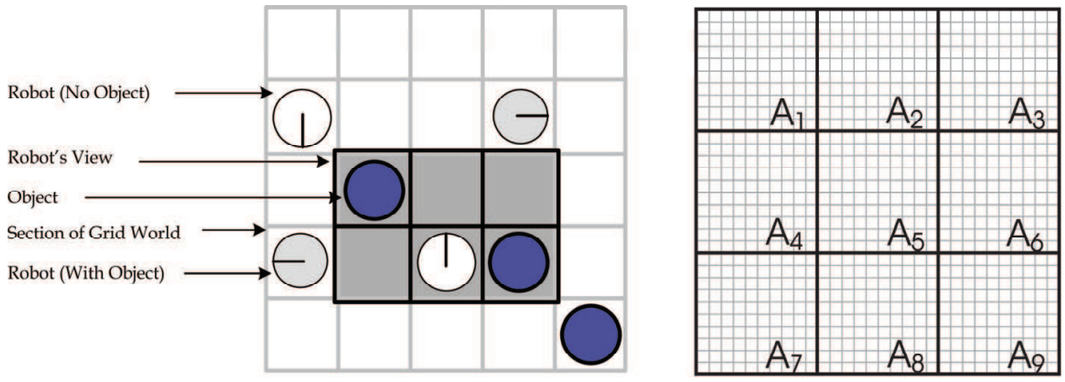


Figure 2. (Left) Typical view of a simulated robot. Circles (with the line indicating orientation) are robots. Dark circles are objects. (Right) Partition of grid world into bins for fitness calculation

Once the pre/post-processing 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 behaviours). The simulated robot's sensor input layout is shown in Fig. 2 (left). The number of possible views for the robot is $3^5 \times 2 = 486$. Each of 5 cells can be free, occupied by an object or by another robot. The robot itself can either be holding an object or not. For an arbitration scheme, we use a lookup-table similar to Cellular Automata, in which the axes are the sensory inputs and the contents of the table, the output behaviours. It could be argued that CAs are the simplest example of a multiagent system for the heap formation task. With 2 basis behaviours and a CA lookup-table of size 486, there are 2^{486} 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.

3.1.2 Experiments

Fig. 3 (left) shows a typical evolutionary run using CA lookup table controllers. Analysis of the population best taken after 150 generations shows that controllers learn to form small piles of objects, which are over time merged into a single large pile. Even with a very simple controller, we can obtain coordinated behaviour amongst the agents. The agents

communicate indirectly among themselves through stigmergy (by manipulating the environment). This is primarily used to seed piles that are in turn merged into clusters. The benefits of using a decentralized multiagent controller is that the system may also be rescaled to a larger world size.

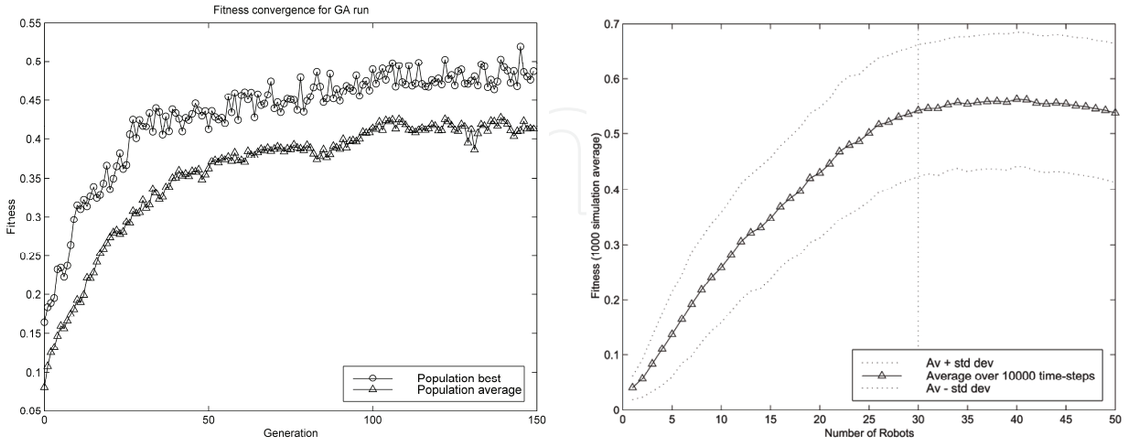


Figure 3. (Left) Convergence history of a typical EA run with a population size of 50. (Right) Max system fitness with number of robots rescaled. Solution was evolved with 30 simulated robots

The controllers were evolved with one particular set of parameters (30 robots, 31×30 world size, 60 objects) but the densities of agents and resources can be rescaled, without rendering the controllers obsolete. This particular trait gives us a better understanding of the robustness of these systems and some of their limitations. Altering the agent density, while keeping all other parameters constant, shows the system performs best under densities slighter higher than during training as shown in Fig. 3 (right), accounting for a few agents getting stuck. With too few agents, the system is under-populated and hence takes longer for coordination while too many agents result in disrupting the progress of the system due to *antagonism*. Thus maintaining a constant density scaling with respect to the training parameters, the overall performance of the system compares well when scaled to larger worlds. What we witness from these experiments is that with very simple evolved multiagent controllers, it is feasible to rescale the system to larger world sizes.

3.2 Tiling Pattern Formation

The tiling pattern formation task (Thangavelautham et al., 2003), in contrast to the heap-formation task, involves redistributing objects (blocks) piled up in a two-dimensional world into a desired tiling structure (Fig. 4). In a decentralized setup, the agents need to come to a consensus and form one 'perfect' tiling pattern. This task also draws inspiration from biology, namely a termite-nest construction task that involves redistributing pheromone-filled pellets on the nest floor (Deneubourg, 1977). Once the pheromone pellets are uniformly distributed, termites use the pellets as markers for constructing pillars to support the nest roof.

In contrast to our emergent task decomposition approach, the tiling pattern formation task could be arbitrarily decomposed into a number of potential subtasks. These may include foraging for objects (blocks), redistributing block piles, arranging blocks in the desired tiling

structure locally, merging local lattice structures, reaching a collective consensus and finding/correcting mistakes in the lattice structure. Instead, we are interested in evolving homogenous decentralized controllers (similar to a nest of termites) for the task without need for human assisted task decomposition.

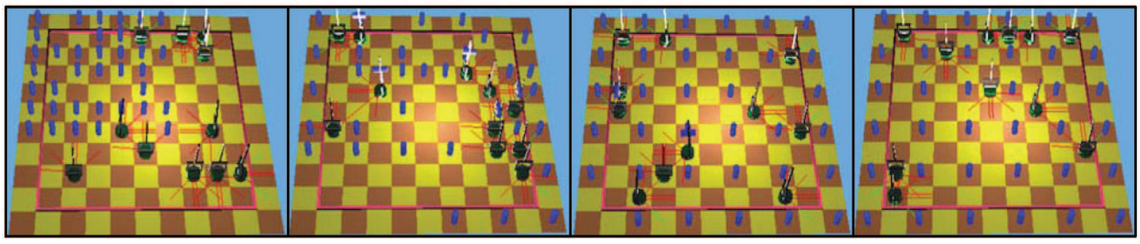


Figure 4. Typical simulation snapshots at various timesteps (0; 100; 400; 410) taken for the 2 × 2 tiling formation task. Solutions were evolved on an 11 × 11 world (11 robots, 36 blocks)

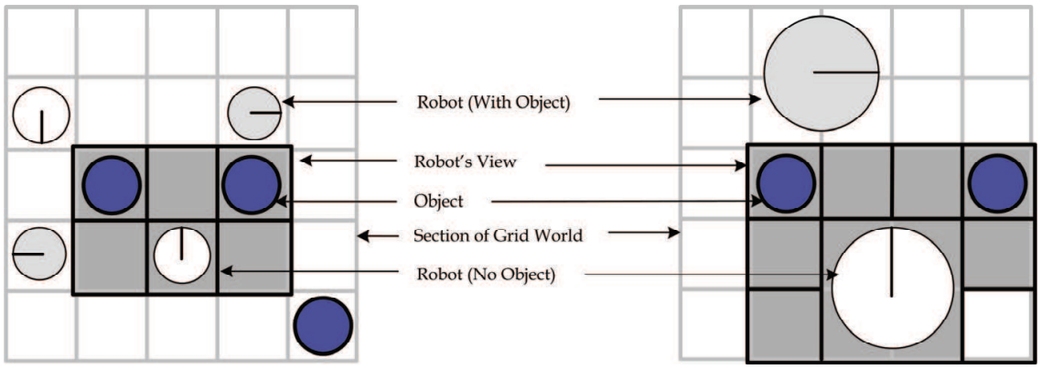


Figure 5. Typical view of a simulated robot for the 2 × 2 (left) and 3 × 3 (right) tiling formation tasks. Each robot can sense objects, other agents and empty space in the 5 (left) and 7 (right) shaded squares surrounding the robot

As shown earlier with the heap formation task, decentralized control offers some inherent advantages including the ability to scale up to a larger problem size. Furthermore, task complexity is dependent on the intended tile spacing, because more sensors would be required to construct a 'wider' tiling pattern. In addition, we use Shannon's entropy to be a suitable fitness function for the tiling formation task. For the $m \times m$ tiling pattern formation task, we use Eq. 3 as the fitness function, with q_j used from Eq. 2.

$$f_i = \frac{-\sum_{j=1}^{J} q_j \ln q_j}{\ln J} \tag{3}$$

The sensor input layouts for the simulated robots used for the 2 × 2 and 3 × 3 tiling formation task are shown in Fig. 5.

3.2.1 Emergent Task-Decomposition Architectures

It turns out that the 3 × 3 tiling pattern formation task is computationally intractable for EAs to find a viable monolithic CA lookup table. To overcome this hurdle, we also considered the use of neural networks as multiagent controllers. Here we discuss a modular neural network architecture called Emergent Task-Decomposition Networks (ETDNs). ETDNs

(Thangavelautham et al., 2004) consist of a set of decision networks that mediate competition and a modular set of expert network that compete for behaviour control. The role of decision networks is to preprocess the sensory input and explicitly 'select' for a specialist expert network to perform an output behaviour. A simple example of an ETDN architecture includes a single decision neuron arbitrating between two expert networks as shown in Fig. 6. This approach is a form of task decomposition, whereby separate expert modules are assigned handling of subtasks, based on an evolved sensory input-based decision scheme.

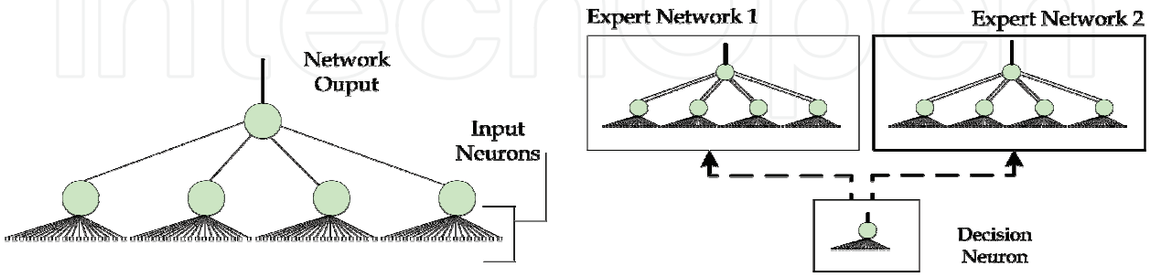


Figure 6. (Left) An example of the non-emergent network used in our experiments. (Right) ETDN architecture consisting of a decision neuron that arbitrates between 2 expert networks

The architecture exploits network modularity, evolutionary competition and specialization to facilitate emergent (self-organized) task decomposition. Unlike traditional machine learning methods, where handcrafted learning functions are used to train the decision and expert networks separately, ETDN architectures require only a global fitness function. The intent is for the architecture to evolve the ability to decompose a complex task into a set of simpler tasks with limited supervision.

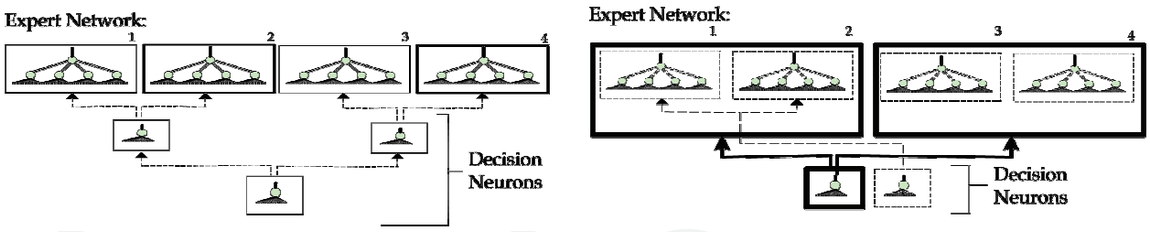


Figure 7. (Left) BDT Architecture with 4 expert networks and 3 decision neurons. (Right) BRL Architecture with 4 expert networks and 2 decision neurons

The ETDNs can also be generalized for n_E expert networks. Here we discuss two extensions to the ETDN architecture, namely the Binary Relative Lookup (BRL) architecture and Binary Decision Tree (BDT) architecture (see Fig. 7). The Binary Relative Lookup (BRL) architecture consists of a set of n_D unconnected decision neurons that arbitrate among 2^{n_D} expert networks. Starting from left to right, each additional decision neuron determines the specific grouping of expert networks relative to the selected group. Since the decision neurons are unconnected, this architecture is well-suited for parallel implementation.

The Binary Decision Tree (BDT) architecture is represented as a binary tree where the tree nodes consist of decision neurons and the leaves consist of expert networks. For this architecture, n_D decision neurons arbitrate among n_D + 1 expert networks. The tree is traversed by starting from the root and computing decision neurons along each selected branch node until an expert network is selected. Unlike BRLs, there is a one-to-one

mapping between the set of decision neurons output states and the corresponding expert network. The computational cost of the decision neurons for both architectures is $C_D \alpha \log n_E$.

We also introduce modularity within each neuron, through the use of a modular activation function, where the EAs are used to train weights, thresholds and choice of activation function. The inputs and output from the modular activation function consist of discrete states as opposed to real values. It is considered a modular activation function, since a neuron's behaviour could be completely altered by changing the selected activation function while holding the *modular* weights constant. The modular neuron could assume one of four different activation functions listed below:

$$\begin{array}{llll} \phi_1: & s_{out} = \begin{cases} 0, & \text{if } p(x) \le t_1 \\ 1, & \text{if } p(x) > t_1 \end{cases} & \phi_3: & s_{out} = \begin{cases} 0, & \text{if } t_2 < p(x) < t_1 \\ 1, & \text{otherwise} \end{cases} \\ \phi_2: & s_{out} = \begin{cases} 0, & \text{if } p(x) \ge t_2 \\ 1, & \text{if } p(x) < t_2 \end{cases} & \phi_4: & s_{out} = \begin{cases} 0, & \text{if } p(x) > 0.5 \\ \text{rand}(0,1), & \text{if } p(x) = 0.5 \\ 1, & \text{if } p(x) < 0.5 \end{cases} \end{array} \quad (3)$$

These threshold functions maybe summarized in a single analytical expression as:

$$\phi = (1 - k_1)[(1 - k_2)\phi_1 + k_2\phi_2] + k_1[(1 - k_2)\phi_3 + k_2\phi_4] \quad (4)$$

Each neuron outputs one of two states $s \in S = \{0, 1\}$, and the activation function is thus encoded in the genome by k_1, k_2 and the threshold parameters t_1, $t_2 \in$ □ , where $p(x)$ is defined as follows:

$$p(x) = \frac{\sum_i w_i x_i}{\sum_i x_i} \quad (5)$$

w_i is a neuron weight and x_i is an element of the input state vector. With two threshold parameters, a single neuron could be used to simulate AND, OR, NOT and XOR functions. The assumption is that a compact yet sufficiently complex (functional) neuron will speed up evolutionary training since this will reduce the need for more hidden layers and thus result in smaller networks.

3.2.2 Experiments

As mentioned above, we find that for the 3 × 3 tiling formation task, a lookup table architecture is found to be intractable (Fig. 8, top left). The CA lookup table architecture appears to fall victim to the bootstrap problem, since EAs are unable to find an incrementally better solution during the early phase of evolution resulting in search stagnation. In contrast, ETDN architectures can successfully solve this version of the tiling formation task and outperform other regular neural network architectures (regardless of the activation function used). Analysis of a typical solution (for ETDN with 16 expert nets) suggests decision neuron assigns expert networks not according to 'recognizable' *distal* behaviours but as *proximal* behaviours (organized according to proximity in sensor space) (Nolfi, 1997). This process of expert network assignment is evidence of task decomposition, through role assignment (Fig. 8, bottom right).

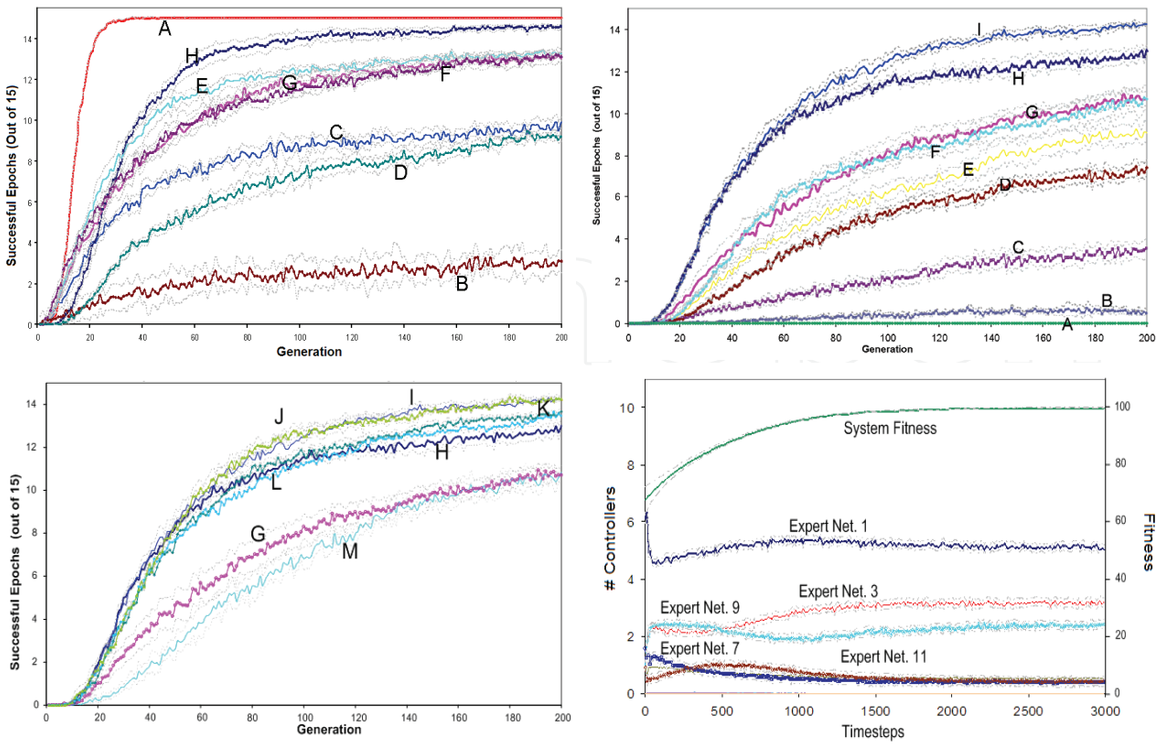


Figure 8. Evolutionary performance comparison, 2 × 2 (Top Left), 3 × 3 (Top Right, Bottom Left) tiling formation task, averaged over 120 EA runs. (Bottom Right) System activity for BRL (16 Expert Nets) (**A**) CA Look-up Table, (**B**) ESP (using Non-Emergent Net), (**C**) Non-Emergent Net (Sigmoid), (**D**) ETDN (2 Expert Nets, Sigmoid), (**E**) Non-Emergent Net. (Threshold), (**F**) ETDN (2 Expert Nets, Threshold), (**G**) Non-Emergent Net. (Modular), (**H**) ETDN (2 Expert Nets, Modular), (**I**) BRL (16 Expert Nets, Modular), (**J**) BRL (32 Expert Nets, Modular), (**K**) BRL (8 Expert Nets, Modular), (**L**) BRL (4 Expert Nets, Modular), (**M**) BDT (4 Expert Nets, Modular)

It should be noted that the larger BRL architecture, with more expert networks outperformed (or performed as well as) the smaller ones (evident after about 80 generations) (Fig. 8, bottom left). It is hypothesized that by increasing the number of expert networks, competition among candidate expert networks is further increased thus improving the chance of finding a desired solution. However, as the number of expert networks is increased (beyond 16), the relative improvement in performance is minimal, for this particular task.

ETDN architectures also have some limitations. For the simpler 2 × 2 tiling pattern formation task, a CA lookup table approach evolves desired solutions faster than the neural network architectures including ETDNs (Fig. 8, top right). This suggests that ETDNs may not be the most efficient strategy for smaller search spaces (2^{486} candidate solutions for the 2 × 2 tiling formation task versus 2^{4374} for the 3 × 3 version). Our conventional ETDN architecture consisting of a single threshold activation function evolves more slowly than the non-emergent architectures. The ETDN architectures include an additional 'overhead', since the evolutionary performance is dependent on the evolution (in tandem) of the expert networks and decision neurons resulting in slower performance for simpler tasks.

However, the ETDN architecture that combines the modular activation function outperforms all other network architectures tested. The performance of the modular neurons appears to partially offset the 'overhead' of the bigger ETDN architecture. A 'richer' activation function set is hypothesized to improve the ability of the decision neurons to switch between suitable expert networks with fewer mutational changes.

3.3 Walking Gait

For the walking gait task (Earon et al., 2000); (Barfoot et al., 2006), a network of leg based controllers forming a hexapod robot (Fig. 9) need to find a suitable walking gait pattern that enables the robot to travel forward. We use Evolutionary Algorithms on hardware, to coevolve a network of CA walking controllers for the hexapod robot, *Kafka*. The fitness is simply the distance travelled by the robot, measured by an odometer attached to a moving tread mill. The robot has been mounted on an unmotorized treadmill in order to automatically measure controller performance (for walking in a straight line only). As with other experiments, the fitness measure is a global one and does not explicitly shape for a particular walking gait pattern, rather we seek the emergence of such behaviours through multiagent coordination amongst the leg controllers.

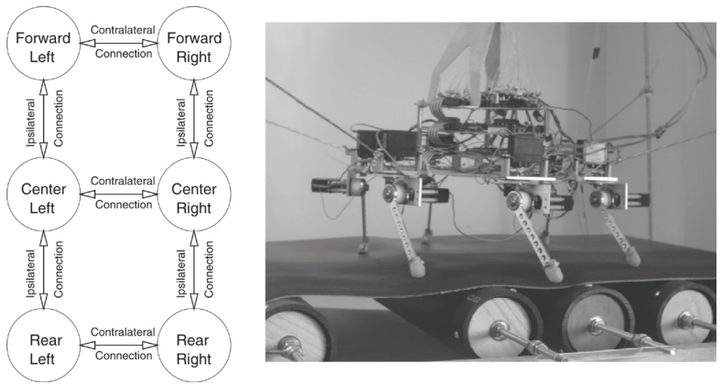


Figure 9. (Left) Behavioural coupling between legs in stick insects (Cruse, 1990). (Right) *Kafka*, a hexapod robot, and treadmill setup designed to evolve walking gaits

3.2.1 Network of Cellular-Automata Controllers

According to neurobiological evidence (Cruse, 1990), the behaviour of legs in stick insects is locally coupled as in Fig. 9 (left). This pattern of ipsilateral and contralateral connections will be adopted for the purposes of discussion although any pattern could be used in general (however, only some of them would be capable of producing viable walking gaits). The states for *Kafka's* legs are constrained to move only in a clockwise, forward motion. The control signals to the servos are absolute positions to which the servos then move as quickly as possible. Based on the hardware setup, we make some assumptions, namely that the output of each leg controller is independent and discrete. This is in contrast to use of a central pattern generator to perform coordination amongst the leg controllers (Porcino, 1990). This may be done by way of a set of basis behaviours. Rather than specify the actuator positions (or velocities) for all times, we assume that we may select a simple behaviour from a finite predefined palette. The actual construction of the behaviours requires careful consideration but is also somewhat arbitrary.

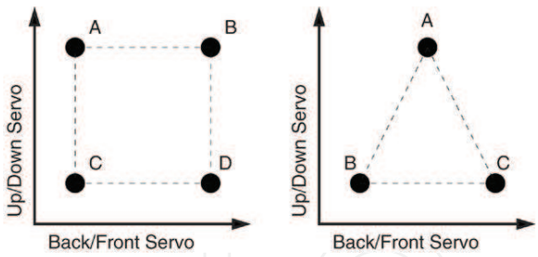


Figure 10. Example discretizations of output space for 2 degree of freedom legs into (Left) 4 zones and (Right) 3 zones

Here the basis behaviours will be modules that move the leg from its current zone (in output space) to one of a finite number of other zones. Fig. 10 shows two possible discretizations of a two-degree-of-freedom output space (corresponding to a simple leg) into 4 or 3 zones. Execution of a discrete behaviour does not guarantee the success of the corresponding leg action due to terrain variability. This is in contrast to taking readings of the leg's current zone, which gives an accurate state (local feedback signal) of the current leg position. The only feedback signal available for the discrete behaviour controller is a global one, the total distance travelled by the robot. The challenge therefore is to find an appropriate arbitration scheme which takes in a discrete input state, d, (basis behaviours of self and neighbours) and outputs the appropriate discrete output, o, (one of M basis behaviours) for each leg. One of the simpler solutions is to use separate lookup tables similar to cellular automata (CA) for each leg controller.

3.2.1 Experiments

Decentralized controllers for insect robots offer a great deal of redundancy, even if one controller fails, the robot may still limp along under the power of the remaining functional legs. The cellular automata controller approach was successfully able to control *Kafka*, a hexapod robot, and should extend to robots with more degrees of freedom (keeping in mind scaling issues). Coevolution resulted in the discovery of controllers comparable to the tripod gate (Fig. 11, 12). One advantage of using cellular automata, is that it requires very few real-time computations to be made (compared to a dynamic neural network approaches). Each leg is simply looking up its behaviour in a table and has wider applicability in hardware. The approach easily lends itself to automatic generation of controllers as was shown for the simple examples presented here.

We found that a coevolutionary technique using a network of CAs were able to produce distributed controllers that were comparable in performance to the best hand-coded solutions. In comparison, reinforcement learning techniques such as cooperative Q-learning, was much faster at this task (e.g., 1 hour instead of 8) but required a great deal more information as it was receiving feedback after shorter time-step intervals (Barfoot et al., 2006). Although both methods were using the same sensor, the reinforcement learning approach took advantage of the more detailed breakdown of rewards to increase convergence rate. The cost of this speed-up could also be seen in the need to prescribe an exploration strategy and thus determine a suitable rewarding scheme by hand. However, the coevolutionary approach requires fewer parameters to be tuned which could be advantageous for some applications.

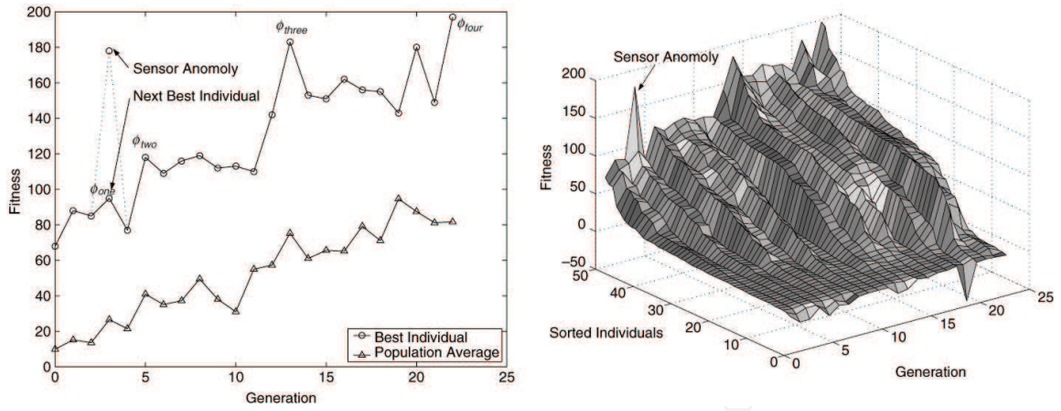


Figure 11. Convergence History of a GA Run. (Left) Best and average fitness plots over the evolution. (Right) Fitness of entire population over the evolution (individuals ordered by fitness). Note there was one data point discounted as an odometer sensor anomaly

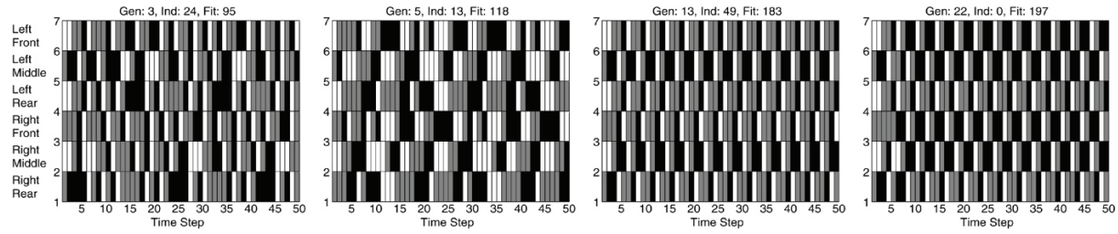


Figure 12. Gait diagrams (time histories) for the four solutions, ϕ_{one}, ϕ_{two}, ϕ_{three}, ϕ_{four} respectively. Colours correspond to each of the three leg zones in Figure 10 (right)

3.4 Resource Gathering

In this section, we look at the resource-collection task (Thangavelautham et al., 2007), which is motivated by plans to collect and process raw material on the lunar surface. Furthermore, we address the issue of *scalability*; that is, how does a controller evolved on a single robot or a small group of robots but intended for use in a larger collective of agents scale? We also investigate the associated problem of *antagonism*. For the resource gathering task, a team of robots collects resource material distributed throughout its work space and deposits it in a designated dumping area by exploiting *templates* (Fig. 13). This is in contrast to the heap formation task, where simulated robots can gather objects anywhere on the 2-D grid world. For this task, the controller must possess a number of capabilities including gathering resource material, avoiding the workspace perimeter, avoiding collisions with other robots, and forming resources into a mound at the designated location. The dumping region has perimeter markings on the floor and a light beacon mounted nearby. The two colours on the border are intended to allow the controller to determine whether the robot is inside or outside the dumping location. Though solutions can be found without the light beacon, its presence improves the efficiency of the solutions found, as it allows the robots to track the target location from a distance instead of randomly searching the workspace for the perimeter. The global fitness function for the task measures the amount of resource material accumulated in the designated location within a finite time period.

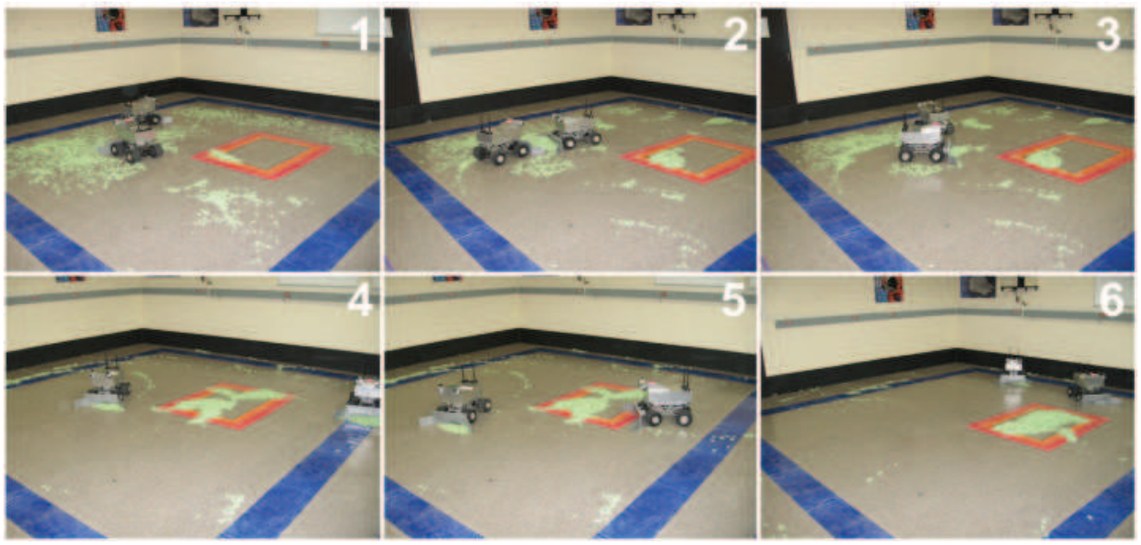
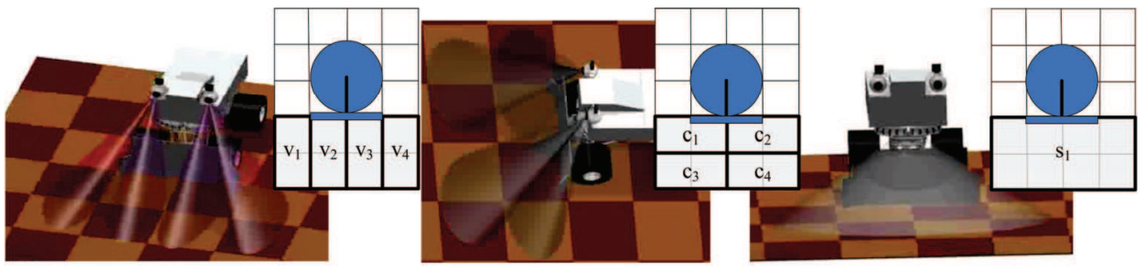


Figure 13. Snapshots of two rovers performing the resource collection task using an ANT controller. Frames 2 and 3 show the 'bucket brigade' behaviour, while frames 4 and 5 show the boundary avoidance behaviour



Resources. Color Template. Obstacle Avoidance.

Figure 14. Robot Input Sensor Mapping, Simulation Model Shown Inset for Resource Gathering task

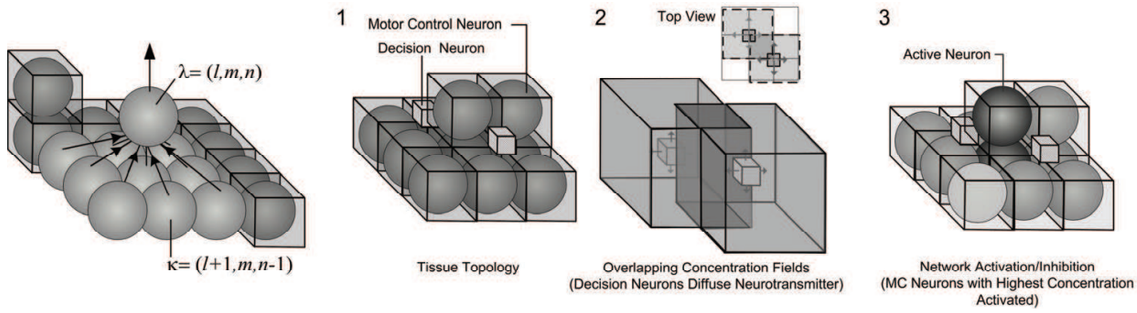
The simulated robots are modelled on a fleet of rovers designed and built at the University of Toronto Institute for Aerospace Studies. The sensor input layout for the rovers is shown in Fig. 14. For this task we consider standard fixed-topology neural networks and a variable-length topology called Artificial Neural Tissues (Thangavelautham & D'Eleuterio, 2005). ANT has been applied on a number of robotic tasks including a tiling pattern formation task, single-robot phototaxis (light homing) task and a sign following task. For the multiagent tiling pattern formation task, ANT outperformed standard fixed-topology neural networks and BRL networks (Thangavelautham & D'Eleuterio, 2005).

3.3.1 Artificial Neural Tissues

One of the limitations with a fixed topology network is that it requires *a priori* supervisor intervention in determining the network topology. An incorrect choice may result in longer training times or inability to find a desired solution. What we are after is a scalable framework that explicitly facilitates self-organized task-decomposition while requiring minimal human intervention. This would be limited to providing a generic palette of basis behaviours, sensory inputs and a global fitness function for a task at hand.

In turn, we expect an algorithm to find a suitable multirobot controller solution such as for the resource gathering task. Artificial Neural Tissues (ANT) addresses all of these stated requirements.

ANT consists of a developmental program, encoded in the genome that constructs a three-dimensional neural tissue and associated regulatory functionality. By regulatory functionality, we mean a dynamic system that can selectively activate and inhibit neuronal groups. The tissue consists of two types of neural units, decision neurons and motor-control neurons, or simply motor neurons. Both the motor neurons and decision neurons are grown by evaluating a growth program defined in the genome. The variable-length genome is modular, consisting of genes that identify characteristics of each gene. The growth program in turn reads the contents of the genes and constructs a three-dimensional lattice structure consisting of the motor control and decision neurons. Mutations to the growth program result in addition of new miscopied genes (a means of gene replication) corresponding to formation of new neurons within the tissue or inhibition of existing genes.



Synaptic Connections. Coarse Coding.

Figure 15. (Left) Synaptic Connections between Motor Neurons from Layer *l*+1 to *l*. (Right) Activated Decision Neurons Diffuse Neurotransmitter Concentration Field Resulting in Activation of Motor Control Neurons with Highest Activation Concentration

Synaptic connections between the motor neurons are local, with up to 9 neighbouring motor neurons feeding from adjoining layers as shown in Fig. 15 (left). Decision neurons are fully connected to all the sensory input; however, these neurons do not share synaptic connections (wired electrical connections) with the motor neurons. Motivated by biological evidence of chemical communication in the nervous system, the decision neurons diffuse neurotransmitters chemicals, forming neurotransmitter fields as shown in Fig. 15 (right). One may think of the neurotransmitter fields as a form of wireless (as opposed to wired) communication between neurons. Motor neurons enveloped by superpositioning of multiple neurotransmitter fields and above a prescribed neurotransmitter density are activated, while the remaining motor neurons are inhibited as shown in Figure 15 (right). Through the coordinated interaction of multiple decision neurons, one observes coarse-coding of the neurotransmitter fields, selecting for wired networks of motor neurons. These selected networks of motor neurons in effect resemble the 'expert networks' from the ETDN architecture but are dynamic. These dynamic interactions result in new neurotransmitter fields being formed or dissipation of existing ones depending on sensory input being fed into the decision neurons.

3.3.2 Experiments

Fig. 16 (left) shows the fitness (population best) of the overall system evaluated at each generation of the artificial evolutionary process for the resource gathering task. The performance of a fixed-topology, fully-connected network with 12 hidden and output neurons is also shown in Fig. 16 (right). While this is not intended as a benchmark network, in a fixed-topology network there tends to be more 'active' synaptic connections present (since all neurons are active), and thus it takes longer for each neuron to tune these connections for all sensory inputs.

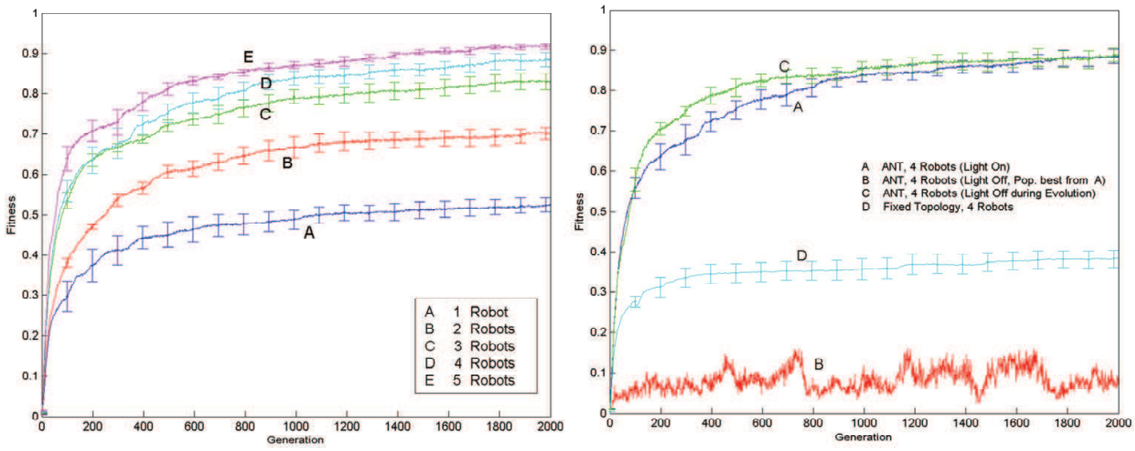


Figure 16. (Left) Evolutionary performance comparison of ANT-based solutions for 1 to 5 robots (averaged over 30 EA runs). (Right) Evolutionary performance with 4 robots for fixed topology and with light beacon off (averaged over 30 EA runs). Error bars indicate one standard deviation

The results with ANT controllers in Figure 16 (left) show that performance increases with the number of robots. With more robots, each robot has a smaller area to cover in trying to gather and dump resources. The simulation runs indicate that a point of diminishing returns is eventually reached. Beyond this point, additional robots have a minimal effect on system performance with the initial resource density and robot density kept constant. The evolutionary process enables the decomposition of a goal task based on a global fitness function, and the tuning of behaviours depending on the robot density.

The behavioural activity of the controllers (see Fig. 17) shows the formation of small networks of neurons, each of which handles an individual behaviour such as dumping resources or detecting visual templates (boundary perimeters, target area markings, etc.). Localized regions within the tissue do not exclusively handle these specific distal behaviours. Instead, the activity of the decision neurons indicate distribution of specialized 'feature detectors' among independent networks.

The ANT controllers discover a number of emergent behaviours including learning to pass resource material from one individual to another during an encounter, much like a 'bucket brigade.' This technique improves the overall efficiency of system as less time is spent travelling to and from the dumping area. In addition, 'bucket brigades' reduce *antagonism* since there are fewer robots to avoid at once within the dumping area.

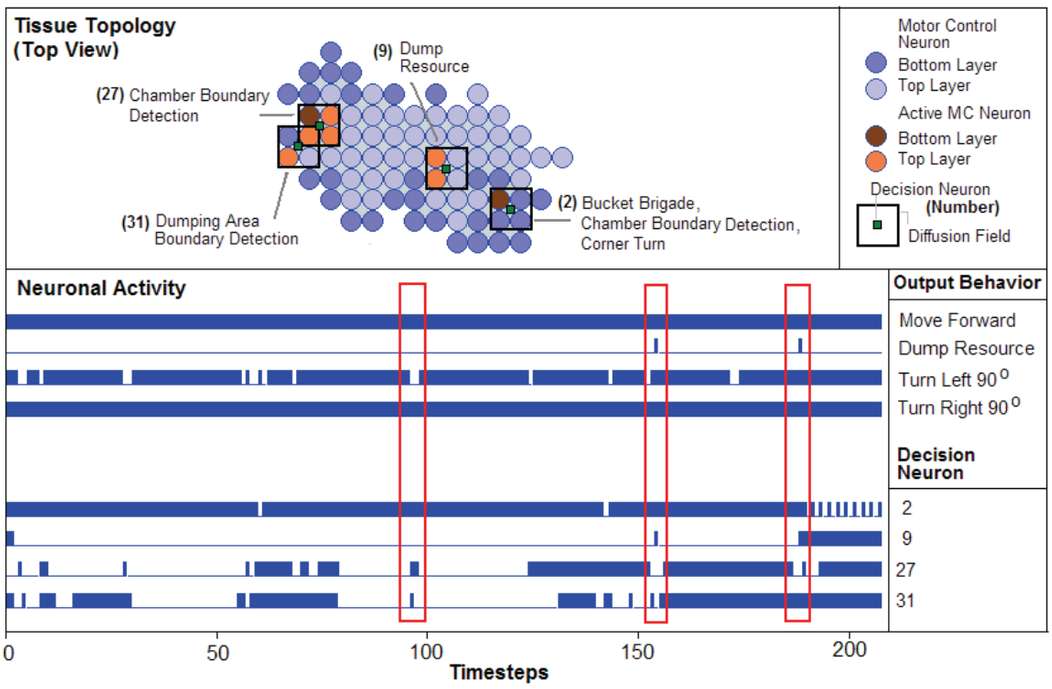


Figure 17. ANT tissue topology and neuronal activity of a select number of decision neurons. These decision neurons in turn 'select' (excite into operation) motor control neurons within its diffusion field

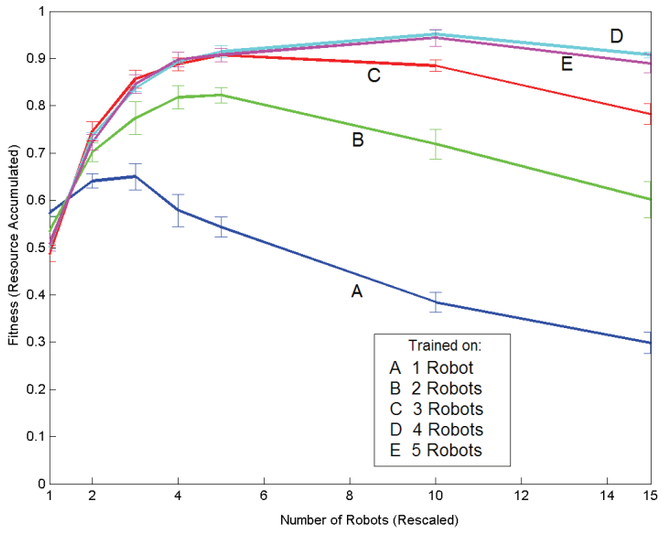


Figure 18. Scaling of ANT-based solutions (population best taken after 2000 generations) from 1 to 5 robots

3.3.3 Controller Scalability and Hardware Demonstrations

We also examine the scalability of the fittest evolved controllers by varying the density of simulated robots, while holding the resource density constant (Fig. 18). Taking the controller evolved for a single robot and running it on a multirobot system shows limited

performance improvement. In fact, using four or more robots results in a decrease in performance, due to the increased antagonism created.
The scalability of the evolved solution depends in large part on the number of robots used during the training runs. The single-robot controller expectedly lacks the cooperative behaviour necessary to function well within a multiagent setting. For example, such controllers fail to develop 'robot collision avoidance' or 'bucket brigade' behaviours and perform poorly when rescaled to multiple robots. Similarly, the robot controllers evolved with two or more robots perform demonstrably better when scaled up, showing that the solutions are dependent on cooperation among the robots. To limit the effects of antagonism, controllers need to be trained under conditions in which the probability of encounters among robots is sufficiently high.
Some of the fittest evolved controllers were also tested on a fleet of nonholonomic robots; snapshots are shown in Fig. 13. The advantage of the ANT framework with generic basis behaviours is that the solutions can be easily ported to various other hardware platforms, provided actual basis behaviours on hardware match those in simulation.

4. Discussion

The use of a global fitness function shows the possibility of training multirobot controllers with limited supervision to perform self-organized task decomposition. A global fitness function encourages solutions that improve system performance without explicitly biasing for a particular task decomposition strategy. This is advantageous for use with multirobot controllers, where it is often easier to define the global goals over required local coordination behaviours that require task specific information. In addition, such an approach facilitates discovery of novel behaviours that otherwise may be overlooked by a human designer. Such novel behaviours include use of 'bucket brigade' behaviours for the multirobot resource gathering task, 'error correction' for the tiling formation task and incremental merging of the object piles for the heap formation task. For the walking gait task, use of a global fitness function facilitated discovery of the local leg behaviours and resultant global gait coordination without the need for multistaged evolution.
While some of the behaviours from evolved controllers may display novel attributes, quantitatively interpreting such solutions remains difficult. This issue is of added importance because we are interested in how a set of local behaviours give rise to emergent global behaviours for multirobot control. As Dawkins (1986) points out, successful evolutionary solutions are not evolved per se to ease our burden of understanding such solutions. Biologically plausible techniques such as the ability to track energy consumption of the individual robots and bias for minimization of energy consumption within the fitness function also does not guarantee more decipherable solutions nor improved evolutionary training performance than without selecting for energy efficient controllers. However, such implicit techniques may well reduce any redundant behaviours that may be energetically wasteful. Alternatively, desired attributes could be explicitly encouraged through *shaping*, helping to simplify deciphering such solutions but this implies dealing with the problem of greater supervision for multirobot control and how best to perform decomposition of the task *a priori*.
Comparison of the different evolutionary techniques discussed in this chapter shows that by exploiting hierarchical modularity and regulatory functionality, controllers can overcome tractability concerns. Simple CA lookup table controllers are shown to be suitable for

solving a multiagent heap and 2 × 2 tiling formation tasks. Although lookup tables are monolithic, the system exploits parallelism through use of decentralized modular agents. To overcome the tractability limitations of monolithic CA lookup table controllers, we introduce modularity through decomposition of agent functionality into subsystems for the 3 × 3 tiling formation task or by exploiting inherent physical modularity for the walking gait task. However, such techniques often require supervisor intervention or a conducive physical setup in devising a suitable modular task decomposition scheme.

Use of neural networks is another form of modularization, where individual neurons acquire specialized functionality (via training) in solving for a given task. The neurons unlike the CA lookup tables can generalize by exploiting correlations between combinations of sensory input. Ensemble networks consist of a hierarchical modularization scheme, where networks of neurons are modularized into expert networks and the gating mechanism used to arbitrate and perform selection among the expert networks. In addition, these networks can also exploit modularity within the neuron activation function. Such techniques are shown to overcome tractability issues including the 'bootstrap problem' and reduce *spatial crosstalk* without use of a supervisor-devised task decomposition schemes.

The limitation with fixed-topology ensemble networks is that they require supervisor intervention in determining the network topology and number of expert networks. Artificial Neural Tissues (ANT) can overcome the need for supervisor intervention in determining the network topologies through use of artificial regulatory systems. Within ANT, the regulatory systems dynamically activate and inhibit neuronal groups by modelling a biologically plausible coarse-coding arbitration scheme. ANT can also overcome tractability concerns affecting other variable-length topologies that lack neural regulation. Variable-length topologies that lack neural regulatory mechanisms are grown incrementally starting from a single cell as the size of the network architecture may inadvertently make training difficult (Stanley & Miikkulainen, 2001).

The techniques used to evolve controllers for the tasks discussed also rely on predetermined sensor input layout and set of predefined basis behaviours. In practice, a sensor-input layout is often constrained due to hardware limitations or based on physical characteristics of a robot and thus relying on body-brain coevolution may not always be practical. Evolving for sensor input layouts in addition to the controller may give useful insight into sensors that are truly necessary to accomplish a task. Naturally, one would devise a fitness function to encourage use of a minimal set of sensors.

From the resource gathering experiments, we observe that evolutionary solutions can attain dependence to sensors and templates exposed during training but that are not necessary to complete a task, nor improve system performance. For the resource gathering task, use of a light beacon for homing to the 'dumping area' could intuitively increase efficiency of the robots in completing a task. It should also be noted that the light beacon is not necessary as the robots can also randomly forage for the dumping area. Although the ANT controllers can become dependent on the light beacon, this doesn't translate into improved system performance. One remedy to overcoming this dependence on optional components is through use of varied training conditions, such as with and without the light beacon. Other more elaborate failure scenarios could be introduced during training to 'condition' the controllers towards handling loss of components.

The use of a predefined palette of basis behaviours is accepted as an *ad hoc* procedure. Alternately, such basis behaviours could be obtained using machine learning techniques

such as artificial evolution in stages. Nevertheless, the intent with some of these experiments is to use a generic palette of basis behaviours that are not task-specific thus reducing the need for basis-behaviour-related design decisions by the experimenter. In practice, there exists a trade off between use of hand-coded basis behaviours (microscopic behaviours) versus use of evolutionary search techniques for multirobot coordination and control (resultant macroscopic behaviours). Where design and implementation of basis behaviours is more involved than determining the coordination behaviours, this approach may have limited practical value.

Our premise based on experience with multirobot controllers is that designing the basis behaviours requires much less effort than determining the unintuitive coordination schemes that give rise to emergent group coordination behaviours. To further emphasize this point, analyses of the evolved solutions indicate they are not organized according to readily 'recognizable' *distal* behaviours but as *proximal* behaviours (organized according to proximity in sensor space) (Nolfi, 1997). One of the more promising features of the evolved multirobot controllers solution is the scalability of these solutions to a larger 'world' size. In contrast, scalability may be limited with a hand-coded solution as other robots tend to be treated as obstacles to be avoided or use of heuristics for interaction requiring ad hoc assumptions based on limited knowledge of the task domain.

5. Conclusion

This chapter has reported on a number of investigations to control and coordinate groups of robots. Through four sets of experiments, we have made a case for carefully selecting a fitness function to encourage emergent (self-organized) task decomposition by evolutionary algorithms. This can alleviate the need to have a human designer arbitrarily break a group task into subtasks for each individual robot. We feel this approach fits well within the space engineering context where mission-level goals are commonly specified and the system designer must decompose these into manageable pieces. We have shown, in limited context of our experiments, that an evolutionary algorithm can take on this system design role and automatically break down a task for implementation by a group of robots. Limited supervision also has it advantages, including the ability to discover novel solutions that would otherwise be overlooked by a human supervisor. We also show that by exploiting hierarchical modularity and regulatory functionality, controllers can overcome tractability concerns.

Our findings are encouraging but we also realize that 'evolved controllers for multirobot systems' have many hurdles to overcome before adoption on a real space exploration mission. We must clearly show the advantages of (i) using multiple robots and (ii) programming them using an evolutionary algorithm. This will certainly be a long road, but we are enthused by the fact that the number of potential applications of multirobot systems in the space exploration world is quickly growing. For example, at the time of writing, there is renewed interest in the Moon and establishing a base near the lunar South Pole. Several robotic precursor missions are being planned to select and prepare the site and then deploy equipment including solar arrays. Once built, rovers will be needed to work alongside astronauts in searching for and gathering resources such as water ice. These are precisely the types of task by which we were originally motivated and we hope that evolutionary robotics will one day aid in the exploration of other worlds.

6. References

Barfoot, T.D. & D'Eleuterio, G.M.T. (1999). An Evolutionary Approach to Multiagent Heap Formation, *Proceedings of the IEEE Congress on Evolutionary Computation (CEC)*, pp. 427-435, IEEE, Washington DC.

Barfoot, T.D. & D'Eleuterio, G.M.T. (2005). Evolving Distributed Control for an Object-Clustering Task. *Complex Systems,* Vol. 15, No. 3, 183-201

Barfoot, T.D.; Earon, E.J.P. & D'Eleuterio, G.M.T. (2006). Experiments in Learning Distributed Control for a Hexapod Robot. *Robotics and Autonomous Systems*, Vol. 54, No. 10, pp. 864-872

Beckers, R.; Holland, O. E. & Deneubourg, J.L. (1994). From Local actions to Global Tasks: Stigmergy and Collective Robots, *Artificial Life IV: Proceedings of the Fourth International Workshop on the Synthesis and Simulation of Living Systems*, pp. 181-189, MIT Press, Cambridge, MA

Beer, R.D. & Gallagher, J.C. (1992). Evolving Dynamic Neural Networks for Adaptive Behavior, *Adaptive Behavior*, 1, 91-122.

Bonabeau, E., et al. (1997). Self-organization in social insects, *Trends in Ecology and Evolution*, Vol. 12, 188-193,

Bonabeau, E.; Dorigo, M. & Theraulaz, G. (1999). *Swarm Intelligence: From Natural to Artificial Systems*, Oxford University Press, New York, NY, pp. 183-205.

Bongard, J. & Pfeifer, R. (2001). Repeated structure and dissociation of genotypic and phenotypic complexity in artificial ontogeny. *Proceedings of the Genetic and Evolutionary Computation Conference*, San Francisco, CA, pp. 829-836

Chantemargue, F.; Dagaeff, T.; Schumacher, M. & Hirsbrunner B., (1996). Implicit Cooperation and Antagonism in Multi-Agent Systems, *Internal Report*, IIUF-PAI, University of Fribourg

Crabbe, F. & Dyer, M. (1999). Second-order networks for wall-building agents, *Proceedings of IEEE International Joint Conference on Neural Networks*, pp. 2178– 2183, IEEE

Cruse, H. (1990). Coordination of leg movement in walking animals, *Proceedings of the Simulation of Adaptive Behaviour: From Animals to Animats*, pp. 105-109, MIT Press

Das, R.; Crutchfield, J.P.; Mitchell, M. & Hanson, J. (1995). Evolving globally synchronized cellular automata, *Proceedings of the Sixth International Conference on Genetic Algorithms*, pp. 336-343, San Fransisco, CA, Morgan Kaufmann

Dawkins, R. (1986). *The Blind Watchmaker: Why the Evidence of Evolution Reveals a Universe without Design*, Norton & Company, New York, NY

Dellaert, F. & Beer, R. (1994). Towards an evolvable model of development for autonomous agent synthesis. *Artificial Life IV: Proceedings of the 4th International Workshop on the Synthesis and Simulation of Living Systems*, pp. 246-257, MIT Press, Cambridge, MA

Deneubourg, J. L., et al. (1991). The dynamics of collective sorting: Robot-like ants and ant-like robots, *Proceedings of the Simulation of Adaptive Behaviour: from Animals to Animats*, pp. 356-363, Paris, France, MIT Press, Cambridge, MA

Deneubourg, J.L. (1977). Application de l'ordre par fluctuations `a la description de certaines 'etapes de la construction du nid chez les termites, *Insectes Sociaux*, Vol. 24, pp. 117-130

Dorigo, M. & Colombetti, M. (1998). *Robot Shaping: An Experiment in Behavior Engineering*, MIT Press, Cambridge, MA

Earon, E.J.P.; Barfoot, T.D. & D'Eleuterio, G.M.T. (2000). From the Sea to the Sidewalk: The Evolution of Hexapod Walking Gaits by a Genetic Algorithm, *Proceedings of the International Conference on Evolvable Systems*, pp. 51-60, Edinburgh, Scotland

Earon, E. J. P.; Barfoot, T.D. & D'Eleuterio, G.M.T. (2001). Development of a Multiagent Robotic System with Application to Space Exploration, *Proceedings of the International Conference on Advanced Intelligent Mechatronics*, pp. 1267-1272, Como, Italy, IEEE, Washington, D.C.

Federici, D. & Downing, K. (2006). Evolution and Development of a Multicellular Organism: Scalability, Resilience, and Neutral Complexification, *Artificial Life,* Vol. 12, pp. 381–409

Grassé, P. (1959) La reconstruction du nid les coordinations interindividuelles; la theorie de stigmergie, *Insectes Sociaux*, Vol. 35, pp. 41–84

Groß, R. & Dorigo, M. (2003). Evolving a Cooperative Transport Behavior for Two Simple Robots, *Proceedings of the 6th International Conference on Artificial Evolution*, pp. 305-317, Springer-Verlag, Berlin

Gruau, F., (1995). Automatic definition of modular neural networks, *Adaptive Behaviour*, Vol. 3, No. 2, pp. 151–184.

Jacob, R.; Jordan, M. & Barto, A. (1991). Task decomposition through competition in a modular connectionist architecture. *Cognitive Science*, Vol. 15, pp. 219-250

Kitano, H. (1990). Designing Neural Networks using genetic Algorithm with Graph Generation System, *Complex Systems*, Vol. 4, pp. 461-476

Kube, R. & Zhang, H. (1996). The use of perceptual cues in multirobot box-pushing. *Proceedings of the IEEE International Conference on Robotics and Automation*, pp. 2085-2090, IEEE, Washington DC

Matarić, M.J. et al. (1995). Cooperative Multi-Robot Box Pushing, *Proceedings of IEEE International Conference on Intelligent Robots and Systems*, pp. 556-561, IEEE, Washington DC

Matarić, M.J. (1997). Behaviour-based control: Examples from navigation, learning, and group behaviour, in Software Architectures for Physical Agents, *Journal of Experimental and Theoretical Artificial Intelligence*, Vol. 9, No. 2, pp. 232–336

Mitchell, M.; Crutchfield, J. & Das, R. (1996). Evolving cellular automata with genetic algorithms, a review of recent work. *Proceedings of the 1st International Conference on Evolutionary Computation and Its Applications*, Russian Academy of Sciences

Nolfi, S. & Floreano, D. (2000). *Evolutionary Robotics : The Biology, Intelligence, and Technology of Self-Organizing Machines*, MIT Press, Cambridge, MA, pp. 13-15

Nolfi, S. (1997). Using emergent modularity to develop control systems for mobile robots. *Adaptive Behavior*, Vol. 3, No. 4, pp. 343-364

Parker, C.; Zhang, H. & Kube, R. (2003). Blind Bulldozing: Multiple Robot Nest Construction, *Proceedings of the 2003 IEEE International Conference on Intelligent Robots and Systems,* pp. 2010-2015, IEEE, Washington DC

Parker, G. & Li, Z. (2003). Evolving Neural Networks for Hexapod Leg Controllers, *Proceedings of the 2003 IEEE International Conference on Intelligent Robots and Systems*, pp. 1376- 1381, IEEE, Washington DC

Porcino, N. (1990). Hexapod Gait Control by Neural Network, *Proceedings of the International Joint Conference on Neural Networks,* pp. 189-194, San Diego, CA, 1990

Roggen D. & Federici D. (2004). Multi-cellular Development: Is There Scalability and Robustnes to Gain?, *Proceedings of 8th Parallel Problem Solving from Nature Conference*, pp. 391-400, Springer-Verlag, Berlin

Sims, K. (1994). Evolving 3D Morphology and Behavior by Competition, *Artificial Life IV: Proceedings of the 4th International Workshop on the Synthesis and Simulation of Living Systems*, pp. 28-39, MIT Press, Cambridge, MA

Stanley, K. & Miikkulainen, R. (2002). Continual Coevolution through Complexification, *Proceedings of the Genetic and Evolutionary Computation Conference*, pp. 113-120, Morgan Kaufmann, San Francisco, CA

Stone, P. & Veloso, M. (2000). Layered Learning, *Proceedings of the 11th European Conference on Machine Learning*, pp. 369-381

Stewart, R. & Russell, A. (2003). Emergent structures built by a minimalist autonomous robot using a swarm- inspired template mechanism, *Proceedings of the 1st Australian Conference on Artificial Life*, pp. 216-230

Thangavelautham, J. & D'Eleuterio, G.M.T. (2004). A Neuroevolutionary Approach to Emergent Task Decomposition, *Proceedings of the 8th Parallel Problem Solving from Nature Conference*, pp. 991-1000, Springer-Verlag, Berlin

Thangavelautham, J.; Barfoot, T.D. & D'Eleuterio G.M.T. (2003). Coevolving Communication and Cooperation for Lattice Formation Tasks, *Advances In Artificial Life: Proceedings of the 7th European Conference on Artificial Life*, pp. 857-864, Springer, Berlin

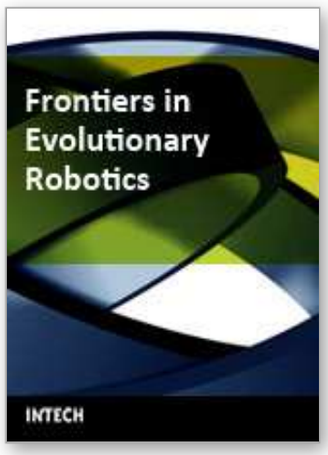
Thangavelautham, J. & D'Eleuterio, G.M.T. (2005). A Coarse-Coding Framework for a Gene-Regulatory-Based Artificial Neural Tissue, *Advances In Artificial Life: Proceedings of the 8th European Conference on Artificial Life*, pp. 67-77, Springer, Berlin

Thangavelautham, J.; Smith, A.; Boucher, D.; Richard, J. & D'Eleuterio, G.M.T. (2007). Evolving a Scalable Multirobot Controller Using an Artificial Neural Tissue Paradigm, *Proceedings of the IEEE International Conference on Robotics and Automation*, pp. 77-84, IEEE, Washington DC

Thomas, L. (1974). *The Lives of a Cell: Notes of a Biology Watcher,* Penguin, USA

Wawerla, J.; Sukhatme, G. & Mataric, M. (2002). Collective construction with multiple robots, *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 2696-2701, IEEE, Washington, DC

Wilson, M.; Melhuish, C.; Franks, A. & Scholes, S. (2004). Algorithms for building annular structures with minimalist robots inspired by brood sorting in ant colonies, *Autonomous Robots*, Vol. 17, pp. 115–136



Frontiers in Evolutionary Robotics

Edited by Hitoshi Iba

ISBN 978-3-902613-19-6

Hard cover, 596 pages

Publisher I-Tech Education and Publishing

Published online 01, April, 2008

Published in print edition April, 2008

This book presented techniques and experimental results which have been pursued for the purpose of evolutionary robotics. Evolutionary robotics is a new method for the automatic creation of autonomous robots. When executing tasks by autonomous robots, we can make the robot learn what to do so as to complete the task from interactions with its environment, but not manually pre-program for all situations. Many researchers have been studying the techniques for evolutionary robotics by using Evolutionary Computation (EC), such as Genetic Algorithms (GA) or Genetic Programming (GP). Their goal is to clarify the applicability of the evolutionary approach to the real-robot learning, especially, in view of the adaptive robot behavior as well as the robustness to noisy and dynamic environments. For this purpose, authors in this book explain a variety of real robots in different fields. For instance, in a multi-robot system, several robots simultaneously work to achieve a common goal via interaction; their behaviors can only emerge as a result of evolution and interaction. How to learn such behaviors is a central issue of Distributed Artificial Intelligence (DAI), which has recently attracted much attention. This book addresses the issue in the context of a multi-robot system, in which multiple robots are evolved using EC to solve a cooperative task. Since directly using EC to generate a program of complex behaviors is often very difficult, a number of extensions to basic EC are proposed in this book so as to solve these control problems of the robot.

Fax: +385 (51) 686 166
www.intechopen.com

Fax: +86-21-62489821

IntechOpen