Application of Coarse-Coding Techniques for Evolvable Multirobot Controllers

Jekanthan Thangavelautham, Paul Grouchy and Gabriele M.T. D'Eleuterio

Abstract Robots, in their most general embodiment, can be complex systems trying to negotiate and manipulate an unstructured environment. They ideally require an 'intelligence' that reflects our own. Artificial evolutionary algorithms are often used to generate a high-level controller for single and multi robot scenarios. But evolutionary algorithms, for all their advantages, can be very computationally intensive. It is therefore very desirable to minimize the number of generations required for a solution. In this chapter, we incorporate the Artificial Neural Tissue (ANT) approach for robot control from previous work with a novel Sensory Coarse Coding (SCC) model. This model is able to exploit regularity in the sensor data of the environment. Determining how the sensor suite of a robot should be configured and utilized is critical for the robots operation. Much as nature evolves body and brain simultaneously, we should expect improved performance resulting from artificially evolving the controller and sensor configuration in unison. Simulation results on an example task, resource gathering, show that the ANT+SCC system is capable of finding fitter solutions in fewer generations. We also report on hardware experiments for the same task that show complex behaviors emerging through self-organized task decomposition.

1 Introduction

Our motivation for evolutionary-based control approaches for multirobot systems originates in the use of robots for space exploration and habitat construction on

Jekanthan Thangavelautham
Mechanical Engineering Dept., Massachusetts Institute of Technology, 77 Massachusetts Ave., Cambridge, MA, USA, 02139
Paul Grouchy and Gabriele M.T. D'Eleuterio
Institute for Aerospace Studies, University of Toronto, 4925 Dufferin St., Toronto, Canada, M3H5T6
e-mail: jthanga@mit.edu, {paul.grouchy, gabriele.deleuterio}@utoronto.ca

alien planets and planetoids, such as Mars and the Moon. Potential scenarios include establishing a distributed antenna for communications, deploying a mobile array of actuators and sensors for geological measurements, or constructing elements of an outpost in preparation for the arrival of humans.

These kinds of project call for not just a single monolithic robotic system but teams of elemental robots working in collaboration and coordination. While space applications may require the use of a multiagent strategy, they are by no means the only ones. Consider, for example, terrestrial applications such as search and rescue, mapping, manufacturing and construction.

A number of factors make the team approach viable and attractive. Among them are increased reliability. One can afford to lose a member of the team without destroying the team's integrity. A team approach can offer increased efficiency through parallelization of operations. As such, multiagent systems are more readily scalable. Most important, however, a team can facilitate task decomposition. A complex task can be parsed into manageable subtasks which can be delegated to multiple elemental units.

Robotic systems can themselves be complex and their environments are generally unstructured. Sound control strategies are therefore not easy to develop. A methodical approach is not only desired but arguably required. It would be ideal if the controller could be automatically generated starting from a 'blank slate,' where the designer is largely relieved of the design process and detailed models of the systems or environment can be avoided. It is by such a road that we have come to the use of evolutionary algorithms for generating controllers that are based on neural-network architectures.

We have developed and tested, both in simulation and hardware, a neuroevolutionary approach called the Artificial Neural Tissue (ANT). This neural-network-based controller employs a variable-length genome consisting of a regulatory system that dictates the rate of morphological growth and can selectively activate and inhibit neuron ensembles through a coarse-coding scheme [31]. The approach requires an experimenter to specify a goal function, a sensory input layout for the robots and the repertoire of allowable basis behaviors. The control topology and its contents emerge through the evolutionary process.

But is it possible to evolve, in addition to the controllers themselves, the necessary sensor configurations and the selection of behavior primitives (motor-actuator commands) concurrently with the evolution of the controller? It is this question that we address in this work.

In tackling this challenge, we turn to a key theme in our ANT concept, namely, coarse coding. Coarse coding is an efficient, distributed means of representation that involves the use of multiple coarse receptive fields to represent a higher-resolution field. As is well known, nature exploits coarse coding in the brain and sensory systems. In artificial systems, coarse coding is used to interpret data. In ANT, however, it is the program (the artificial neural structure responsible for computation) that is coarse coded. This allows the development of spatially modular functionality in the architecture that mimics the modularity in natural brains.

We present in this work a Sensory Coarse Coding (SCC) model that extends capabilities of ANT and allows for the evolution of sensor configuration and coupled motor primitives. The evolution of sensor configuration and behavior primitives can be used to take advantage of regularities in the task space and can help to guide and speed up the evolution of the controller.

Training neural network controllers using an evolutionary algorithms approach like ANT for robotics is computationally expensive and tends to be used when other standard search algorithms like gradient descent are unsuitable or require substantial supervision for the given task space. The controllers are developed using a biologically motivated development process and replicated on one or more robotic platforms for evaluation. Training on hardware is often logistically difficult, requiring a long-term power source and a means of automating the controller evaluation process. The alternative is to simulate the robotic evaluation process on one or more computers. The bulk of the required time in training is the evaluation process. Genetic operations including selection, mutation and crossover tend to take less than one percent of the computational time. Therefore any method that can reduce the number of genetic evaluations will have a substantial impact on the training process. Furthermore, a significant reduction in the number of generations required can also make the training process feasible on hardware. Robotic simulations often take into account the dynamics and kinematics of robotic vehicle interactions. However, care has to be taken to ensure the simulation environment resembles or is compatible with actual hardware. In other circumstances, it may not be beneficial to prototype and demonstrate capabilities and concepts in simulation before proceeding towards expensive hardware demonstration. With robotics application on the lunar surface, the low gravity environment cannot be easily replicated on earth and hence high fidelity dynamics simulation maybe need to demonstrate aspects of system capability.

For multirobotic tasks, the global effect of local interactions between robots is often difficult to gauge, and the specific interactions required to achieve coordinated behavior may even be counterintuitive. Furthermore, it is not at all straightforward to determine the best sensor configuration. Often detailed analysis of the task needs to be performed to figure out the necessary coordination rules and sensory configurations. The alternatives is to use optimization techniques to in effect shrink the search space sufficiently to enable evolutionary search algorithms to find suitable solutions. Evolving this configuration may also give useful insight into the sensors necessary for a task. This may help guide a robotic designer in their design processes—we do not presume to make the designer completely redundant—by helping them determine which sensors and actuators are best to achieve a given objective. In addition, the evolution of the sensor configuration in conjunction with the controller would allow us to mimic nature more closely. Nature perforce evolves body and brain together.

The remainder of this chapter is organized as follows. First, we provide background to our problem by reviewing past work on the use of evolutionary algorithms for the development of multirobot controllers and on ‘body and brain’ evolution. We present the workings of the Artificial Neural Tissue approach followed by the Sensory Coarse Coding model. We refer to the integration of the latter into the former

as the ANT+SCC system. Next, we report on a number of experiments we have conducted to demonstrate the ANT+SCC system on a group of robots, concentrating on an example of resource gathering. This is followed by a discussion of the findings and finally we venture some concluding remarks.

2 Background

Coordination and control of multirobot systems are often inspired by biology. 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 [3]. In insect colonies, templates may be a natural phenomenon or they may be created by the colonies themselves. They may include temperature, humidity, chemicals, or light gradients. *Stigmergy* is a form of indirect communication mediated through the environment [12]. One way in which ants and termites exploit stigmergy is through the use of pheromone trails. Self-organization describes how local or microscopic behaviors give rise to a macroscopic structure in systems [2]. However, many existing approaches suffer from another emergent feature called *antagonism* [5]. This is the effect that arises when multiple agents trying to perform the same task interfere with each other and reduce the overall efficiency of the group.

Within the field of robotics, many have sought to develop multirobot control and coordination behaviors based on one or more of the prescribed mechanisms used in nature. These solutions have been developed using user-defined deterministic 'if-then' rules or preprogrammed stochastic behaviors. Such techniques in robotics include template-based approaches that exploit light fields to direct the creation of walls [33] and planar annulus structures [34]. Stigmergy has been used extensively in collective-robotic construction tasks, including blind bull dozing [24], box pushing [21] and heap formation [1].

Inspired by insect societies, the robots are equipped with the necessary sensors required to demonstrate multirobot control and coordination behaviors. Furthermore, the robot controllers are often designed by hand 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 very difficult 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 [24], [1]. Other more sophisticated techniques involve the use of explicit communication or designing an extra set of coordination rules to gracefully handle agent-to-agent interactions [33]. These approaches are largely heuristic, rely on ad hoc assumptions that often require knowledge of the task domain and are implemented with a specified robot configuration in mind.

2.1 The Body and the Brain

The ecological balance principle [25] states that the complexity of an 'agent' must match the complexity of the task environment. In the natural world, this matching is done by the evolutionary process which molds both the body and the brain of individual organisms for survival. Adaptive systems may evolve to exploit regularities in the task environment that concurrently impact the physical design and control. This has been demonstrated with artificial evolution of both body and brain of artificial creatures. Early work by Sims [27] involved breeding virtual creatures that could swim and hop in a simulated 3-D environment. A generative encoding system was used to describe the phenotype and was based on Lindenmayer's L-system [19]. The creatures were fully described by the genome and composed of a hierarchical description outlining three-dimensional rigid parts and various joints. Reactive control laws were also evolved that described the interaction of the various parts.

Framsticks, another method of evolving artificial body-brain system, combined a neural-network controller with a specified morphology [17]. The resultant virtual creatures were evolved to perform locomotion in various environments, both on land and in water. Grammar-based techniques have also been used to evolve brain and bodies for robotic applications [22] and have demonstrated simple braintenberg tasks. Work by Lipson and Pollack [20] involved the evolution of robot designs and actuators that were realized by a 3-D printer. Standard components such as motors were added allowing the system to perform locomotion. Further work in this area by Zykov et al. [35] has been directed towards evolving self-replicating machine configurations using a physical substrate.

In robotics, there are both potential advantages and disadvantages when designing both body and brain concurrently for solving individual tasks. One advantage is that the end design may be specifically tuned towards solving a specialized task in an efficient manner (equivalent to finding a niche in nature). However, this may be at the cost of losing multipurpose capabilities. On the other hand, specific needs and performance considerations may warrant the need for special-purpose robots.

In many practical situations, one may not have the resources necessary to design specialized robots for a task at hand. Hence one has to use standard robot configurations and implement a controller for this configuration. However, it may be practical to reconfigure placement of sensors on a standard robot configuration for use on a specified task. How best to configure these sensors and implement the associated controllers remains a crucial question in multirobot systems.

2.2 Task Decomposition

Sensor configuration is particularly key to task decomposition. The ability to partition or segment a complex task into subtasks is a vital capability for both natural and artificial systems. Part of solving a real-world task involves sensing the environment and using it to provide feedback when performing actions.

One of the advantages of multirobot systems, as mentioned above, is precisely the opportunity to facilitate task decomposition. With the use of such systems, control and coordination are critical. Both the control and coordination capabilities are dependent on the individual robot's ability to sense its environment and its ability to perform actions.

2.3 Machine-Learning Techniques and Modularization

Machine-learning techniques, particularly artificial evolution, exploit self-organization and relieve the designer of having to determine a suitable control strategy and sensor configuration. Using these techniques, controllers and sensor configurations develop cooperation and interaction strategies by setting the evolving system loose in the environment. By contrast, it is difficult to design controllers by hand with cooperation in mind because it is difficult, given the complexity of the system as well as the environment, to predict or control the global behaviors that will result from local interactions.

One machine-learning technique to overcome the difficulties of design by hand is based on Cellular Automata (CA) look-up tables. A genetic algorithm can be used to evolve the table entries [7]. The assumption is that each combination of sensory inputs will result in a particular choice of output behaviors. This approach is an instance of a 'tabula rasa' technique. The control system starts off as a blank slate with limited assumptions regarding control architecture and is guided through training by a fitness function (system goal function). Such approaches can be used to obtain robust, scalable controllers that exploit multirobot mechanisms such as stigmergy and self-organization. Furthermore, these approaches are beneficial for hardware experiments as there is minimal computational overhead incurred, especially if onboard sensor processing is available.

One of the limitations with a look-up table approach is that the table size grows exponentially with the number of inputs. For a 3×3 tiling formation task, a single look-up table architecture is found to be intractable owing to premature search stagnation [29]. To address this limitation, the controller can be modularized into subsystems by exploiting regularities in the task environment. These subsystems can explicitly communicate and coordinate actions with other agents. This act of dividing the agent functionality into subsystems is a form of user-assisted task decomposition. Such intervention requires domain knowledge of the task and ad hoc design choices to facilitate searching for a solution.

Use of neural networks is also another form of modularization, where each neuron can communicate and perform some form of sensory information processing. The added advantage of neural-network architectures is that the neurons can generalize (unlike CAs) by recognizing correlations between a combination of sensory inputs, thus effectively shrinking the search space. Fixed-topology neural-network architectures have been used extensively for multirobot tasks, including building walls [33], tile formation [30], and cooperative transport [13]. However, monolithic

fixed-topology neural-network architectures also face scalability problems. With an increasing number of hidden neurons, one must contend with the effects of *spatial crosstalk* where noisy neurons interfere and drown out signals from feature-detecting neurons [16].

Crosstalk in combination with limited supervision (through use of a global fitness function) can lead to the 'bootstrap problem' [23], where evolutionary algorithms are unable to pick out incrementally fitter solutions 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 a task or is difficult to train [31].

2.4 Fixed versus *Variable Topologies*

Fixed-topology architectures accordingly have limitations, particularly in robotics, for the very reason that the topology must be determined a priori and there is no opportunity to modifying it without starting over. However, variable-topology architectures allow for the evolution of both the network architecture and the neuronal weights simultaneously. The genotypes for these systems are encoded in a one-to-one mapping such as in Neuro-Evolution of Augmenting Topologies (NEAT) [28]. The use of recursive rewriting of the genotype contents to a produce a phenotype such as in Cellular Encoding [14], L-systems [27], or through artificial ontogeny [8]. 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 [4] and Developmental Embryonal Stages (DES) [10].

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 regulates (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. These variable topologies also have to be grown incrementally starting from a single cell in order to minimize the dimensional search space as the size of the network architecture may inadvertently make training difficult [28]. 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 once again in premature search stagnation [26].

The Artificial Neural Tissue concept [31] is intended to address limitations evident with existing variable topologies through the modeling of a number of biologically plausible mechanisms. 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 the expression of incremental innovations [10].

2.5 Regularity in the Environment

Most of the developmental systems described above deal with techniques used to facilitate evolution of network topologies. A critical advantage of the neural-network approach is its ability to generalize. Generalization often relies on regularities and patterns in the sensory input space to be effective. However the sensor configuration used by these developmental controllers still need to be specified by the experimenter. As discussed earlier, in a multirobot environment, it is often counterintuitive as to what control rules are necessary to obtain desired global behaviors. Furthermore, it is difficult to know what sensor configuration is necessary to facilitate these control rules. A number of techniques including the one presented here attempt to address this limitation. By allowing for the evolutionary search process to modify sensor configuration and geometry, it expected that this will facilitate finding efficient solutions with fewer genetic evaluations.

Blondie24 [6], an evolved checkers-playing neural network is one of the early efforts to exploit regularity in task space. A standard fixed neural-network topology was designed to take into account the regularity of the checkerboard. Hidden nodes of the network were tied to subsquares (cell regions arranged in square shape). The resultant network was coevolved with real players on an Internet checkers server and reached an expert level of proficiency in the game.

HyperNEAT extends NEATs capabilities by combining a variable neural-network topology with a hypercube-based generative description of the topology [11]. Instead of encoding for every weight in the network separately in the genome, HyperNEAT uses a type of Compositional Pattern Producing Network (CPPN) to produce a network that represents the weight (connectivity) parameters of the phenotype network (controller). It also allows for the CPPNs to represent symmetries and regularities from the geometry of the task inputs directly in the controller.

Geometric regularities can also be extracted using coarse-coding techniques. Coarse coding allows for the partitioning of separate geometric locations and thus allows for each part to be learned separately through task decomposition [11]. While this is advantageous it has also been argued that it may prevent the learning system from discovering interdimensional regularities and alternate approaches have been shown to overcome this limitation using a priori knowledge [18]. However, as we show here, this capability can also be evolved.

ANT+SCC extends the ANT approach with a sensor mapping and filtering scheme. This functionality is performed by a group of sensory neurons that interact in a coarse-coding fashion. This interaction helps determine resultant sensor geometry and resolution and aids in the filtering process. The output from these sensor neurons feeds into an ANT controller where higher-level processing is performed. This laminar scheme bears some resemblance in functionality to how the visual cor-

tex in the mammalian brain operates with lower layers performing sensory filtering such as edge and line detection which in turn is used by higher-level functionality to make, for example, optical flow measurements.

3 Artificial Neural Tissue Model

The ANT architecture (Fig. 1a) presented in this paper consists of a developmental program, encoded in the "genome," that constructs a three-dimensional neural tissue and associated regulatory functionality. The tissue consists of two types of neural units, decision neurons and motor-control neurons, or simply motor neurons. Regulation is performed by the decision neurons that dynamically exhibit or inhibit motor-control neurons within the tissue based on a coarse-coding techniques. The following sections discuss the computational aspects of the tissue and how it is created.

3.1 Computation

We imagine the motor neurons of our network to be spheres arranged in a regular rectangular lattice in which the neuron N_λ occupies the position $\lambda = (l,m,n) \in \mathbb{I}^3$ (sphere centered within cube). The state s_λ of the neuron is binary, i.e., $s_\lambda \in S = \{0,1\}$. Each neuron N_λ nominally receives inputs from neurons N_κ where $\kappa \in \Uparrow(\lambda)$, the nominal input set. Here we shall assume that these nominal inputs are the 3×3 neurons centered one layer below N_λ; in other terms, $\Uparrow(\lambda) = \{(i,j,k) \,|\, i = l-1,l,l+1;\ j = m-1,m,m+1;\ k = n-1\}$. (As will be explained presently, however, we shall not assume that all the neurons are active all the time.) The activation function of each neuron is taken from among four possible threshold functions of the weighted input σ:

$$\begin{aligned}
\psi_{\text{down}}(\sigma,\theta_1) &= \begin{cases} 0, \text{ if } \sigma \geq \theta_1 \\ 1, \text{ otherwise} \end{cases} \\
\psi_{\text{up}}(\sigma,\theta_2) &= \begin{cases} 0, \text{ if } \sigma \leq \theta_2 \\ 1, \text{ otherwise} \end{cases} \\
\psi_{\text{ditch}}(\sigma,\theta_1,\theta_2) &= \begin{cases} 0, \min(\theta_1,\theta_2) \leq \sigma < \max(\theta_1,\theta_2) \\ 1, \text{ otherwise} \end{cases} \\
\psi_{\text{mound}}(\sigma,\theta_1,\theta_2) &= \begin{cases} 0, \sigma \leq \min(\theta_1,\theta_2) \text{ or } \sigma > \max(\theta_1,\theta_2) \\ 1, \text{ otherwise} \end{cases}
\end{aligned} \tag{1}$$

The weighted input σ_λ for neuron N_λ is nominally taken as

$$\sigma_\lambda = \frac{\sum_{\kappa \in \Uparrow(\lambda)} w_\lambda^\kappa s_\kappa}{\sum_{\kappa \in \Uparrow(\lambda)} s_\kappa} \tag{2}$$

with the proviso that $\sigma = 0$ if the numerator and denominator are zero. Also, $w_\lambda^\kappa \in \mathbb{R}$ is the weight connecting N_κ to N_λ. We may summarize these threshold functions in a single analytical expression as

$$\psi \;=\; (1 - k_1)[(1 - k_2)\psi_{\text{down}} + k_2\psi_{\text{up}}] + k_1[(1 - k_2)\psi_{\text{ditch}} + k_2\psi_{\text{mound}}] \tag{3}$$

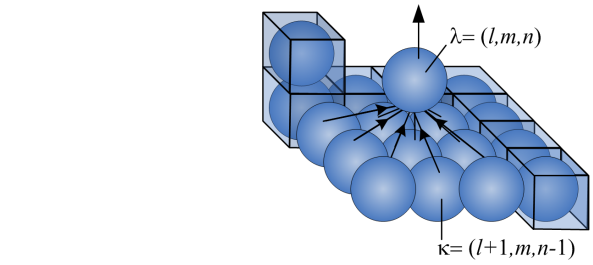
where k_1 and k_2 can take on the value 0 or 1. The activation function is thus encoded in the genome by k_1, k_2 and the threshold parameters $\theta_1, \theta_2 \in \mathbb{R}$.

It may appear that ψ_{down} and ψ_{up} are mutually redundant as one type can be obtained from the other by reversing the signs on all the weights. However, retaining both increases diversity in the evolution because a single 2-bit "gene" is required to encode the threshold function and only one mutation suffices to convert ψ_{down} into ψ_{up} or *vice versa* as opposed to changing the sign of every weight.

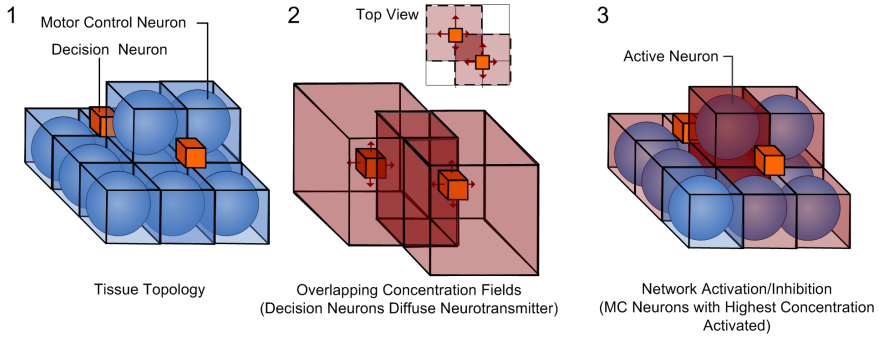
The sensor data are represented by the activation of the sensor input neurons $N_{\alpha_i}, i = 1 \ldots m$, summarized as $A = \{s_{\alpha_1}, s_{\alpha_2} \ldots s_{\alpha_m}\}$. Similarly, the output of the network is represented by the activation of the output neurons $N_{\omega_j}, j = 1 \ldots n$, summarized as $\Omega = \{s_{\omega_1^1}, s_{\omega_2^2} \ldots s_{\omega_n^b}\}$, where $k = 1 \ldots b$ specifies the output behavior. Each output neuron commands one behavior of the agent. (In the case of a robot, a typical behavior may be to move forward a given distance. This may involve the coordinated action of several actuators. Alternatively, the behavior may be more primitive such as augmenting the current of a given actuator.) If $s_{\omega_j^k} = 1$, output neuron ω_j votes to activate behavior k; if $s_{\omega_j^k} = 0$, it does not. Since multiple neurons can have access to a behavior pathway, an arbitration scheme is imposed to ensure the controller is deterministic where $p(k) = \sum_{s_\diamond, j=1}^{n_k} s_{\omega_j^k}/n_k$ and n_k is the number of output neurons connected to output behavior k resulting in behavior k being activated if $p(k) \geq 0.5$.

As implied by the set notation of Ω, the outputs are not ordered. In this embodiment, the order of activation is selected randomly. We are primarily interested here in the statistical characteristics of relatively large populations but such an approach would likely not be desirable in a practical robotic application. However this can be remedied by simply assigning a sequence *a priori* to the activations (as shown in Table 2 for the resource gathering task).

We moreover note that the output neurons can be redundant; that is, more than one neuron can command the same behavior, in which case for a given time step one behavior may be "emphasized" by being voted multiple times. Neurons may also cancel out each other.



(a) Synaptic connections



(b) Coarse-coding

Fig. 1 Synaptic connections between motor neurons and operation of neurotransmitter field.

3.2 The Decision Neuron

The coarse-coding nature of the artificial neural tissue is provided by the decision neurons. Decision neurons can be thought of as rectangular structures occupying nodes in the lattice as established by the evolutionary process (Fig. 1). The effect of these neurons is to excite into operation or inhibit (disable) the motor control neurons (shown as spheres). Once a motor control neuron is excited into operation, the computation outlined in (2) is performed. Motivated as we are to seek biological support for ANT, we may look to the phenomenon of chemical communication among neurons. In addition to communicating electrically along axons, some neurons release chemicals that are read by other neurons, in essence serving as a "wireless" communication system to complement the "wired" one.

In ANT, the state of a decision neuron T_μ, μ is binary and determined by one of the same activation functions 1 that also embedded within the motor control neurons. The inputs to T_μ are all the input sensor neurons N_α; *i.e.*, $s_\mu = \psi_\mu(s_{\alpha_1} \ldots s_{\alpha_m})$ where $\sigma_\mu = \sum_\alpha v_\alpha^\mu s_\alpha / \sum_\alpha s_\alpha$ and v_α^μ are the weights. The decision neuron is dormant if $s_\mu = 0$ and releases a virtual neurotransmitter chemical of uniform concentration

c_μ over a prescribed field of influence if $s_\mu = 1$. Motor control neurons within the highest chemical concentration field are excited into operation. Only those neurons that are so activated will establish the functioning network for the given set of input sensor data. Owing to the coarse-coding effect, the sums used in the weighted input of 1 are over only the set $\overline{\Uparrow}(\lambda) \subseteq \Uparrow(\lambda)$ of active inputs to N_λ. Likewise the output of ANT is in general $\overline{\Omega} \subseteq \Omega$. The decision neuron's field of influence is taken to be a rectangular box extending $\pm d_\mu^r$, where $r = 1,2,3$, from μ in the three perpendicular directions. These three dimensions along with μ and c_μ, the concentration level of the virtual chemical emitted by T_μ, are encoded in the genome.

Tissue Gene

Specifier	Neuron Replication Prob.	Neuron Replication Ratios		Seed Address
D	T_r	n_d	n_m	T_s
Integer [0,2]	Real [0.001,0.1]	Integers [1,10]		Integer

Decision Neuron Gene

Specifier	Reference Address	Position			Diffusion Param.			Diffusion Concentration	Activation Function Parameters							Gene Activate
D	A	x	y	z	d_x	d_y	d_z	c	w_1	w_2	...	w_v	θ_1	θ_2	k_1	k_2	G
Integer [0,2]	Integer	Integers			Integers [1,3]			Integers [0,1]	Real [0,1]						Integers [0,1]	Binary

Motor Control Neuron Gene

Specifier	Reference Address	Position			Activation Function Parameters							Gene Activate	Cell Death	Replication Prob.	Output Behaviour	Reference Pointer
D	A	x	y	z	w_1	w_2	...	w_v	θ_1	θ_2	k_1	k_2	G	C	R	k	P
Integer [0,2]	Integer	Integers			Real [0,1]						Integers [0,1]		Binary	Binary	Real [0,1]	Integer [0,b]	Integer

Fig. 2 Gene map for the Artificial Neural Tissue

3.3 Evolution and Development

A population of ANT controllers is evolved in an artificial Darwinian manner. The "genome" for a controller contains a "gene" for each cell with a specifier D that is used to distinguish the functionality (between motor control, decision and tissue). A constructor protein (an autonomous program) interprets the information encoded in the gene and translates this into a cell descriptor protein (see Fig. 2). The gene "activation" parameter is a binary flag resident in all the cell genes and is used to either express or repress the contents of gene. When repressed, a descriptor protein of the gene content is not created. Otherwise, the constructor protein "grows" the tissue in which each cell is located relative to a specified seed-parent address. A cell death flag determines whether the cell commits suicide after being grown. Once again, this feature in the genome helps in the evolutionary process for a cell, by committing suicide, still occupies a volume in the lattice although it is dormant. In otherwise retaining its characteristics, evolution can decide to reinstate the cell by merely toggling a bit.

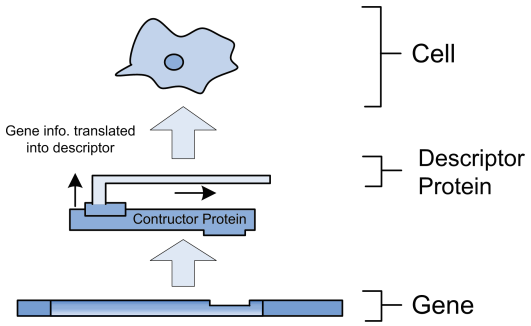


Fig. 3 Genes are "read" by constructor proteins that transcribe the information into a descriptor protein which is used to construct a cell. When a gene is repressed, the constructor protein is prevented from reading the gene contents.

In turn mutation (manipulation of gene parameters with uniform random distribution) to the growth program results in new cells being formed through cell division. The rate at which mutation occurs to a growth program is also specified for each tissue and is dependent on the neuron replication probability parameter. Cell division requires a parent cell (selected with highest replication probability relative to the rest of the cells within the tissue) and involves copying $m\%$ of the original cell contents to a daughter cell (where m is determined based on uniform random distribution), with the remaining cell contents initialized with a uniform random distribution. The cell type of each new cell is determined based on the ratio of motor control to decision neurons specified in the tissue gene. The new cell can be located in one of six neighboring locations (top, bottom, north, south, east, west) sharing a common side with the parent and is not occupied by another cell.

3.4 Sensory Coarse Coding Model

In this section we present the Sensory Coarse Coding (SCC) model. The model includes two components that allow for filtering and mapping locations of sensory inputs. Sensory Coarse Coding provides additional functionality not found within the ANT model, namely the ability to search for spatial mappings of sensory inputs while simultaneously filtering these inputs for further processing. Biological motivation for this capability comes from analyzing the visual cortex.

Within the tissue architecture, we include one additional type of neuron, the sensor neuron. There exists a group of v sensory neurons $\Pi = [\Phi_{\tau^1}, \Phi_{\tau^2} \ldots \Phi_{\tau^v}]$, where each neuron has a position $\tau = (l,m)$, $l \in [0,h-1]+0.5$, $m \in [0,h-1]+0.5$ on a spatial map spanning $h \times h$ grid squares and representing the agent/robot and its surroundings (Fig. 4 left). The state s_τ of a sensor neuron can assume one of up to q states, i.e., $s_\tau \in A' = \{s_{\alpha_1}, s_{\alpha_2} \ldots s_{\alpha_q}\}$.

Each neuron Φ_τ receives input from spatial map locations L_φ, where $\varphi \in \Uparrow(\tau)$, the input set. Each grid square L_φ assumes a sensor reading, one of q states, i.e., $L_\varphi \in A'$. Here we shall assume that the receptive field for this sensor neuron is a bounded area containing $a_\tau \times b_\tau$ grid squares centered at (l,m), i.e., $\Uparrow(\tau) = \{(i,j) \,|\, i \in [l - a_\tau/2, l + a_\tau/2], j \in [m - b_\tau/2, m + b_\tau/2]\}$. The sensory neurons are not necessarily fed their entire input set. A coarse coding system is used to decide which inputs, if any, a sensory neuron will receive. Each sensory neuron emits a stimulus chemical in the area $\Uparrow(\tau)$ such that the amount of chemical diffused at location φ due to sensory neuron Φ_{τ^i} is the following:

$$c_{\varphi,\tau^i} = \begin{cases} 1, & \text{if } \varphi \in \Uparrow(\tau^i) \\ 0, & \text{otherwise} \end{cases} \tag{4}$$

Therefore, the net concentration of chemical diffused due to the v sensory neurons at φ is:

$$c_\varphi = \sum_{i=1}^{v} c_{\varphi,\tau^i} \tag{5}$$

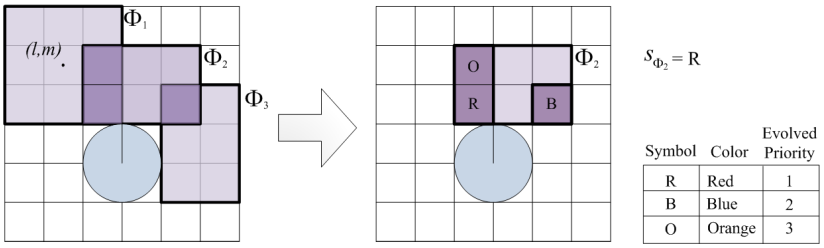


Fig. 4 (Left) Three sensor neurons and their respective receptive field shown shaded. With $S = 1$, only Φ_2 is selected since $\sum_{\varphi \in \Uparrow(\tau)} c_\varphi$ is the highest among the three. (Right) Once Φ_2 is activated, only locations with the highest chemical concentrations (shaded in dark gray) are fed as inputs to the evolved priority filter. The result is a single output from the neuron, indicating red.

To determine which grid squares a sensory neuron Φ_τ will receive, the chemical concentration at each location $\varphi \in \Uparrow(\tau)$ is calculated. The states of the locations L_φ that have a maximum chemical concentration in the grid are fed to the sensory neuron inputs. If $\sum_{\varphi \in \Uparrow(\tau)} c_\varphi = 0$, sensory neuron Φ_τ is deactivated. Furthermore, S sensor neurons with the highest $\sum_{\varphi \in \Uparrow(\tau)} c_\varphi$ are activated. $S \in \mathbb{I}$ and can be evolved within the tissue gene or be specified for a task.

Therefore, we define $I_\tau = \{L_\varphi | c_\varphi = \max_{\varphi' \in \Uparrow(\tau)} (c_{\varphi'})\}$ as the input set after coarse coding to sensory neuron Φ_τ. For sensor neurons that are active, we calculate s_τ:

$$s_\tau = \min_{i \in [1,\ldots,q]} (p_i) \ni p_j \cap I_\tau \neq \emptyset, \forall j < i \tag{6}$$

where p_j is an element of a global priority list P of sensory states, $P = [p_1 \ldots p_q]$ and where $p_j \in A'$. The global priority list is obtained by polling a group of filter units

and is described in Section 3.4.1. In summary, each sensory neuron takes inputs from $\Uparrow(\tau)$ and produces a single output s_τ, where both inputs and outputs are restricted to the states in A'. This reduction of inputs to a single output is done through prioritized filtering using the global priority list P. Thus if a sensory neuron's input set $\Uparrow(\tau)$ contains one or more states p_1, the sensory neuron's output s_τ is set to p_1, regardless of its other input states. Similarly, if a sensory neuron's input set contains one or more states p_2 and no states p_1, the output is set to p_2, regardless of its other inputs, and so on down the priority list.

3.4.1 Input Filtering

The priority list P is generated by polling a group of n filter units. Each of these independent units takes in as input q weighted inputs and produces a single binary output using the threshold activation function ψ_{up} from (1). Each filter unit j has q weights w_{jk}, $1 \leq k \leq q$. To poll the filter units for a particular input state $s_{\alpha_k} \in A'$, the units are given an input vector of size q containing all zeros, except for at position k, which is set to one, and their outputs are summed to yield $V_{s_{\alpha_k}}$. Thus to tally the votes for input state s_{α_3}, the filter units receive the input vector $[0\ 0\ 1\ 0 \ldots 0]$ of size q and their outputs are summed as given below:

$$V_{s_k} = \sum_{j}^{n} \psi_{\text{up}}\left(w_{jk}, \theta_j\right) \tag{7}$$

This process is repeated for all states in A', and the priority list is generated by assigning the state with the highest number of votes to p_1, assigning the state that garnered the second highest number of votes to p_2, etc. In case of a tie, the tie-breaker is the sum of the raw outputs of the filter networks, i.e., before the ψ_{up} activation function is applied.

3.4.2 Evolution and Development

Fig. 5 shows the additional types of genes included in the tissue genome. These genes are developed similar to the motor neurons and decision neurons as described in Section 3.3. The sensor neurons are grown on a two-dimensional spatial map. Mutations can perturb the contents of an existing gene or result in the development of new ones. Both the filter units and sensor neurons also have a "sensor type" specifier which restricts each genome to access certain types of sensory inputs such as obstacle detection or color detection (See Section 4 for further details). Furthermore, sensor neurons have the capability of referencing different groups of filter units using the "Filter Reference" parameter. However for the experiments presented here we set this value to 0.

Filter Units

Specifier	Reference Address	Sensor Type	Filter ID	Function Parameters	Gene Activate
D	A	ρ	x	w_1 w_2 ... w_v θ	G
Integer [0,4]	Integer	Integers [0,1]	Integer	Integer [0,100]	Binary

Sensor Neuron

Specifier	Reference Address	Position		Receptive Field		Filter Reference	Sensor Type	Gene Activate
D	A	x	y	a	b	F	ρ	G
Integer [0,4]	Integer	Real		Integers [1,3]		Integer	Integers [0,1]	Binary

Fig. 5 Gene map for the Sensory Coarse Coding Model

4 An Example Task: Resource Gathering

The effectiveness of the ANT controller is demonstrated in simulation on the resource gathering task [32]. A team of robots collects resource material distributed throughout its work space and deposits it in a designated dumping area. The workspace is modeled as a two-dimensional grid environment with one robot occupying four grid squares.

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 berm at the designated location. (In the present experiment, a berm is simply a mound of the resource material.) The berm location has perimeter markings on the floor and a light beacon mounted nearby. The two colors on the border are intended to allow the controller to determine whether the robot is inside or outside the berm location (Fig. 6).

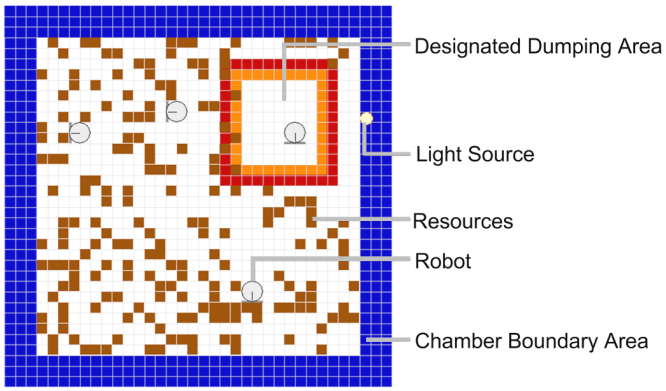


Fig. 6 2D grid world model of experiment chamber.

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 number of time steps, in this case $T = 300$. Darwinian selection is performed based on the fitness value of each controller averaged over 100 different initial conditions.

Table 1 Predefined Sensor Inputs

Sensor Variables	Function	Description
$V_1 \ldots V_4$	Resource Detection	Resource, No Resource
$C_1 \ldots C_4$	Template Detection	Blue, Red, Orange, Floor
S_1, S_2	Obstacle Detection	Obstacle, No Obstacle
LP_1	Light Position	Left, Right, Center, No Light
LD_1	Light Range	0-10 (distance to light)

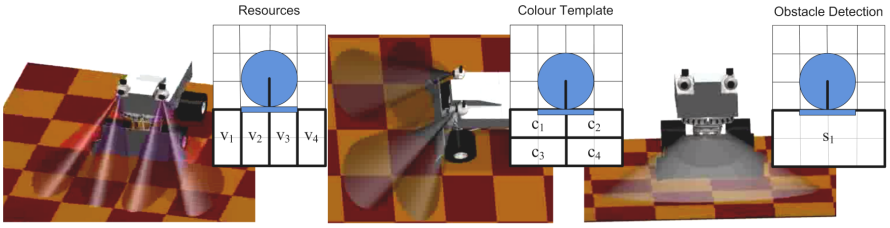


Fig. 7 Predefined input sensor mapping, with simulation model inset.

Table 2 Preordered Basis Behaviors

Order	Behavior	Description
1	Dump Resource	Move one grid square back; turn left
2	Move Forward	Move one grid square forward
3	Turn Right	Turn 90° right
4	Turn Left	Turn 90° left
5, 7, 9, 11	Bit Set	Set memory bit i to 1, $i = 1 \ldots 4$
6, 8, 10, 12	Bit Clear	Set memory bit i to 0, $i = 1 \ldots 4$

Simple feature-detection heuristics are used to determine the values of $V_1 \ldots V_4$ and $C_1 \ldots C_4$ based on the grid locations shown. For detection of the light beacon, the electronic shutter speed and gain are adjusted to ensure that the light source is visible while other background features are underexposed. The position of the light LP_1 is determined based on the pan angle of the camera. The distance to the light source LD_1 is estimated based on its size in the image. The robots also have access to four memory bits, which can be manipulated using some of the basis behaviors. Table 2 lists the basis behaviors the robot can perform. These behaviors are activated based on the output of the ANT controller, and all occur within a single time-step.

4.1 Coupled Motor Primitives

In this section we consider an alternative setup, where the ANT controllers are provided building blocks for the basis behaviors in the form of motor primitive [9] sequences. The motor primitives are taken as discrete voltage signals over a discrete time window applied on DC motors as shown in Fig. 8 and as arguments to the motor primitive commands in Table 3. These voltage output signals feed to the actuators and can be in one of three states, $\{1,0,-1\}V$ for a discrete time window, Δt_n, $n \in \{0,1,2,3,4,5\}$ as shown. In addition, each actuator takes on a default voltage value of 0. The actual value of V, the voltage constant, is dependent on the actuator.

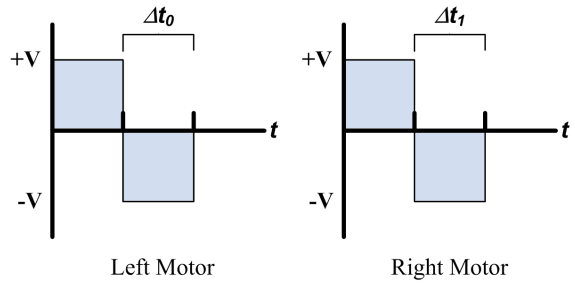


Fig. 8 Motor primitives composed of discretized voltage signals shown for a simulated robot.

Tissue Gene

Specifier	Neuron Replication Prob.	Neuron Replication Ratios		Seed Address	Output Signal Order						
D	T_r	n_d	n_m	T_s	o_1	o_2	o_3	...	$o_{\varepsilon-2}$	$o_{\varepsilon-1}$	o_ε
Integer [0,2]	Real [0.001,0.1]	Integers [1,10]		Integer	Integer [1, ε]						

Fig. 9 Modified tissue gene that includes order of execution of motor primitive sequences.

Table 3 Coupled Motor Primitives for the Sign-Following Task.

Neuron ID	Behavior	Coupled Motor Signals
1	Move Forward	Left Motor 1 || Right Motor 1
2	Turn Right 90°	Left Motor 1 || Right Motor -1
3	Turn Left 90°	Left Motor -1 || Right Motor 1
4	Pivot Right	Left Motor 0 || Right Motor -1
5	Pivot Left	Left Motor 0 || Right Motor 1
6	Pivot Right	Left Motor 1 || Right Motor 0
7	Pivot Left	Left Motor -1 || Right Motor 0
8, 10, 12, 14	Bit set	Set memory bit i to 1, $i = 1 \cdots 4$
9, 11, 13, 15	Bit clear	Set memory bit i to 0, $i = 1 \cdots 4$

The ANT controller also needs to determine the order of execution of these motor primitive sequences. The modified tissue gene is shown in Fig. 9. The order of the output coupled motor primitive (CMP) sequences are evolved as additional parameters in the tissue gene and is read starting from the left. The elements of the table, $o_1, \cdots, o_\varepsilon$ contain the Neuron ID values. The order is randomly initialized when starting the evolutionary process and with each Neuron ID occupying one spot on the gene. Point mutations to this section of the tissue gene involves swapping Neuron ID values between sites. Table 3 shows the repertoire of coupled motor primitives provided for the ANT controllers and thus $\varepsilon = 15$ for this particular setup (Fig. 9). The motor primitives are coupled, where for example the left drive motor and the right drive motor are executed in parallel (indicated using ||). Under this setup, it is still possible for the controller to execute a sequence of motor primitives in a serial fashion.

4.2 Evolutionary Parameters

The evolutionary algorithm population size for the experiments is $P = 100$, with crossover probability $p_c = 0.7$, mutation probability $p_m = 0.025$ and a tournament size of $0.06P$. The tissue is initialized as a "seed culture" , with 3×6 motor control neurons in one layer. After this, the tissue is grown to include 70–110 neurons (selected from a uniform random distribution) before starting the evolutionary process These seeding parameters are not task specific and have been observed to be sufficient for a number of different robotic tasks.

5 Results

We compare evolutionary performance of various evolvable control system models in Fig. 10. Included is a Cellular Automata lookup table that consists of a table

of reactive rules that spans 12^{16384} entries for this task which was evolved using population, selection and mutation parameters from Section 4.2. The genome is binary and is merely the contents of the lookup table. For this approach, we also assumed that the light beacon is turned off. Hence there exists $2^4 \times 4^4 \times 2^2 = 16384$ possible combinations of sensory inputs states, accounting for resource detection, template detection and obstacle detection sensors respectively (Table 1). For each combination of sensory input, the 12 allowable behaviors outlined in Table 2 could be executed. As can be seen, the population quickly stagnates at a very low fitness due to the 'bootstrap problem' [23]. With limited supervision, the fitness function makes it difficult to distinguish between incrementally fitter solutions. Instead the system depends on 'bigger leaps' in fitness space (through sequences of mutations) for it to be distinguishable during selection. However, bigger leaps become more improbable as evolution progresses, resulting in search stagnation. The performance of a population of randomly initialized fixed-topology, fully-connected networks, consisting of between 2 and 3 layers, with up to 40 hidden and output neurons is also shown in Fig. 10.

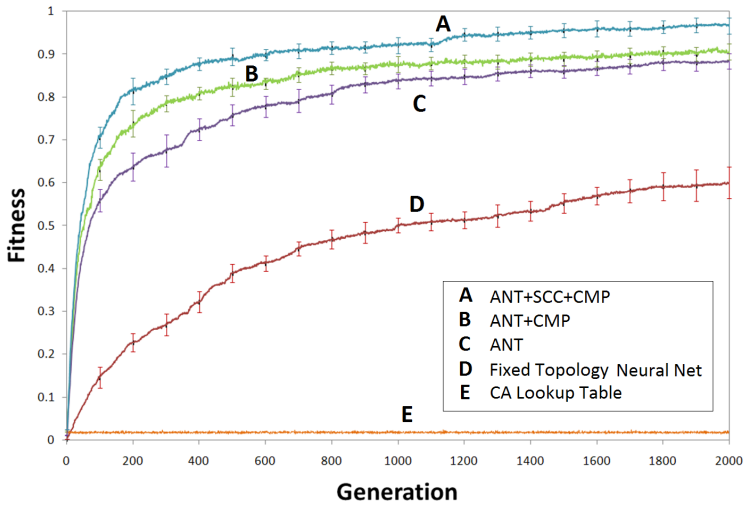


Fig. 10 Evolutionary performance comparison, showing population best averaged over 30 Evolutionary Algorithm runs of various evolvable control architectures. Error bars indicate standard deviation. As shown, ANT combined with Sensory Coarse Coding (SCC) and Coupled Motor Primitives (CMP) ordered through evolution obtains desired solutions with fewer genetic evaluations. The CA lookup table approach as shown remains stagnant and is unable to solve the task while fixed-topology neural nets converge at a much slower rate.

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. In this regard ANT is advantageous, since the topology is evolved and decision neurons learn to inhibit noisy neurons through

a masking process. The net result is that ANT requires fewer genetic evaluations to evolve desired solutions in comparison to standard neural networks. The standard ANT model using sensory inputs and basis behaviors outlined in Tables 1 and 2, respectively, shows a noticeable improvement in evolutionary performance over the lookup table and fixed topology architectures. Further improvement is gained when we allow the ANT architecture to evolve the execution order scheme and coupled motor primitives (Section 4.1) instead of using a list of preordered basis behaviors.

Finally, we also compare ANT+SCC using coupled motor primitives with these models. To make the comparison meaningful with respect to the other models, we impose some restrictions on the ANT+SCC configuration. This includes limiting the maximum number of active (selected) sensor neurons from the SCC model to 4 for the resource detection layer and 4 for the template detection layer. We also used predefined layouts for the other spatial sensors, namely obstacle detection. As can be seen in these results, ANT+SCC shows a noticeable performance advantage over the baseline ANT model. Furthermore, we obtain equivalent population best fitness values requiring approximately 5 times less genetic evaluations than with the baseline ANT model (Table 4). It should be noted that desired solutions ($f \geq 0.885$) were not obtained using standard neural networks within 10,000 generations of evolution. Examples of an evolved execution order scheme and sensor priority table using ANT+SCC+CMP are shown in Fig. 12.

Table 4 Number of generations required to obtain desired solutions ($f \geq 0.885$).

Method	Avg. Generations	Standard Deviation
ANT+SCC+CMP	421	133
ANT+CMP	1,142	241
ANT+SCC+CMP Control Experiment 1	1,343	104
ANT+SCC+CMP Control Experiment 2	1,968	312
ANT	1,983	235
Fixed Topology Neural Net.	$>$ 10,000	NA

A typical evolutionary run for the resource gathering task using ANT takes approximately six hours on a dual core Intel T7200, 2GHz desktop processor, with only one core being used for the evolutionary run. With ANT + SCC + CMP, one can get a comparably suitable solution in less than one hour thirty minutes. Furthermore since this five fold improvement in performance is due to enhancements in the search process, the improvement is expected to carry over with faster processors. Using regular neural networks comparable solutions were not obtained even after approximately 30 hours (10,000 generations) of evolution.

The solutions obtained in Table 4 can accumulate at least 88.5% of the dispersed resources in the designated dumping area within $T = 300$ timesteps and has been determined to be of sufficient quality to complete the task (see Fig. 18 for hardware demonstration). Given more timesteps, it is expected that the robots will have accumulated the remaining resources. One would ideally like to provide as input raw

sensor data to the robot controller. However this results in an exponential increase in search space for a linear increase in sensor states. The alternative would be to filter out and guess which subset of the sensory states maybe useful for solving a prespecified task. A wrong guess or poor design choice may make the process of finding a suitable controller difficult or impossible. Hence, ANT+SCC allows for additional flexibility, by helping to filter out and determine suitable sensory input states.

Fig. 13 shows the evolved population of sensory neurons on the body-centric spatial map. Fig. 11 (left) shows the average area used by selected sensor neurons and the average number of sensor neurons that participated in the selection process during evolution. The average area remains largely constant indicating there is strong selective pressure towards particular geometric shapes and area. This makes sense for the resource gathering task, as controllers need to detect a sufficiently large area in front to identify color cues indicating whether the robot is inside or outside the dumping area. What is interesting is that with $S = 4$, for template detection sensors neurons, we still see a steady increase in the number of sensor neurons competing to get selected. The increased number of neurons can potentially act in a cooperative manner, reinforcing and serving as redundant receptive fields covering key locations on the spatial map. Redundancy is beneficial in limiting the impact of deleterious mutations. Fig. 11 (right) shows that the individuals in the evolutionary process start off by sensing a smaller area and that this area is steadily increased as the solutions converge. If each sensor neuron senses just one grid square area, then filtering is effectively disabled. At the beginning of the evolutionary process, individuals take on reduced risk by sensing a smaller effective area, but as the filtering capability evolves concurrently (correctly prioritizing the sensory cues), it allows for the individual controllers to sense and filter a larger area. The number of active filter units continue to get pruned, until they reach a steady state number. This trend is consistent with experiments using ANT [31], where noisy neurons are shut off as the controllers converge towards a solution.

In order to measure the impact of the coarse-coding and filtering towards ANT+SCC performance improvement, we performed control experiment 1, where the maximum size of the sensor cells was restricted to one grid square and where the net concentration of each grid square within the spatial map was set to 1 (Table 4). These two modifications effectively prevent coarse sensor cells from forming and interacting to form fine representations. Instead, what is a left is a group of fine sensor neurons that are always active. With the sensor cell area being restricted to one grid square, the priority filter has no effect, since it requires at least two grid squares with differing sensory input states. The fitness performance of this model is comparable to the baseline ANT model. However, since this model also uses coupled motor primitives and it performed worse than ANT+CMP alone, the net impact of these imposed constraints is actually a decrease in performance. Furthermore, we performed a second control experiment (control experiment 2), where we imposed the receptive field sizes to 3×3 grid squares and set the net concentration at each grid square to 1 (Table 4). These two modifications ensure the receptive field remains coarse and prevents coarse coding interactions from occurring, while leaving

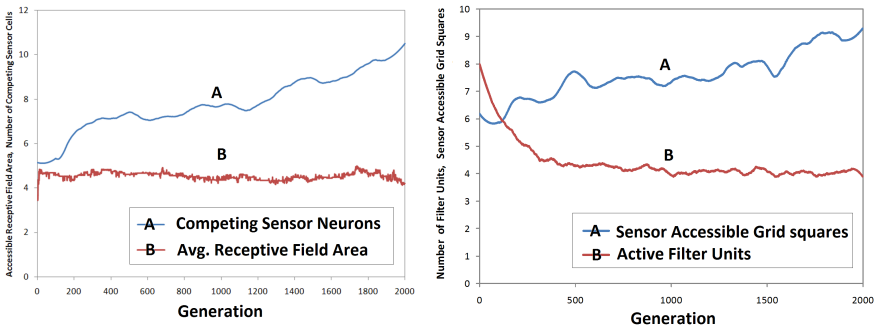


Fig. 11 (Left) Average area occupied by selected sensor neurons and number of sensor neurons that participated in the selection process during evolution (Right) Number of active filter units and number of grid squares accessible by the sensor neurons during evolution. Both plots show parameters from population best averaged over 30 Evolutionary Algorithm runs.

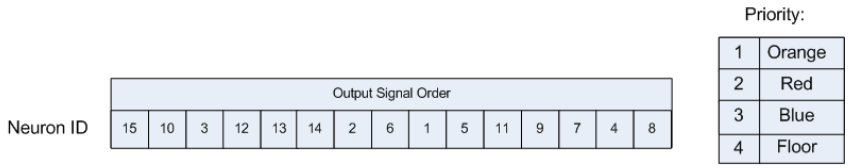


Fig. 12 Evolved coupled motor primitives ordering scheme and sensor priority list for template detection shown for an ANT+SCC+CMP controller with a fitness of 0.98. See Table 3 and 1 for reference.

the filter functionality within SCC turned on. The net effect is that we a see noticeable drop in performance due to SCC. Both of these experiments indicate that coarse-coding interaction between sensor neurons is helping to find desired solution within fewer genetic evaluation.

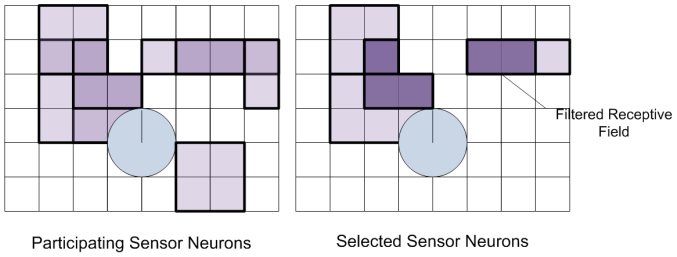


Fig. 13 Example of an evolved sensor layout (fitness of 0.98) using the ANT+SCC+CMP model. (Left) Participating sensor neurons and receptive fields (template detection) shown. (Right) Selected sensor neurons shown. Shaded area indicates resultant regions sensed by the controller.

5.1 Evolution and Robot Density

Fig. 14 shows the fitness (population best) of the overall system evaluated at each generation of the artificial evolutionary process using the baseline ANT model, with a specified initial resource density and various robot densities. These results show that system performance increases with the number of robots present (with total area held constant). For scenarios initialized with more robots, each robot has a smaller area to cover in trying to gather and dump resources.

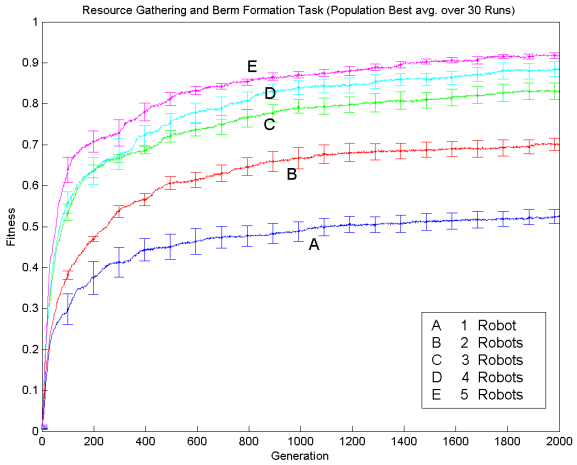


Fig. 14 Evolutionary performance comparison of ANT-based solutions for one to five robots. Error bars indicate standard deviation.

5.2 Behavioral Adaptations

In an ANT-based architecture, networks are dynamically formed with decision neurons processing the sensory input and in turn "selecting" motor-control neurons through coarse-coding [31]. The behavioral activity of the controllers (see Fig. 16) shows the formation of small networks of neurons which handle individual behaviors, 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 user-defined, distal behaviors. Instead, the activity of the decision neurons indicate distribution of specialized "feature detectors" among independent networks.

Some of the emergent solutions evolved indicate that the individual robots all figure out how to dump nearby resources into the designated berm area, but that not

all robots deliver resource all the way to the dumping area every time. Instead, the robots learn to pass the resource material from one individual to another during an encounter, forming a "bucket brigade" (see Fig. 15, 18). This technique improves the overall efficiency of the system as less time is spent traveling to and from the dumping area. Since the robots cannot explicitly communicate with one another, these encounters happen by chance rather than through preplanning. As with other multiagent systems, communication between robots occurs through the manipulation of the environment in the form of stigmergy. The task in [33] is similar in that distributed objects must be delivered to a confined area; however, the hand-designed controller does not scale as well as the "bucket brigade" solution that the ANT controllers discovered here. We also noticed that the robot controllers do make use of the light beacon to home in on the light beacon that is located next to a dumping area, however there is no noticeable difference in fitness performance, when the robot controllers are evolved with light turned off [32]. In these simulation experiments, the robots have no way to measure the remaining time available; hence, the system cannot greedily accumulate resource materials without periodically dumping the material at the designated area.

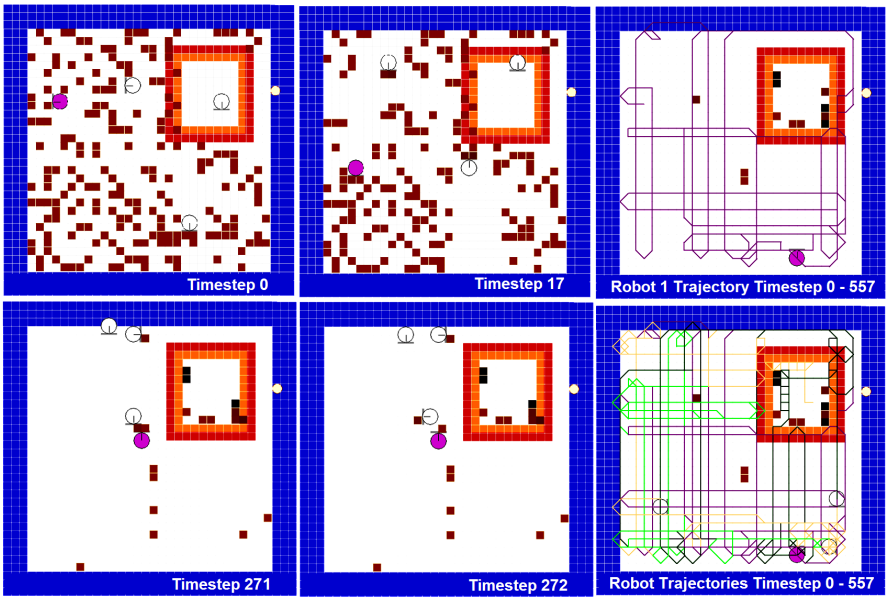


Fig. 15 Snapshots of robots and trajectories of a task simulation (4 robots).

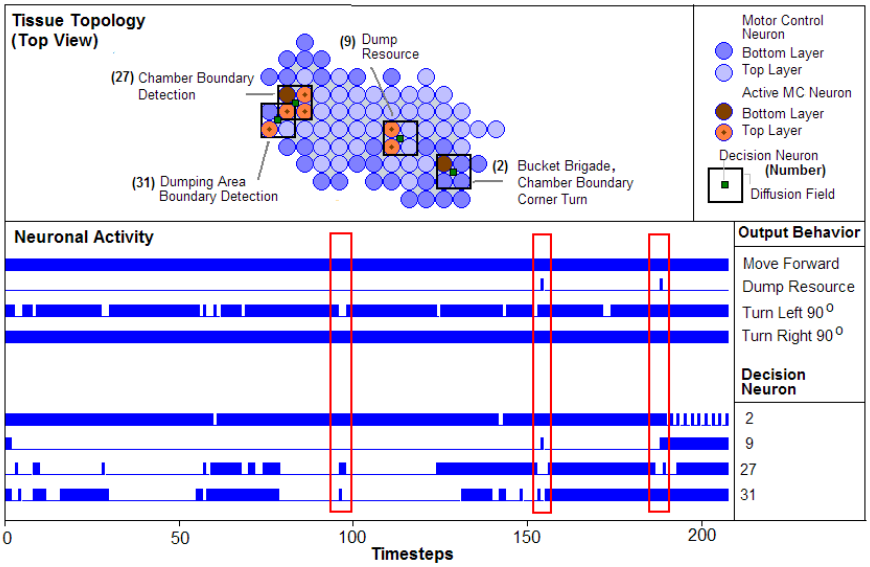


Fig. 16 Tissue Topology and neuronal activity of a select number of decision neurons. Decision neurons in turn "select" (excite into operation) motor control neurons within its diffusion field.

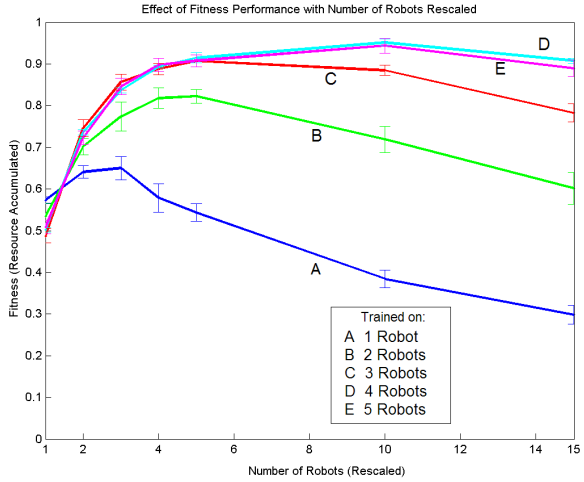


Fig. 17 Scaling of ANT-based solutions from one to five robots.

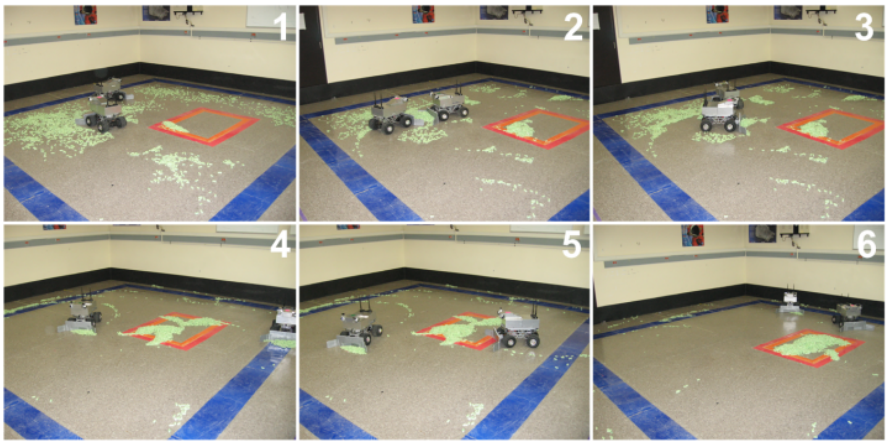


Fig. 18 Snapshots of two rovers performing the resource gathering task using an ANT controller. Frames 2 and 3 show the "bucket brigade" behavior, while frames 4 and 5 show the boundary avoidance behavior.

5.3 Evolved Controller Scalability

We examine the fittest solutions from the simulation runs shown in Fig. 17 for scalability in the number of robots while holding the amount of resources constant. 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 behavior necessary to function well within a multiagent setting. For example, such controllers fail to develop "robot collision avoidance" or "bucket brigade" behaviors. Similarly, the robot controllers evolved with two or more robots perform demonstrably worse when scaled down to a single robot, showing that the solutions are dependent on cooperation among the robots.

6 Discussion

In this chapter, we use a global fitness function to train multirobot controllers with limited supervision to perform self-organized task decomposition. Techniques that perform well for the task make use of modularity and generalization. Modularity involves use/reuse of components, while generalization involves finding patterns or making inferences from many particulars. With a multirobot setup, modularity together with parallelism is exploited by evolved controllers to accomplish the task. Rather than have one centralized individual attempting to solve a task using global

information, the individuals within the group are decentralized, make use of local information and take on different roles through a process of self-organization. This process of having different agents solve different subcomponents of the task in order to complete the overall task is a form of task decomposition.

In this multirobot setup, there are both advantages and disadvantages to consider. Multiple robots working independently exploit parallelism, helping to reduce the time and effort required to complete a task. Furthermore, we also see that solutions show improved overall system performance when evolved with groups of robots. It should be noted that the density of robots is critical towards solving the task. Higher densities of robots results in *antagonism*, with robots spending more time getting out of the way of one another rather than progressing on the task, leading to reduced system performance.

It was shown that a CA look-up table architecture that lacked both modularity and generalization is found to be intractable due to the 'bootstrap problem,' resulting in premature search stagnation. This is due to the fact that EAs are unable to find an incrementally better solution during the early phase of evolution. Use of neural networks is a form of functional modularization, where each neuron performs sensory-information processing and makes solving the task more tractable. However 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 [16]. Crosstalk in combination with limited supervision (through use of a global fitness function) can again lead to the 'bootstrap problem' [23]. Thus, choosing the wrong network topology may lead to a situation that is either unable to solve the problem or difficult to train [31].

With the use of Artificial Neural Tissues (ANT), we introduce hierarchical functional modularity into the picture. The tissue consists of modular neurons that can form dynamic, modular networks of neurons. These groups of neurons handle specialized functionality as we have shown and can be reused repeatedly for this purpose. In contrast, with a standard fixed topology neural network, similar functionality may need to evolve independently multiple times in different parts of the network. In these various neural network architectures, modularity is functional, with behaviors and capabilities existing in individual neurons or in groups and triggered when necessary. ANT facilitates evolution of this capability by allowing for regulatory functionality that enables dynamic activation and inhibitions of neurons within the tissue. Groups of neurons could be easily activated or shut-off through a coarse-coding process. Furthermore, with the ANT+SCC model, we allow for evolution of both spatial and functional modularity. Spatial modularity is possible with the SCC model, since we may get specialized sensory neurons that find spatial sensory patterns. The output from these sensory neurons are used as input by various groups of neurons active within ANT. These sensor neurons act as specialized feature detectors looking for either color cues or resources.

Comparison of the various evolvable control system models indicates that controllers with an increased ability to generalize, evolve desired solution with far fewer genetic evaluations. The CA lookup table architecture lacks generalization capability and performed the worst. For the CA lookup table, evolved functionality needs to

be tuned for every unique combination of sensory inputs. A regular fixed topology network performed better, but since the topology had no capacity to increase in size or selectively activate/inhibit neurons within the network, it needed to tune most of the neurons in the network towards both helping perform input identifications or actions and preventing these same neurons from generating spurious outputs. Thus the same capabilities may have to be acquired by different neurons located in different parts of the network requiring an increased number of genetic evaluations to reach a desired solution.

The standard ANT architecture can quickly shut off (mask out) neurons generating spurious output and thus does not require having sequences of mutation occur, tuning each neuron within the tissue to acquire compatible (or similar) capabilities or remain dormant. Thus certain networks of neurons within the tissue can acquire and apply a certain specialized capability (Fig. 16), while most others remain dormant through the regulatory process. Hence within ANT, increased functional generalization is achieved through specialization. With the fixed topology neural network, the net effect of all the neurons having to be active all the time implies that the controllers have to evolve to individually silence each of the spurious neurons or acquire the same capabilities repeatedly, thus implying reduced functional generalization.

ANT+SCC can generalize even further. Apart from being able to selectively activate/inhibit neurons, it can also choose to receive a coarse or fine representation of the sensory input. In other words, it can perform further sensor generalization. A coarse representation of the sensory input in effect implies some degree of generalization. The priority filtering functionality prioritizes certain sensor states over others, while the coarse coding representation selects a subset of the inputs to send to the filter. The resultant input preprocessing facilitates finding and exploiting underlying patterns in the input set. The net effect is that the controller does not have to deal with as many unique conditions since the number of unique sensory input combinations seen by the ANT controller is reduced by SCC. This in turn facilitates evolution of controllers that require fewer generations to reach a desired solution. At the same time, over-generalization of the sensory inputs is problematic (see ANT+SCC+CMP control experiment 2). By imposing coarse receptive fields and preventing coarse-coded interactions the controllers may miss key (fine) features through prioritized filtering. Hence, although the sensory input space may effectively have shrunk, through over-generalization valuable information is lost. These results justify the need for representations that selectively increase or decrease generalization of sensory input through coarse-coding.

This increased ability to generalize by the ANT+SCC model also seems to offset the increase number of parameters (increased search space) that needs to be evolved. Herein lies a tradeoff, as a larger search space alone may require a greater number of genetic evaluations to reach a desired solution, but this may also provide some unexpected benefits. In particular, a larger space may help in finding more feasible or desirable solutions than those already present and may even reduce the necessary number of genetic evaluations by guiding evolution (as in the ANT+SCC case). As pointed out, ANT+SCC with its ability to further generalize sensory input appears to

provide a net benefit, even though it needs to be evolved with additional parameters (in comparison to the standard ANT model).

This benefit is also apparent when comparing the baseline ANT controller with ANT-ordered coupled motor primitives. The additional genomic parameters appears to be beneficial once again, since the search process has access to more potential solutions. Furthermore, it should be noted that these additional degrees of freedom within the ANT+SCC controller do not appear to introduce deceptive sensory inputs or capabilities. Deceptive inputs and capabilities can slow down the evolutionary process, since the evolving system may retain these capabilities when they initially provide a slight fitness advantage. However, these functionalities can in turn limit or prevent the controllers from reaching the desired solution. Thus in effect, the evolving population can get stuck in a local minimum, unable to transcend towards a better fitness peak.

7 Conclusion

This chapter has reported on a number of experiments used to automatically generate neural network based controllers for multirobot systems. We have shown that with judicious selection of a fitness function, it is possible to encourage self-organized task decomposition using evolutionary algorithms. We have also shown that by exploiting hierarchical modularity, regulatory functionality and the ability to generalize, controllers can overcome tractability concerns. Controllers with increased modularity and generalization abilities are found to evolve desired solutions with fewer training evaluations by effectively reducing the size of the search space. These techniques are also able to find novel multirobot coordination and control strategies. To facilitate this process of evolution, coarse-coding techniques are used to evolve ensembles of arbitration neurons that acquire specialized functionality. Similar techniques are used to evolve sensor-filter configurations. Both techniques facilitate functional and spatial modularity and generalization. This combination allows for a methodical approach to control development, particularly one where the controller and robot sensory configurations can be automatically generated starting from a blank slate, where the designer can be largely relieved of the design process and where detailed models of the system or environment can be avoided.

References

1. R. Beckers, O. E. Holland, and J. L. Deneubourg, (1994) "From local actions to global tasks: Stigmergy and collective robots," *Fourth International Workshop on the Syntheses and Simulation of Living Systems*. MIT Press, pp. 181–189.
2. E. Bonabeau, G. Theraulaz, J.-L. Deneubourg, S. Aron, and S. Camazine, "Self-organization in social insects," *Trends in Ecology and Evolution*, vol. 12,

May 1997, pp. 188–193.
3. E. Bonabeau, M. Dorigo, and G. Theraulaz, *Swarm Intelligence: From Natural to Artificial Systems*. New York: Oxford Univ. Press, 1999.
4. 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 2001*, San Francisco, CA, pp. 829-836
5. F. Chantemargue, T. Dagaeff, M. Schumacher, and B. Hirsbrunner, "Implicit cooperation and antagonism in multi-agent systems," University of Fribourg, Technical Report, 1996.
6. K. Chellapilla and D. B. Fogel. "Evolving an expert checkers playing program without using human expertise," *IEEE Transactions on Evolutionary Computation*, 5(4):422-428, 2001.
7. R. Das, J.P. Crutchfield, M. Mitchell,J. Hanson, (1995) "Evolving globally synchronized cellular automata," *Proceedings of the Sixth International Conference on Genetic Algorithms 1995*, pp. 336-343, San Fransisco, CA, Morgan Kaufmann
8. F. Dellaert, R. Beer, (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.
9. J. Demeris, M.J. Matarić, (1998) "Perceptuo-Motor Primitives in Imitation," *Autonomous Agents '98 Workshop on Agents in Interaction Acquiring Competence*.
10. D. Federici, K. Downing, "Evolution and Development of a Multicellular Organism: Scalability, Resilience, and Neutral Complexification," *Artificial Life*, (2006) vol. 12, pp. 381–409.
11. Gauci, J., Stanley K., (2008) "A Case Study on the Critical Role of Geometric Regularity in Machine Learning," *Proceedings of the 23rd AAAI Conference on AI*, AAAI Press
12. P. Grassé, "La reconstruction du nid les coordinations interindividuelles; la theorie de stigmergie," *Insectes Sociaux*, vol. 35, 1959, pp. 41–84.
13. R. Groß, M. Dorigo, (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
14. F. Gruau, D. Whitley, L. Pyeatt, (1996) "A comparison between cellular encoding and direct encoding for genetic neural networks," *Genetic Programming 1996*, 81-89. Cambridge, MA: MIT Press.
15. T. Hastie, R. Tibshirani, and R. Friedman, *The Elements of Statistical Learning*. New York: Springer, 2001.
16. R. Jacobs, M. Jordan, A. Barto, "Task decomposition through competition in a modular connectionist architecture," *Cognitive Science*, (1991) (15):219–250.
17. M. Komosinski and S. Ulatowski, (1998) "Framsticks: towards a simulation of a nature-like world, creatures and evolution," *Proceedings of the 5th European Conference on Artificial Life*, Berlin, Springer-Verlag.

18. B. R. Leffler, M. L. Littman, and T. Edmunds. "Efficient reinforcement learning with relocatable action models," *AAAI Journal*, pages 572577. AAAI Press, 2007.
19. A. Lindenmayer, "Mathematical models for cellular interaction in development I. Filaments with one-sided inputs," *Journal of Theoretical Biology*, (1968) 18:280–289
20. H. Lipson and J. Pollack, "Automatic design and manufacture of artificial lifeforms," *Nature*, 406:974-978, 2000.
21. M. J. Matarić, M. Nilsson, and K. T. Simsarian, (1995) "Cooperative multi-robot box-pushing," *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 556–561.
22. C. Mautner and R.K. Belew, (1999) "Evolving Robot Morphology and Control," In M. Sugisaka, editor, *Proceedings of Artificial Life and Robotics 1999* (AROB99), Oita, ISAROB.
23. S. Nolfi, D. Floreano, *Evolutionary Robotics : The Biology, Intelligence, and Technology of Self-Organizing Machines*, MIT Press, 2000.
24. C. A. Parker, H. Zhang, , and C. R. Kube, (2003) "Blind bulldozing: Multiple robot nest construction," *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 2010–2015.
25. R. Pfeifer and C. Scheier, *Understanding Intelligence*. MIT Press, Cambridge, MA, 1999.
26. Roggen, D., Federici, D., (2004) "Multi-cellular Development: Is There Scalability and Robustnes to Gain?" *Proceedings of Parallel Problem Solving from Nature*, pp. 391–400
27. K. Sims, (1994) "Evolving 3D Morphology and Behavior by Competition," *Proceedings of Artificial Life IV*, MIT Press, pp. 28–39.
28. K. Stanley, R. Miikkulainen, (2002) "Continual Coevolution through Complexification," *Proceedings of the Genetic and Evolutionary Computation Conference 2002*, San Francisco, California, Morgan Kaufmann.
29. J. Thangavelautham, T. Barfoot, and G.M.T. D'Eleuterio, (2003) "Coevolving communication and cooperation for lattice formation tasks (updated)," *Advances In Artificial Life: Proceedings of the 7th European Conference on ALife*, pp. 857–864.
30. J. Thangavelautham and G.M.T. D'Eleuterio, (2004) "A neuroevolutionary approach to emergent task decomposition," *8th Parallel Problem Solving from Nature Conference*, vol. 1, pp. 991–1000.
31. J. Thangavelautham and G.M.T. D'Eleuterio, (2005) "A coarse-coding framework for a gene-regulatory-based artificial neural tissue," *Advances In Artificial Life: Proceedings of the 8th European Conference on ALife*, pp. 67–77.
32. J. Thangavelautham, S. Alexander, D. Boucher, J. Richard, G.M.T. D'Eleuterio, (2007) "Evolving a Scalable Multirobot Controller Using an Artificial Neural Tissue Paradigm," *IEEE International Conference on Robotics and Automation*, Washington D.C.

33. J. Wawerla, G. Sukhatme, and M. Mataric, (2002) "Collective construction with multiple robots," *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 2696–2701.
34. M. Wilson, C. Melhuish, A. B. Sendova-Franks, and S. Scholes, "Algorithms for building annular structures with minimalist robots inspired by brood sorting in ant colonies," *Autonomous Robots*, vol. 17, 2004, pp. 115–136.
35. V. Zykov, E. Mytilinaios, B. Adams, H. Lipson (2005) "Self-reproducing machines," *Nature* Vol. 435 No. 7038, pp. 163-164