Presented at the International Conference on Evolvable Systems
Edinburgh, Scotland, 17–19 April 2000

From the Sea to the Sidewalk: The Evolution of Hexapod Walking Gaits by a Genetic Algorithm

Ernest J P Earon, Tim D Barfoot, and Gabriele M T D'Eleuterio

Institute for Aerospace Studies
University of Toronto
4925 Dufferin Street
Toronto, Ontario, Canada
M3H 5T6
{eea,tdb}@sdr.utias.utoronto.ca, gde@utias.utoronto.ca

Abstract. A simple evolutionary approach to developing walking gaits for a legged robot is presented. Each leg of the robot is given its own controller in the form of a cellular automaton which serves to arbitrate between a number of fixed basis behaviours. Local communication exists between neighbouring legs. Genetic algorithms search for cellular automata whose arbitration results in successful walking gaits. An example simulation of the technique is presented as well as results of application to *Kafka*, a hexapod robot.

1 Introduction

Insects and spiders are examples of relatively simple creatures from nature which are able to successfully operate many legs at once in order to navigate a diversity of terrains. Inspired by these biological marvels, robotics researchers have attempted to mimic insect-like behaviour in legged robots [2][3]. Typically, however, the control algorithms for these types of robots are quite complicated (e.g., dynamic neural networks), requiring fairly heavy on-line computations to be performed in real time. Here a simpler approach is presented which reduces the need for such computations by using a coarsely coded control scheme. This simplicity in the control code (and corresponding low computational requirements on the controller itself) allows for algorithms of this type to be easily implemented using embedded microcontrollers and thus making applications of similar mobile robots more easily realisable. This becomes more appealing if one is interested in making extremely small walkers.

Work has been done to increase the feasibility of using simulation to develop evolutionary control algorithms for mobile robots [11][12][18]. It has been suggested that hardware simulation is often too time consuming, and software simulation can accurately develop control strategies instead. It has been argued that the primary drawback to software simulation is the difficulty in modeling hardware interactions in software. While Jacobi has developed "minimal simulations" [12] in order to develop such controls, it has been found that while adequate behaviours can be evolved, there are still hardware concerns that are difficult to predict using minimal simulations (e.g. damaging current spikes in the leg drive motors) [10].

This work follows a similar vein to that of [8][9][10]. In [10], Gomi and Ide use a genetic algorithm to develop walking controllers for a mobile robot, an octopod robot. However, rather than reducing the search space through preselecting a reduced set of behaviours, they very precisely ***shaped*** [6] the fitness function for reinforcement. One benefit of this is that such fitness function optimization is much more easily scaled up to allow for more complicated behaviour combinations.

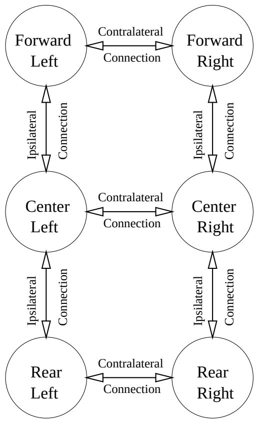
What is important to note, though, is that while Gomi and Ide [10] used measurements of the electric current supplied to the leg drive motors as one feature of their fitness function, as opposed to using only the observable success of the controller (i.e., the distance walked as measured with an odometer) as done in the present work, the results have similar attributes. For example, behaviours that could cause a high current flow in the leg motors such as having two or more legs in direct opposition, are also behaviours that would exhibit low fitness values when measuring the forward distance travelled. Thus when the controllers are evolved on hardware such parameters are often taken into account by the robot itself, and modeling is not necessarily required.

As with dynamic neural network approaches, each leg of a robot is given its own controller rather than using a central pattern generator. Communication between controllers is local (legs "talk" to their neighbours) resulting in a global behaviour (walking gait) for the robot. There is some neurobiological evidence that this notion of local control holds for some biological insects [4]. The controllers used here are cellular automata [16] or simple lookup tables. Each leg is given its own cellular automaton (CA) which can be in one of a finite number of states. Based on its state and those of its neighbours, a new state is chosen according to the lookup table. Each state corresponds to a fixed ***basis behaviour*** but can roughly be thought of as a specific leg position. Thus, new leg positions are determined by the old leg positions. Typically, all legs are updated synchronously such that the robot's behaviour is represented by a set of locally coupled difference equations which are much quicker to compute in real time than differential equations.

Under this framework, design of a control algorithm is reduced to coming up with the cellular automata which produce successful walking gaits. Our approach has been to use genetic algorithms to this end [15].

2 Cellular Automata

According to neurobiological evidence described by Cruse [4], the behaviour of legs in stick insects is locally coupled as in Figure 1. 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).



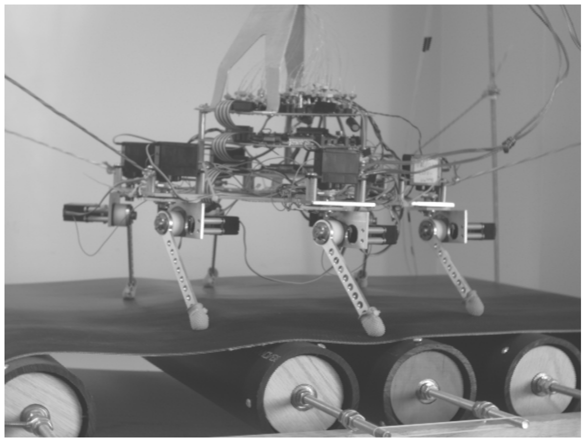


Fig. 1. (left) Behavioural coupling between legs in stick insects [4]. (right) *Kafka*, a hexapod robot, and treadmill setup designed to evolve walking gaits

We assume that the output of each leg controller may be discrete. This may be done by way of a set of ***basis behaviours*** [1][13]. 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. This may be considered a postprocessing step which takes a discretized output and converts it to the actuator control. This postprocessing step will not be allowed to change once set. The actual construction of the postprocessing requires careful consideration but is also somewhat arbitrary. Here the basis behaviours will be modules which move the leg from its current zone (in output space) to one of a finite number of other zones. Figure 2 shows two possible discretizations of a 2-degree-of-freedom output space (corresponding to a simple leg) into 4 or 3 zones. It is important to distinguish between a basis behaviour and a leg's current zone. If legs always arrive where they are asked to go, there is little difference. However, in a dynamic robot moving on rough terrain, the requested leg action may not always be successful. The key difference is that the ***output state*** of a leg will correspond to the current basis behaviour being activated, rather than the leg's current zone in output space. The ***input state*** of a leg could be either the basis behaviour or leg zone, depending on whether one wanted to try and account for unsuccessful requests. By using basis behaviours, the leg controllers may be entirely discrete. Once all the postprocessing has been set up, the challenge remains to find an appropriate arbitration scheme which takes in a discrete input state, x, (current leg positions or basis behaviours of self and neighbours) and outputs the appropriate discrete output, a, (one of M basis behaviours) for each leg. There are several candidates for this role but the one affording the most general decision surfaces between input and output is a straightforward lookup table similar to *cellular automata* (CA) In fact, this work follows on that of the Evolving Cellular Automata (EvCA) group at the Santa Fe Institute, New Mexico. In various papers

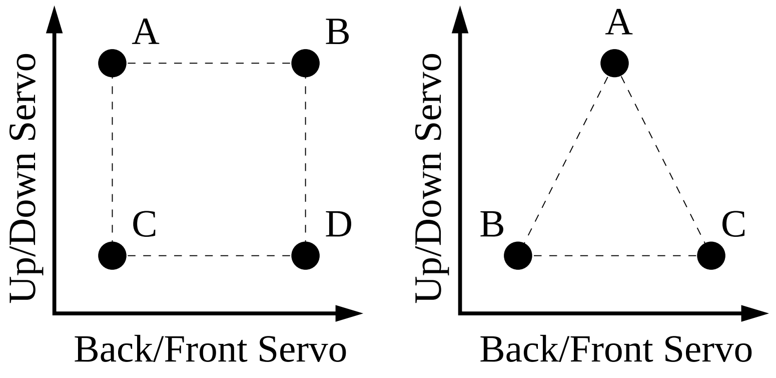


Fig. 2. Example discretizations of output space for 2 degree of freedom legs into (left) 4 zones and (right) 3 zones

(e.g., [5]) this group showed that genetic algorithms were able to evolve cellular automata which performed prescribed tasks requiring global coordination. This is essentially what we wish to achieve but we have the added difficulty of dealing with the physical environment of our robots. This type of lookup table control in autonomous robots is often called *reactive*. For every possible input sequence the CA scheme stores a discrete output value. In other words, for every possible input sequence there is an output corresponding to one of the basis behaviours. At each time-step, the leg controller looks up the action which corresponds to its current input sequence and carries it out. The size of the lookup table for a leg which communicates with $K-1$ other legs will then be M^K. The approach is therefore usually rendered feasible for only modest numbers for K and M. The number of all possible lookup tables is $M^{(M^K)}$. Again, modest numbers of basis behaviours keep the size of the search space reasonable. For example, with a hexapod robot with coupling as in Figure 1 and output discretization as in Figure 2 (left) the forward and rear legs will require lookup tables of size 4^3 and the central legs 4^4. If we assume left-right pairs of legs have identical controllers the combined size of the lookup tables for forward, center, and rear legs will be $4^3+4^3+4^4=384$ and the number of possible table combinations for the entire robot will be 4^{384}. From this point on, the term *CA lookup table* will refer to the combined set of tables for all legs in the robot (concatenation of individual leg lookup tables).

3 Genetic Algorithms

The crucial step in this approach is the discovery of particular CA lookup tables which cause the connected legs to produce successful walking gaits. We must discover the local rules which produce the desired global behaviour, if they indeed exist, and then find out why those rules stand out. The obvious first method to attempt is to design the local rules by hand. John Horton Conway picked his

Game of Life rules seemingly out of a hat with much success. How hard can it be? Well, it turns out to be very difficult to do this for all but the most trivial examples. We are typically faced with a combinatorial explosion. In the spirit of the EvCA group, an evolutionary global optimization technique will be employed, namely a genetic algorithm (GA), to search for good cellular automata.

GAs are based on biological evolution. For a good review see [7]. A random initial population of P CA lookup tables is evolved over G generations. Each CA lookup table, ϕ, has a *chromosome* which consists of a sequence of all the discrete values taken from the table. At each generation, a fitness is assigned to each CA lookup table (based on how well the robot performs on some walking task when equipped with that CA lookup table). A CA lookup table's fitness determines its representation in the next generation. Genetic crossovers and mutations introduce new CA lookup tables into the population. The best $K \leq P$ CA lookup tables are copied exactly from one generation to the next. The remaining $(P - K)$ CA lookup tables are made up by single site crossovers where both parents are taken from the best K individuals. Furthermore, they are subjected to random site mutations with probability, p_m, per site. The variable used to determine mutation is selected from a uniform random distribution.

4 Example Controller

In this section, the results of constructing an example controller in simulation will be presented. The purpose of the simulation is not to model an insect robot, but rather to demonstrate that a network of cellular automata controllers could produce patterns which resemble known walking gaits. Furthermore, we would like to show that a genetic algorithm is indeed capable of finding particular CA lookup tables which perform well on a user defined task.

One well established gait for hexapod robots is the "tripod" gait. The legs are divided into two sets

$$\{\text{Forward Left}, \text{Center Right}, \text{Rear Left}\}$$
$$\{\text{Forward Right}, \text{Center Left}, \text{Rear Right}\}$$

While one set is on the ground, the other is lifted, swung forward, and then lowered.

For the discretization of figure 2 (left), there are $I = 4^6 = 4096$ possible initial states for a hexapod robot. The fitness of a particular CA lookup table will be defined as

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

$$f_i = \begin{cases} 1 & \text{if tripod gait emerges within } T \text{ time-steps} \\ & \text{starting from initial condition, } i \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

This fitness function was used to evolve a particular CA lookup table named ϕ_{tripod}, which has a fitness of 0.984. That is, from 98.4% of all possible initial conditions, a tripod gait is reached. In our experiments, we have used a GA population size of $P = 50$, number of generations $G = 150$, keepsize $K = 15$, and mutation probability $p_m = 0.005$. The number of initial conditions per fitness evaluation was $I = 4096$ (all possible initial conditions) and the number of time-steps before testing for a tripod pattern was $T = 50$. Figure 3 depicts

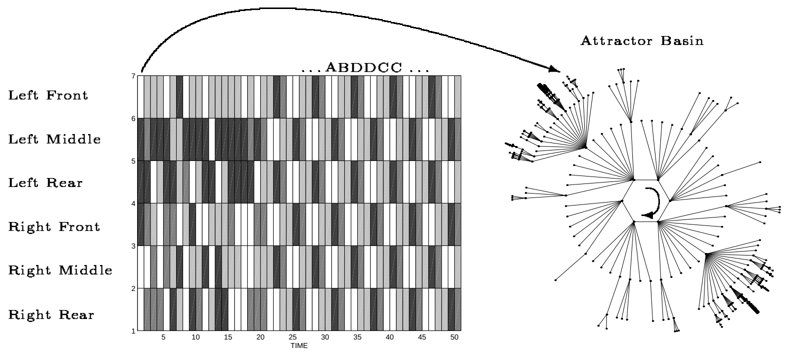


Fig. 3. (left) Gait diagram (time history) of ϕ_{tripod} on a particular initial condition. Each 6-state column represents an entire leg configuration of the robot; the left-most column is the initial condition. (right) Attractor basin portrait of ϕ_{tripod}. Each node represents an entire 6-state leg configuration of the robot. Lines represent transitions from one leg configuration to another. The inner hexagon represents the $ABDDCC\ldots$ tripod gait.

two aspects of ϕ_{tripod}. The left side shows a typical *gait diagram* or *time history* of ϕ_{tripod} on a particular initial condition. Each column shows the states of the 6 legs at a given time-step (different shades of grey represent different leg states). The left-most column is the initial condition. The right side is an attractor basin portrait of ϕ_{tripod} as described by [17]. Each node in this plot represents an entire 6-state leg configuration of the robot (i.e., one column of the left plot). The inner hexagon represents the tripod gait which is unidirectional (indicated by a clockwise arrow). The purpose of the right plot is to draw a picture of ϕ_{tripod} as a whole and to make a connection with the concept of stability. Beginning from any node on the attractor basin portrait, and following the transitions inward (one per time-step), one will always wind up on the inner hexagon (tripod gait).

The conclusion we may draw from this simple exercise is that this type of controller certainly is able to produce patterns resembling known walking gaits and that genetic algorithms are capable of finding them.

5 Hardware and Results

This section briefly describes *Kafka* [14], a 12-degree-of-freedom, hexapod robot. Figure 1 shows Kafka in action on a treadmill. Kafka's legs are powered by twelve JR Servo NES 4721 hobby servomotors controlled by a 386 66MHz PC. The control signals to the servos are absolute positions to which servos then move as fast as possible. In order to control the speed of the leg motions, then, the path of the leg is broken down into many segments and the legs are sent through each of those. The fewer the intervening points, the faster the legs moved. The robot has been mounted on an unmotorized treadmill in order to automatically measure controller performance (for walking in a straight line only). As Kafka walks, the belt on the treadmill causes the rollers to rotate. An odometer reading from the rear roller is fed to Kafka's computer such that distance versus time-step plots may be used to determine the performance of the controller. The odometer measures net distance travelled. For this experiment, the states for Kafka's legs were constrained to move only in a clockwise, forward motion which is based on the three state model as shown in figure 2 (right) (i.e., rather than having each leg move from one state to an arbitrary next state, it is constrained to either increment through the sequence of states in the clockwise motion, or to not change states). Each entry in the lookup table determines whether to increment the position of the leg or keep it stationary (binary output). This reduces the number of possible lookup tables from 3^{135} (2.58×10^{64}) for the full three-state model to 2^{135} (or 4.36×10^{40}). The GA parameters were

$$P = 50 \qquad K = 15 \qquad p_m = 0.05 \tag{3}$$

Each individual in the population was given the same initial condition, or stance, at the start of the evaluation ($\{C, B, C, B, C, B\}$).

Due to the necessary time duration required to evolve the controllers on Kafka, the number of generations that the GA was run was 23 (the evolution of the controllers on Kafka took approximately eight hours to complete). The convergence history for one evolution (population best fitness and population average fitness vs. generation) is shown in Figure 4. Figure 5 shows the gait diagrams for four of the best individuals from various generations (ϕ_{one}, ϕ_{two}, ϕ_{three}, ϕ_{four}, from generations 3, 5, 13, and 22 respectively, with ϕ_{four} being the best gait generated during the GA run). All four of the gaits performed fairly well, walking in a consistently forward direction during their evaluations. ϕ_{one} corresponds to the first sharp increase in fitness from previous generations. The improvement between ϕ_{one} and ϕ_{two} is due to the shorter period for the step cycle and the gait pattern itself is simpler. This trend continues in ϕ_{three} and ϕ_{four} with a much reduced step cycle period and a more consistent and simplified gait pattern. The fitness improvements corresponding to these gaits is indeed quite high compared to the earlier two individuals.

Figure 4 shows the fitness generated by each individual throughout the entire GA run. The rather high fitness value reported in generation 3 (Figure 4) is a result of an anomalous sensor output. This was confirmed when, in the next

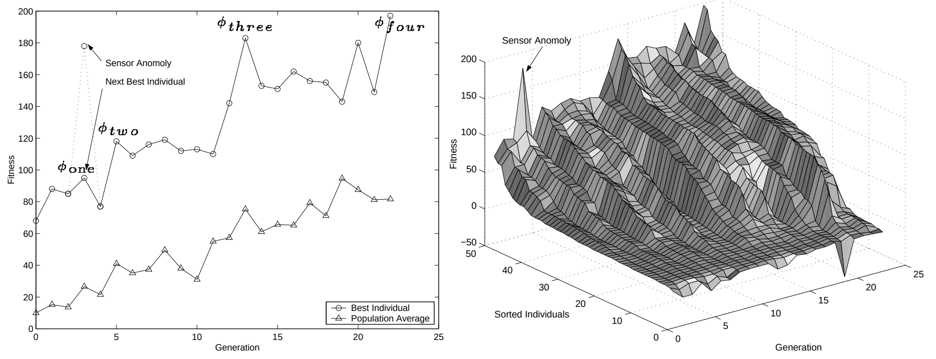


Fig. 4. Convergence History of a GA Run. (left) Best and average fitness plots over the evolution. (right) Fitness of entire population over the evolution (individuals ordered by fitness). Note there was one data point discounted as an odometer sensor anomaly. In the best individual convergence history it was replaced by the next best individual (as indicated on left graph).

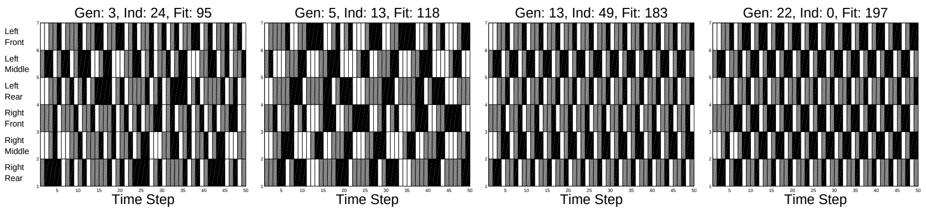


Fig. 5. Gait diagrams (time histories) for the four solutions, ϕ_{one},ϕ_{two},ϕ_{three},ϕ_{four}, respectively.

generation, that individual achieved an extremely low fitness (-17). The rest of the plot shows quite clearly the trend of increasing fitness for the kept segment of the population as well as the variance in fitness reported for different individuals. Due to the difficulty in running large populations on a hardware apparatus such as Kafka, the population size is somewhat low. However, as can be seen in Figure 4 by this fitness variance, the genetic diversity is maintained.

As mentioned earlier, the number of generations over which the GA was allowed to evolve was rather limited. This can been seen in Figure 4 by the fact that there is still a rather high deviation for the best individuals during the latter generations. The average fitness would not necessarily converge to the values reported for the best individuals, even if those gaits had saturated the population. This is due to the fact that the mutation rate of $p_m = 0.05$ is sufficiently high to keep many of the child individuals in each generation from being simply copies of the parents.

It should be pointed out that fitness does not monotonically increase with generation. There is a fluctuation in the fitness of the individuals after convergence. This is due to the hardware aspect of the experiment. Even the best K

individuals were re-evaluated each generation. As the experiment progressed, the rubber foot pads used by Kafka simply wore out. This loss of footing corresponded to a variable loss in traction and fitness for each gait.

The use of a state table with binary output greatly reduced the search space for this experiment. The next stage for this type of work involves using a full state lookup table (3 or more output possibilities). The expected result of this is the same as for the reduced version, however, the time to develop successful gaits would be increased, and the convergence history would likely be much more gradual (likely making it impractical). For example, if there were 9 output zones for each leg, the chromosome would have length 8019 and the search space would be of size 9^{8019}. Thus, our approach may not scale very well to increases in model complexity (due to the long training times required).

6 Conclusion

Many researchers believe that highly parallel and decentralized methods are the key to endowing artificial systems with intelligence. Decentralized controllers for insect robots offer a great deal of redundancy in that if one controller fails, the robot may still limp along under the power of the remaining functional legs [3]. The cellular automata controller approach outlined here was able to successfully control Kafka, a hexapod robot, and should extend to robots with more degrees of freedom (keeping in mind scaling issues). One advantage of using such a coarse controller is that it requires very few real-time computations to be made (compared to dynamic neural network approaches) as each leg is simply looking up its behaviour in a table. The approach easily lends itself to automatic generation of controllers through genetic algorithms (or any global optimization technique) as was shown for the simple examples presented here. Although it would be costly to evolve walking gaits on real robots (inverse design by optimization is always computational expensive), this may be done off-line (ahead of time). This work was motivated by evidence that biological insects generate walking patterns by means of decentralized control [4]. We hope that studying such types of control for artificial walkers may also in turn tell us something about the natural systems by which they were inspired.

Acknowledgments

Kafka, the hexapod robot featured in this work, was constructed by David McMillen as part of a Master's thesis at the University of Toronto Institute for Aerospace Studies.

The authors would like to thank Thierry Cherpillod for much help in setting up the experiment as well as the Natural Sciences and Engineering Research Council of Canada and the Canadian Space Agency for supporting this work.

References

1. T D Barfoot and G M T D'Eleuterio. An evolutionary approach to multiagent heap formation. Congress on Evolutionary Computation, July 6-9 1999.
2. Randall D. Beer, Hillel J. Chiel, and Leon S. Sterling. A biological perspective on autonomous agent design. *Robotics and Autonomous Systems*, 6:169–186, 1990.
3. Hillel J. Chiel, Randall D. Beer, Roger D. Quinn, and Kenneth S. Espenschied. Robustness of a distributed neural network controller for locomotion in a hexapod robot. *IEEE Transactions on Robotics and Autonomation*, 8(3):292–303, june 1992.
4. Holk Cruse. Coordination of leg movement in walking animals. In J. A. Meyer and S. Wilson, editors, *Simulation of Adaptive Behaviour: from Animals to Animats.* MIT Press, 1990.
5. Rajarshi Das, James P. Crutchfield, Melanie Mitchell, and James E. Hanson. Evolving globally synchronized cellular automata. In L.J. Eshelman, editor, *Proceedings of the Sixth International Conference on Genetic Algorithms*, pages 336–343, San Fransisco, CA, April 1995.
6. Dorigo and Colombetti. *Robot Shaping.* MIT Press, A Bradford Book, 1998.
7. David E. Goldberg. *Genetic Algorithms in Search, Optimization, and Machine Learning.* Addison-Wesley Pub. Co., Reading, Mass., 1989.
8. Takashi Gomi. Practical applications of behaviour-based robotics: The first five years. In *IECON98, Proceedings of the 24th Annual Conference of the IEEE*, pages 159–164, vol 4. Industrial Electronics Society, 1998.
9. Takashi Gomi and Ann Griffith. Evolutionary robotics - an overview. In *Proceedings of IEEE International Conference on Evolutionary Computation*, pages 40–49, 1996.
10. Takashi Gomi and Koichi Ide. Evolution of gaits of a legged robot. In *The 1998 IEEE International Conference on Fuzzy Systems*, pages 159–164, vol 1. IEEE World Congress on Computational Intelligence, 1998.
11. Freédéric Gruau. Automatic definition of modular neural networks. *Adaptive Behaviour*, 3(2):151–184, 1995.
12. Nick Jakobi. Runnig across the reality gap: Octopod locomotion evolved in a minimal simulation. In P Husbands and J-A Meyer, editors, *Proceedings of Evorob98.* Evorob98, Springer-Verlag, 1998.
13. Maja J. Matarić. Behaviour-based control: Examples from navigation, learning, and group behaviour. *Journal of Experimental and Theoretical Artificial Intelligence*, 9(2):232–336, 1997. Special Issue on Software Architectures for Physical Agents, Editors: H. Hexmoor, I. Horswill, D. Kortenkamp.
14. David Ross McMillen. Kafka: A hexapod robot. Master's thesis, University of Toronto Institute for Aerospace Studies, 1995.
15. Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction.* A Bradford Book, MIT Press, 1998.
16. Jon von Neumann. *Theory of Self-Reproducing Automata.* University of Illinois Press, Urbana and London, 1966.
17. Andrew Wuensche. The ghost in the machine: Basins of attraction of random boolean networks. In Chris G. Langton, editor, *Artificial Life III: SFI Studies in the Sciences of Complexity, vol. XVII.* Addison-Wesley, 1994.
18. David Zeltzer and Michael McKenna. Simulation of autonomous legged locomotion. In Chris G. Langton, editor, *Artificial Life III: SFI Studies in the Sciences of Complexity, vol. XVII.* Addison-Wesley, 1994.