A Flexible Evolutionary Method for the Generation and Implementation of Behaviors for Humanoid Robots

Downloaded from: https://research.chalmers.se, 2024-10-07 06:08 UTC

Citation for the original published paper (version of record):
Sandholt, H., Wahde, M., Pettersson, J. (2001). A Flexible Evolutionary Method for the Generation and Implementation of Behaviors for Humanoid
Robots. IEEE-RAS International Conference on Humanoid Robots: 279-286

N.B. When citing this work, cite the original published paper.

© 2001 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, or reuse of any copyrighted component of this work in other works.

(article starts on next page)

A flexible evolutionary method for the generation and implementation of behaviors for humanoid robots

Jimmy Pettersson, Hans Sandholt, Mattias Wahde

Division of Mechatronics, Chalmers University of Technology,
412 96 Göteborg, Sweden
{jimmy.pettersson, hans.sandholt, mattias.wahde}@me.chalmers.se

Abstract

A flexible method for generating behaviors for bipedal robots is presented and applied to the case of motor behaviors. The method is biologically inspired and is based on evolutionary algorithms in connection with generalized finite state machines (FSMs). The evolutionary process acts directly on the FSMs and optimizes both their parameters and their structure.

In this method, only a rough indication of the desired behavior needs to be specified as an initial condition to the evolutionary algorithm, which then performs further optimization of the behavior.

We apply the method to two test cases, namely energy optimization and robust balancing. It is found that the method performs very well in both cases, and that its ability to modify the structure of the FSMs is very useful. In the case of energy optimization, the walking length for a given amount of energy is improved by 134 %.

Keywords: bipedal robots, evolutionary robotics, behavior–based robotics

1. Introduction

During the early decades of the 21st century, it is expected that humanoid robots will come to play an increasingly important role, both in industries and as household robots. However, in order for this to happen, the robots will need to become much more complex than today, and the development of such robots presents a formidable challenge to researchers and engineers. As the complexity of humanoid robots increases, there will be a strong need for a flexible and versatile representation for motor behaviors (and other behaviors) [9]. In addition to a flexible representation, an efficient optimization method for generating robust and energy-optimal motor behaviors will also be needed.

The development of a representation and the choice of an optimization method are difficult problems. However, the fact that the systems that are being generated – humanoid robots – are modelled on biological systems – humans – indicates that it would be wise to consider optimization methods inspired by biological considerations, such as e.g. evolutionary algorithms.

The application of evolutionary computation to robotics has given rise to the very active research field of evolutionary robotics [12]. The use of evolutionary methods to the case of bipedal robots has mainly been restricted to parameteric optimization within a pre-specified structure (see e.g. [1], [3], [4], and [6]). Notable exceptions are provided by Arakawa and Fukuda [1], who allowed a certain flexibility in the representation of the control system and Paul and Bongard [13], who allowed the morphology of the bipedal robot to vary.

The aim of this paper is to introduce a flexible and general method for the construction of robotic behaviors. We will describe the representation of the behaviors, and also show how evolutionary optimization can be applied successfully to this representation, optimizing not only the parameters of the system but also its structure. While the focus of the paper is on the description of the method as such, we will also present some early results obtained with this method.

2. The robot

For our simulations, we have used a five-link robot, constrained to move in the sagittal plane. The robot has five degrees of freedom: torques can be applied at both knee joints and at both hip joints. In addition, a fifth actuator controls the posture of the upper body. The lengths of the leg links have been based on the corresponding values for a 1.5 m tall human.

The structure of our robot, which is shown in Fig. 1 is similar to that earlier used by e.g. Cheng and Lin [3] and Mitobe *et al.* [10]. While this robot model is perhaps somewhat simplistic, it is still sufficient

Copyright © 2001

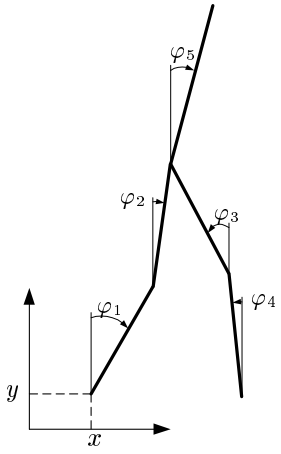

Figure 1: Configuration of the bipedal walking robot.

for the purposes of demonstrating the feasibility of our method for representing behaviors for bipedal robots. We have used a lagrangian formulation for the equations of motion (see e.g. [11], Ch.4), which take the form

$$\mathbf{M}(\mathbf{z})\ddot{\mathbf{z}} + \mathbf{C}(\mathbf{z},\dot{\mathbf{z}})\dot{\mathbf{z}} + \mathbf{N}(\mathbf{z}) + \mathbf{A}^{\mathrm{T}}\boldsymbol{\lambda} = \boldsymbol{\Gamma}, \quad (1)$$

where $\mathbf{M}$ is the generalized inertia matrix, $\mathbf{C}$ contains centrifugal and Coriolis terms, $\mathbf{N}$ contains gravity terms, $\mathbf{A}$ is the constraint matrix and $\boldsymbol{\lambda}$ the corresponding Lagrange multipliers, and $\boldsymbol{\Gamma}$ contains the generalized forces. The derivation of the various matrices and vectors is straightforward, and thus will not be given here. The generalized coordinate vector $\mathbf{z}$ is given by

$$\mathbf{z} = [\varphi_1, \ldots, \varphi_5, x, y]^{\mathrm{T}}, \quad (2)$$

where the angular variables $\varphi_1, \ldots, \varphi_5$ determine the orientation of the limbs (see Fig. 1), and x, y are the coordinates for one foot (i.e. the tip of a leg) of the robot.

The vector of generalized forces $\boldsymbol{\Gamma}$ is related to the torques $\mathbf{T}$ applied at the five joints through the transformation $\boldsymbol{\Gamma} = \mathbf{D}\mathbf{T}$, where

$$\mathbf{D} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & -1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \quad (3)$$

The constraint matrix $\mathbf{A}$ varies in size and structure depending on the number of feet (0, 1, or 2) that are in contact with the ground [7].

Lagrange's equation for impulsive motion is used to model ground impacts and perturbations and is stated as

$$\left.\frac{\partial \mathrm{T}}{\partial \dot{\mathbf{z}}}\right|_{t^+} - \left.\frac{\partial \mathrm{T}}{\partial \dot{\mathbf{z}}}\right|_{t^-} = \hat{\mathbf{Q}}\,, \quad (4)$$

where t^+ and t^- denote the instants immediately after and immediately before the impulse, respectively, $\hat{\mathbf{Q}}$ is the vector of generalized impulses, and T is the kinetic energy of the system. Using the fact that the generalized inertia matrix ($\mathbf{M}$) is symmetric, the generalized momenta can be expressed as: $\partial \mathrm{T}/\partial \dot{\mathbf{z}} = \mathbf{M}\dot{\mathbf{z}}$, which, when inserted into Eq. (4), gives the generalized postimpact velocities as

$$\dot{\mathbf{z}}^+ = \mathbf{M}^{-1}\hat{\mathbf{Q}} + \dot{\mathbf{z}}^-\,. \quad (5)$$

3. The method

The implementation of motor behaviors (and other behaviors) in robots consists of two parts which will now be introduced: an architecture for storing the behaviors of the robot, and a method for obtaining the behaviors that are to be implemented.

3.1 The representation

While this paper will deal exclusively with bipedal *motor* behaviors, the ultimate goal of this work is to arrive at a method which is sufficiently general to be able to accomodate not only bipedal gaits but also other aspects of the behavior of a robot[1], such as the ability to avoid obstacles, grip objects etc. Thus, an architecture which can *only* hold fully specified reference trajectories for bipedal gaits will not be sufficient.

Instead, we have chosen to use an architecture based on (generalized) finite state machines (FSMs). FSMs have the advantage of allowing combination of several behaviors into a complete behavioral repertoire [14], and they have often been used in connection with behavior-based robotics [2]. Furthermore, a system based on FSMs is generally transparent and easy to interpret.

A standard FSM consists, as the name implies, of a finite number of states and conditional transitions between those states. Furthermore, the allowed set of actions is usually chosen from a finite alphabet. The FSMs introduced in this paper are slightly different. First, each state in an FSM is here associated with a set of variables specific to that state, whereas in a standard FSM, the variables are associated with the transitions between states. In addition, we use

[1]For this reason, we will use the term *robotic brain* for the computer program that determines the actions of the robot, rather than the term *control system*. The latter term would indicate a more limited representation employing classical control theory.

Copyright © 2001

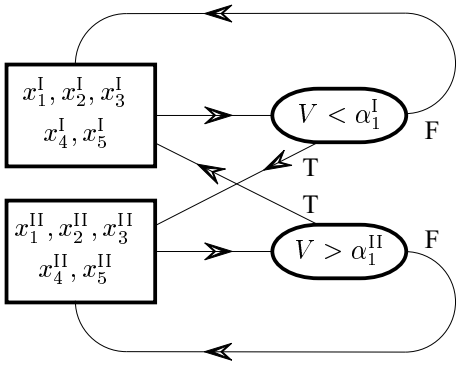

Figure 2: A simple two-state FSM, with five state variables and one transition condition per state. The arrows indicate the direction of signal flow. If the condition under consideration is true, the corresponding arrow marked with a T is followed. If instead the condition is false, the arrow marked with an F is followed.

continuous variables rather than a discrete alphabet. Each state has a number of conditional transitions, each with a specified target state.

A simple, generic, example of a two-state FSM is shown in Fig. 2. In this FSM both states contain the values of five variables (which may, for example, represent the reference angles for a given posture for the five–link bipedal robot). From the first state, the FSM can jump to the second state if the condition $V < \alpha_1^{\mathrm{I}}$ is fulfilled. Note that the variables V (of which only one was introduced in Fig. 2) defining the transition conditions need not be the same as the variables x_i^s specified in the states s. In this case, the condition variable V may, for instance, measure the deviation between the actual posture of the robot, and the posture specified in the active state. If the deviation is sufficiently small, the robot may proceed to the second state etc.

If no condition is fulfilled, the FSM remains in the same state, as indicated in Fig. 2 by the links emanating on the right hand side of the transition conditions. Note that, in subsequent figures, these links are not explicitly shown.

The number of transition conditions, as well as the number of variables defining the conditions, may vary from state to state. In cases where there is more than one transition condition associated with a state, the conditions are checked in order from left to right, so that the leftmost condition has the highest priority, since it is always checked.

3.2 The evolutionary algorithm

Evolutionary algorithms constitute, in our opinion, a natural choice for the generation of motor behaviors and other behaviors for autonomous robots in general, and bipedal robots in particular. After all, it is known that evolution is capable of generating highly complex structures in nature, and that evolutionary algorithms, which are based on natural evolution, often prove to be highly efficient in problems involving large and complicated search spaces. Clearly, the construction of robotic motor behaviors, which is the subject of this paper, is indeed a problem involving a very large search space.

The most commonly used type of evolutionary algorithm is the genetic algorithm (GA) [8]. Most of the work to date on evolutionary algorithms in connection with bipedal robots has been based on GAs ([1], [3], [4], and [6]). However, standard genetic algorithms may not the best choice from the point of view of the construction of robotic brains. A standard GA is useful when carrying out parametric optimization, where the parameters of the system under study easily can be coded into a string of digits.

However, we wish to go beyond parametric optimization, and optimize not only the parameters but also the *structure* of the robotic brain. Thus, a more flexible scheme is required. The use of evolutionary algorithms in connection with FSMs, known as evolutionary programming, was pioneered by Fogel (see e.g. [5]). In evolutionary programming, the evolutionary process acts directly on the FSMs, by optimizing both the parameters of the FSMs and their structure, e.g. the number of states and transition conditions.

Our method is an adaptation of evolutionary programming to the case of generalized FSMs as described above, and it includes both crossover and mutation operators, by contrast with the original form of evolutionary programming which only used mutation operators.

Briefly, the process operates as follows: A fitness measure is specified before the simulation. An example of a fitness measure suitable for bipedal locomotion is given by the distance covered by the robot as it uses up a pre-specified amount of energy. In the beginning of a simulation, a population of random FSMs is generated. Normally, the initial population consists of rather simple FSMs. Then, all individuals in the population are evaluated, and each individual obtains a fitness value based on its performance.

The following sequence is then repeated until a satisfactory solution has been found: two individuals are selected from the population using tournament selection. Then, two offspring are formed by the procedures of crossover and mutation outlined below. The two new individuals are then inserted into the population, replacing the two worst individuals. Finally the two new individuals are evaluated, and the procedure is repeated again.

Copyright © 2001

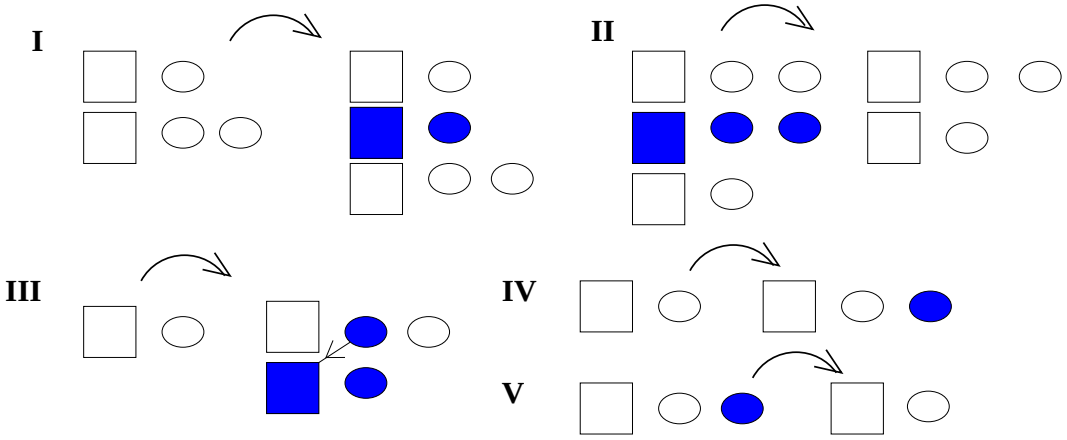

Figure 3: Structural mutations: I) *Insert state*: inserts a state with one transition condition, whose variables are defined as the average of the variables in the two adjacent states, II) *Delete state*: simply removes a state, III) *Add prioritized state*: adds, to an already present state, a transition (with top priority) to a new state. The variables of the new state are taken as slight mutations of the variables in the state to which the new transition was added, IV) *Add transition condition*: adds a transition condition (with lowest priority) to a state, and, V) *Delete transition condition*: deletes the transition condition with lowest priority for a given state. Note that, for clarity, the transitions are not explicitly shown (except one transition in case III) in this figure.

3.2.1 Crossover

Combination of material from different individuals is an important part of evolutionary algorithms. Crossover is easy to implement in a standard GA, but somewhat more difficult in our case, in which the structures to be crossed are more complicated than the strings used in GAs. We have chosen to introduce a crossover procedure which simply swaps two selected states between two FSMs. The procedure begins by the selection of one state in each of the FSMs that are to be crossed. Next, the states with their transition conditions are swapped between the FSMs, forming two new FSMs. As a final step, it is checked that the targets for the conditional jumps are consistent, i.e. that no condition generates a jump to a non-existent state (which may occur if the FSMs contain different numbers of states). If an inconsistent jump is detected, the target is arbitrarily set to state 1. This does not imply a significant restriction, since subsequent mutations can change the transition target to any of the available states.

3.2.2 Mutations

Two kinds of mutations are used: *parametric mutations*, which modify the value of any parameter in the FSM by a small, random amount, and *structural mutations* which modify the structure of the FSMs. The structural mutations, which are needed in order to arrive at the desired flexibility, are illustrated in Fig. 3.

3.3 The simulation program

The generalized FSM representation and the evolutionary algorithm described above have been implemented in a computer program written in Delphi Object-oriented Pascal. The program is fully object-oriented, so that the data structures, e.g. the FSMs, are flexible and can be of arbitrary size and complexity. Thus, the program permits an open-ended evolutionary process that can lead to very complex structures.

At the outset of a simulation, the user provides a set of parameters, such as link lengths and masses (for the robot), the fitness measure, initial structural parameters for the FSMs (e.g. the number of states) as well as ranges for the parameters (variables and transition conditions) defining the states. Parameters related to the simulation of a single individual, such as e.g. the length of the time steps for the numerical integration of the equations of motion, must also be specified. Furthermore, it is possible to provide limits on the joint torques and their first derivative with respect to time.

The user may also choose between two different types of initial FSMs, *linear FSMs*, in which each state s has a single transition condition whose target is state $s + 1$, except for the last state, for which the target of the transition condition is state 1, and *general FSMs*, with a completely arbitrary structure. The linear FSMs are useful for generating cyclic behaviors, such as a step sequence, whereas the more flexible general FSMs are needed e.g. to cope with perturbations during a step or other non-cyclic motor behaviors. Note that the specification of an FSM

Copyright © 2001

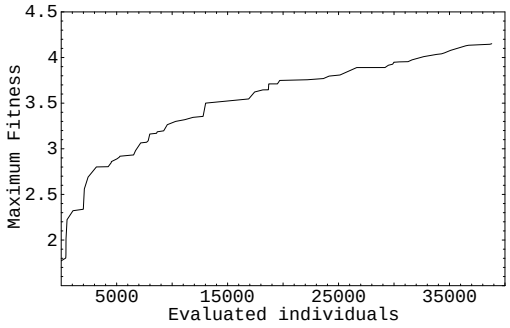

Figure 4: Fitness of the best individual as a function of the number of individuals for test case 1 (energy optimization).

type only relates to the *initial* population. The evolutionary process has full freedom to add and delete states, as outlined above, should the need arise.

In keeping with the aim of developing a sufficiently flexible representation that can hold different kinds of behaviors, great care has been taken to make the data structures for the FSMs as general as possible. Thus, an FSM can consist of states of many different types (i.e. with different variables defining the states), and with various transition conditions of, in principle, any form.

However, here we are concerned with motor behaviors, and we have therefore used a specific kind of FSM, the components of which will now briefly be described.

FSM states In any state of the FSMs used here, the requested torque at joint i is given by

$$\tau_i^{\mathrm{req}} = K_i^P(\varphi_i - \varphi_i^{\mathrm{ref}}) + K_i^D\dot{\varphi}_i + K_i^0 \quad (6)$$

where K_i^P, φ_i^{ref}, K_i^D, and K_i^0 are constants. Thus, for the representation of motor behaviors, each FSM state holds a set of 20 variables (4 for each link). Since we, for realism, normally impose limits on the torque derivatives, the actual torque delivered at a joint is not always equal to the requested torque. In most situations, however, the actual torque approaches the requested torque within a few time steps.

Transition conditions For each state s, there are N^s transition conditions which, in this case, take the form

$$\text{if } (V_i\,[\mathrm{Op}]\,\alpha_k) \text{ then jump to target}, \quad (7)$$

where [Op] denotes one of the operators $<$ and $>$, α_k is a constant, specific to transition condition k, and the target is any state in the FSM (cf. Fig. 2).

	Early FSM	Best FSM
Energy used (J)	500	500
Length walked (m)	1.77	4.15
Total time (s)	2.56	4.11
Average speed (m/s)	0.69	1.01

Table 1: A comparison between the first individual that managed to walk (left) and the best individual, in test case 1.

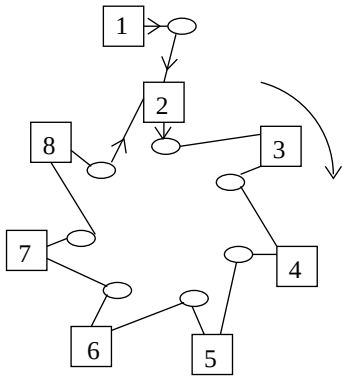

Figure 5: Structure of the best FSM obtained in test case 1 (energy optimization).

The variables V_i can be choosen freely. In this application, we have chosen to use six condition variables, namely

$$V_i = \varphi_i - \varphi_i^{\mathrm{ref}},\ i = 1, \ldots, 5, \quad (8)$$

and

$$V_6 = \sqrt{\frac{1}{5}\sum_{i=1}^{5}(\varphi_i - \varphi_i^{\mathrm{ref}})^2}. \quad (9)$$

4. Results

In order to test the efficiency of both the representation and the evolutionary algorithm, a number of runs of the simulation program have been made. Two specific applications have been used as test cases, namely the generation of smooth and energy-efficient bipedal gaits, and the construction of robust balancing in the presence of perturbations.

4.1 Test case 1: Energy optimization

For any autonomous robot that carries its own energy source (e.g. batteries), it is clearly of paramount importance to move with as little use of energy as possible. In nature, evolution has optimized human walking (and, in general, animal locomotion), to make it very energy efficient. While we do apply

Copyright © 2001

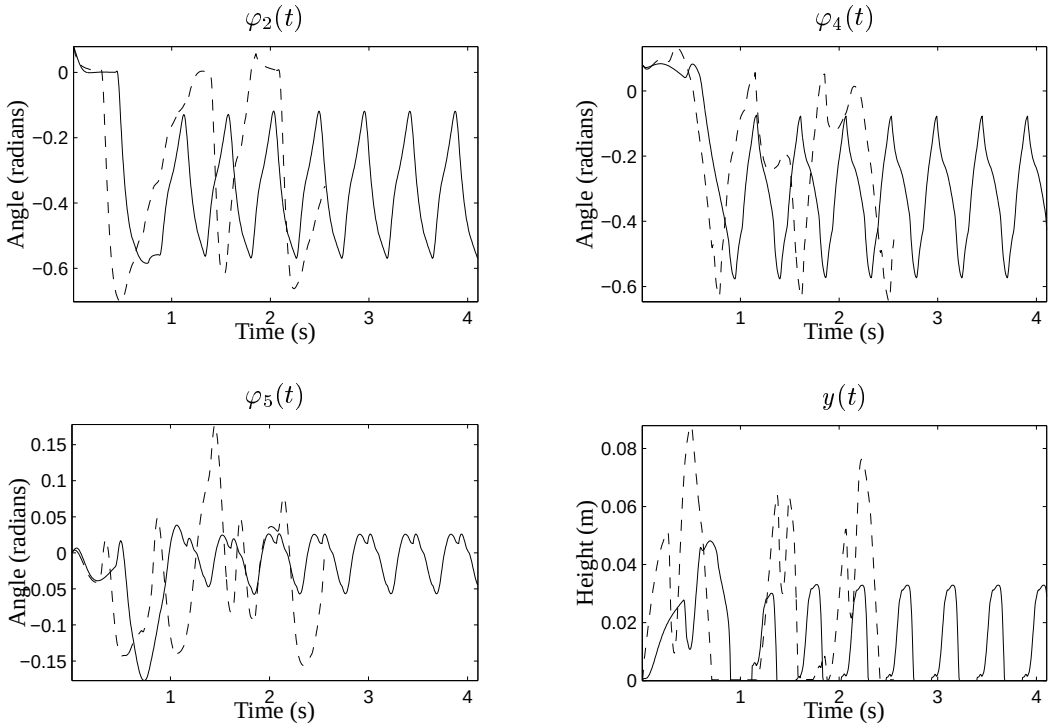

Figure 6: Energy optimization. Each plot shows the variation with time of one generalized coordinate for the first FSM that was able to make the robot walk (dashed) and the best FSM in the run (solid). Only some of the 7 generalized coordinates are shown in this figure.

artificial evolution to optimize the gait of our simulated robot, it should be pointed out that our optimization problem differs from the optimization carried out by natural evolution. In our case, the configuration of the robot, i.e. its bipedal nature and its structure with five links of given length and mass, are given whereas in natural optimization both the structure of the animal and its method of locomotion are optimized. However we do, as described above, allow a considerable freedom concerning the structure of the *brain* of the robot.

For the energy optimization runs, the fitness measure was chosen as the length walked by the robot until it had used an energy of 500 J. By using this fitness measure, energy optimization is obtained without explicitly having to include the energy usage in the fitness measure in an ad hoc fashion. In order to prevent the robot from walking very slowly, a time limit of 6 simulated seconds was introduced as well.

The population size was set to 400, and the structural and parametric mutation rates were set to 0.02 and 0.03, respectively. The crossover probability was equal to 0.10. The time step length was 0.005 seconds. Furthermore, limits were set on the maximum torque delivered at the joints (200 Nm), as well as the maximum rate of change of the torques (3000 Nm/s).

One of the main purposes with our method is to allow for the possibility of specifying, in a very loose sense, a sequence of motions, which will then be further optimized by evolution. In the development of energy-optimized gaits, we therefore specified only 8 reference states, 4 for the step with the left foot, and 4 for the right step.

The reference angles were set so as to generate a very rough representation of the two steps. The proportional and derivative constants were given random values centered on -250 Nm for the proportional constants and -15 Nms for the derivative constants. The K_i^0 parameters were given random values in the range $[-10, 10]$ Nm. The initial population consisted of linear FSMs (see Sect. 3.3).

In the beginning of the run, it was clear that the initial specification of the motion was much to rough to generate smooth walking: the few robots that managed to walk at all, stumbled forward in a very inefficient manner. Many robots used up their 500 J without getting anywhere. However, the optimization algorithm very quickly began to improve the gait, and the length walked by the robot increased considerably, from 1.77 m early in the run, to 4.15 m at the end, as shown in Fig. 4 and Table 1.

The total number of states of the best FSM at the end of the run was also equal to 8. However, this was in no way enforced. Indeed, during the run, several of the best FSMs that appeared used more than 8

Copyright © 2001

states. The 8 states of the final FSM were totally different from the states specified in the beginning of the run.

Furthermore, the evolutionary optimization method was able to improve the structure of the FSM. Clearly, a cyclic sequence of states is convenient when walking at full speed. However, the robot starts from rest, and thus the very first part of the motion differs from the rest. This was indeed exploited: the structure of the best FSM at the end of the run contained one state that was used only to get the robot started, and 7 states that were used in a cyclic fashion for the continued motion, as shown in Fig. 5.

Finally, we note that the bipedal gait generated by the best FSM in the run was very smooth (see Fig. 6) and symmetric compared to the FSMs obtained early in the run, despite the fact that symmetry was not explicitly required.

4.2 Test case 2: Robustness

A bipedal robot moving in an unstructured environment, such as e.g. a busy street or a hospital, will invariably find itself in situations where it cannot rely on prespecified reference trajectories. For example, the robot may encounter an unexpected moving obstacle, or it may lose its balance due to an external perturbation or simply a bump in the ground. Thus, for such robots to be useful, they must be able to cope with unexpected situations. As a simple example, and as a test of our method, we have considered the following case: Assume that a bipedal robot is about to begin climbing some stairs, and as it lifts the front leg, it is perturbed. A sequence of three point perturbations, modeled as impulsive forces, are applied. The generalized velocities after each perturbation are computed using Eq. (5). The first perturbation is applied on the thigh of the supporting leg, the second on the upper body, and the third on the lower part of the lifted leg, as shown in the right panel of Fig. 7.

At the start of each simulation, the robot was placed with both feet on the ground, and the FSM of the robot contained a single state which made it lift the front leg. The fitness measure was defined simply as the inverse of the integrated total deviation between a desired position, with one leg lifted as shown in Fig. 7, and the actual position of the robot. The total deviation was computed as the root mean square of the deviation of each generalized coordinate. The fitness computation began after 0.6 s, giving the robot some time to reach the desired position from its starting position. Each simulation lasted for the equivalent of 3.6 s, and the three perturbations were applied after 0.8 s (perturbation a, see Fig. 7), 1.4 s (b), and 2.0 s (c), respectively. The

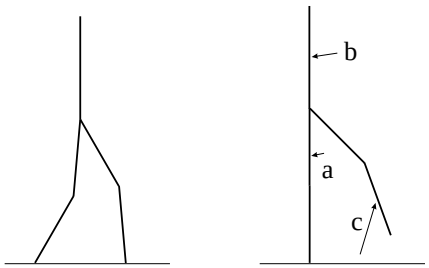

Figure 7: Starting posture (left) and desired posture for the robot in test case 2. The arrows indicate the magnitudes, directions, and points of application of the perturbations.

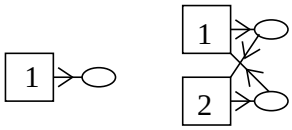

Figure 8: Initial (left) and final structure of the FSMs from test case 2. The added state helps the robot cope with the perturbations.

simulated robots were given a maximum of 500 J of energy to lift the leg and to handle the perturbations. The parameters of the evolutionary algorithm were the same as in test case 1 (see Sect. 4.1).

While the initial FSMs generally had severe difficulties in keeping the robot upright, FSMs capable of doing so appeared fairly quickly as a result of the optimization. More interestingly, the final FSM obtained from this run had undergone a structural mutation in which an additional state was added to cope with the perturbation. A schematic view of the structure of the initial FSMs and the best FSM obtained is shown in Fig. 8.

5. Discussion and Conclusion

In this paper, we have introduced a method for the generation of motor behaviors in bipedal robots. With our procedure, it is sufficient to provide the optimization algorithm with a rough indication of the desired motor behavior (rather than a complete trajectory specification), and then allow the algorithm to optimize it.

Ideally, it should be possible to generate a bipedal gait, or some other motor behavior, without specifying even a rough set of reference values. However, if no specification is made at all, it is not evident that a human-like gait will result. For instance, the evolutionary process may select a bird-like gait instead. Thus, some guidance should be given to the optimization algorithm, for instance in the form of a few reference positions as in our method.

Copyright © 2001

It is obvious that for bipedal robots to be useful, they must be able to cope with unstructured and unpredictable environments. Our procedure may be useful in the construction of such robots, chiefly because of the structural flexibility of the corresponding robotic brains and the fact that the optimization method proceeds with a minimum of bias.

We believe that the ability to optimize the structure of the robotic brain, in addition to its parameters, is of great importance, and allows a kind of open-ended evolutionary process, which can produce structures that are much more complex than those initially specified. A possible indication supporting this hypothesis is the fact that the fitness values continued to increase during the full extent of the runs, rather than reaching a plateau quickly, as is often the case in evolutionary algorithms. A stronger indication is derived from the fact that the possibility to modify the structure of the FSMs was exploited in both of the test cases considered here. Thus, even though it probably would be possible, at least for simple gaits, to specify a useful FSM by other means (or even by hand), it has been our policy to give the evolutionary optimization method as much freedom as possible.

The two test cases also showed that considerable improvements could be obtained in a reasonable amount of time. In the case of energy optimization, a 134% improvement in walking length was obtained in a run that lasted approximately 28 hours on an 800 MHz pentium III computer.

The results presented here are, to a great extent, preliminary, and further experiments are underway to test the procedure in more challenging situations. The aim is to develop a full behavioral repertoire for bipedal locomotion using the procedure described in this paper, and to combine these behaviors using e.g. the method for evolutionary combination of separate behaviors described by Wahde and Sandholt [14]. Furthermore, we plan to implement the resulting robotic behaviors in the bipedal robot which is currently under development in our group.

References

[1] T. Arakawa and T. Fukuda, Natural Motion Trajectory Generation of Biped Locomotion Robot using Genetic Algorithm through Energy Optimization. In: *Proc. of the 1996 IEEE International Conference on Systems, Man and Cybernetics*, pp. 1495-1500, 1996

[2] R.C. Arkin, *Behavior-based robotics*, MIT Press, Cambridge, MA, 1998

[3] M.-Y. Cheng and C.-S. Lin, Genetic Algorithm for Control Design of Biped Locomotion. In: *Proc. of the 1995 IEEE International Conference on Systems, Man and Cybernetics*, pp. 1315-1320, 1995

[4] S.-H. Choi, Y.-H. Choi, and J.-G. Kim, Optimal Walking Trajectory Generation for a Biped Robot Using Genetic Algorithm. In: *Proc. of the 1999 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 1456-1461, 1999

[5] L. Fogel, *Intelligence through simulated evolution*, Wiley, NY, 1999

[6] T. Fukuda, Y. Komata, and T. Arakawa, Stabilization Control of Biped Locomotion Robot based Learning with GAs having Self-adaptive Mutation and Recurrent Neural Networks. In: *Proc. of the 1997 IEEE International Conference on Robotics and Automation*, pp. 217-222, 1997

[7] J. Furusho et al., Realization of Bounce Gait in a Quadruped Robot with Articular-Joint-Type Legs. In: In: *Proc. of the 1995 IEEE International Conference on Robotics and Automation*, pp. 697-702, 1995

[8] J.H. Holland, *Adaptation in Natural and Artificial Systems*, 1st ed. University of Michigan Press, Ann Arbor; 2nd ed. MIT Press, Cambridge, MA, 1992

[9] F. Kanehiro et al., Developmental Methodology for Building Whole Body Humanoid System. In: *Proc. of the 1999 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 1210-1215, 1999

[10] K. Mitobe et al., Non-linear feedback control of a biped walking robot. In: *Proc. of the 1995 IEEE International Conference on Robotics and Automation*, pp. 2865-2870, 1995

[11] R.M. Murray, Z. Li, and S.S. Sastry, *A Mathematical Introduction to Robotic Manipulation*, CRC Press, 1994

[12] S. Nolfi and D. Floreano, *Evolutionary Robotics*, MIT Press, Cambridge, MA, 2000

[13] C. Paul and J.C. Bongard, The Road Less Travelled: Morphology in the Optimization of Biped Robot Locomotion. In: *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS2001)*, in press

[14] M. Wahde and H. Sandholt, Evolution of complex behaviors on autonomous robots. In: *Proc. of Mechatronics 2000, the 7th UK Mechatronics Forum International Conference*, Elsevier, 2000

Copyright © 2001