Multi-Phase Sumo Maneuver Learning

Jiming Liu Shiwu Zhang

Abstract

In this paper, we demonstrate a *multi-phase genetic programming* (MPGP) approach to an autonomous robot learning task, where a *sumo wrestling* robot is required to execute specialized pushing maneuvers in response to different opponent postures. The sumo robot used has a very simple, minimalist hardware configuration. This example differs from the earlier studies in evolutionary robotics in that the former is carried out on-line during the performance of a robot. whereas the latter is concerned with the evolution of a controller in a simulated environment based on extended genetic algorithms. As illustrated in several sumo maneuver learning experiments, strategic maneuvers with respect to some possible changes in the shape and size of an opponent can readily emerge from the on- line MPGP learning sessions.

Index Terms

Multi-phase genetic programming (MPGP), autonomous robots, sumo tasks, maneuver learning, evolutionary robotics.

I. Introduction

In recent years, researchers have tackled various computational and applicational issues in evolutionary robotics. Generally speaking, evolutionary robotics has been concerned with the use of evolutionary algorithms to obtain control strategies or structural configurations for a robot. Examples of evolutionary algorithms used include genetic algorithms, genetic programming, and their variations. By applying evolutionary algorithms, robot designers need only to specify the optimal performance or structure requirements of a robot rather than to hard-code the detailed

Dr. Jiming Liu is with the Department of Computer Science, Hong Kong Baptist University, Kowloon Tong, Hong Kong. *E-mail: jiming@comp.hkbu.edu.hk*

Mr. Shiwu Zhang is with the Department of of Precision Machinery & Precision Instrumentation, University of Science and Technology of China.

motion primitives that control how the robot may interact with its environment. Once the initial conditions and parameters of an evolutionary algorithm are defined, the robot that utilizes such an algorithm can readily evolve its reactive behavior, without human interventions, in order to adapt to the external environment.

Broadly speaking, the problems of evolutionary robotics can be viewed from three aspects: (1) the end product of evolution – what is to be evolved, (2) the process of evolution – which evolutionary method is to be used, and (3) the implementation – whether the evolution is implemented on a physical robot or just in a simulated world – in order to become tractable, the right balance is desired that takes into consideration the computational costs and the evaluation of candidate solutions.

In what follows, we will highlight some of the previous work in evolutionary robotics corresponding to the above-mentioned three aspects.

A. *The End Product of Evolution – What to Evolve?*

Previous studies in evolutionary robotics have investigated various robotic tasks for complex (*e.g.*, unstructured, dynamically changing) environments. Readers can readily find examples of such tasks as well as the general issues involved from some representative papers, such as [1], [2], [3], [4], [5], [6]. The end product of applying an evolutionary algorithm in those tasks frequently concerns an effective motion control system for a physical or simulated robot. The evolved controllers often include artificial neural networks, classifier systems, and computer control programs in either high-level languages or machine codes.

For instance, Miglino *et al.* have applied a genetic algorithm to evolve the weights of an artificial neural network for controlling a robot in an enclosed environment. The performance requirement of the robot is to move in high-speed while avoiding any collision with obstacles

[7]. The main reason that the design of a neural network is employed for a robot control system is that the neural network design is well suited to the control problem. That is, the inputs of a neural network correspond to a robot's sensory inputs whereas the outputs of a neural network control the motion execution units of the robot. Other researchers have adopted similar methods of evolving motion controllers to enable robots to perform different tasks, although with slight differences in the forms of neural networks or genetic algorithms implemented. Among them, Nolfi *et al.* have developed a physical robot named *Khepera* to perform the tasks of grasping and releasing objects or avoiding obstacles [8], [9]. Floreano and Mondada have explored the emergence of navigation and obstacle avoidance behaviors in the *Khepera* robots [10]. Gomez and Miikkulainen's research deals with evolving a complex robot behavior by using an incremental NE (neural-evolution) method [11].

Recently, Floreano and Urzelai have noted that it is more credible to evolve the mechanism of parameter selection than to directly evolve parameters themselves [12], [13]. In the evolutionary robotics example used in our work, we are concerned with the evolution of a robot's action sequences, called *maneuvers*, in response to different opponent postures in a sumo contest. In this work, the evolution involves the search for effective strategic maneuvers among many candidates.

B. The Process of Evolution – How to Evolve?

In the above-mentioned studies, genetic algorithms have been often used to evolve the parameters of neural networks acting as controllers. Other evolutionary methods that have also been applied include genetic programming, self-organizing adaptation, as well as their variations. As they are related to our present work, let us take a look at some of the previous studies that explored the use of genetic programming to evolve computer control programs for achieving various adaptive behaviors.

Reynolds developed a system that utilized a method of steady-state genetic programming

(SSGP) for evolving obstacle-avoidance and corridor-following behaviors [14], [15]. Nordin and Banzhaf illustrated a compiling genetic programming system (CGPS) that evolved a robotic control program in machine codes [16]. Besides robot motion control, Martin demonstrated a method of evaluating traditional robot vision programs using simulated CCD images as inputs [17].

The rationale for applying genetic programming in the above examples is that genetic programming has a flexible chromosome design, which enlarges the scope of evolutionary objectives and robot control strategies. In our present work, we propose an approach, called multi-phase genetic programming, to evolving robot specialized maneuvers in a sumo contest. The advantage of evolving the maneuvers of different composition granularities during the different phases of evolution is that it can accelerate the convergence of adaptive behavior learning.

In addition to genetic programming, other designs of evolutionary algorithms have also been proposed, catering to certain robotic tasks or performance objectives. For instance, Hornby *et al.* combined an evolutionary algorithm with a Lindenmayer system in order to create a generative design system of a specific objective [18], [19]. Colombetti and Dorigo evolved a classifier system using a distributed genetic algorithm. Their results indicated that complex behaviors, such as preying and escaping, can readily emerge from basic behaviors [20]. Steels implemented an on-line selectionist mechanism for a robot to acquire its new behavior [21]. Xiao *et al.* performed extended genetic operations directly on a path population in order to find an optimal solution in a robot path-planning task [22].

C. The Problem

This work is concerned with the design and development of an on-line evolutionary robotic system that incorporates evolution based on simulated robot-environment interaction and real-world validation with the best individual from each generation of the evolution. Central to this on-line behavior learning system is a multi-phase genetic programming (MPGP) approach that is aimed at enabling the robot to gradually acquire sumo maneuvers.

Our present work differs from Nordin and Banzhaf's on-line genetic programming for evolving behavior to control a miniature robot in real time [16], [23]. As in our work, the evolved individual is a sequence of basic behaviors, instead of machine codes as in Nordin and Banzhaf's work.

II. Multi-Phase Genetic Programming (MPGP)

Genetic programming (GP) applies genetic manipulations to the functions and operators of a control program or other representations of a controller in an autonomous system. It was firstly proposed by John Koza in 1992 [24]. Generally speaking, in GP, the entities to be evolved consist of a function set and a terminal set. The function set often corresponds to the set of ordinary arithmetic functions and conditional operators, whereas the terminal set corresponds to the set of variables and constants. For detailed background on GP, readers are referred to [24].

GP and their variations can readily be implemented to model and acquire a robot's adaptive behavior in an unknown environment. As illustrated by an experimental example given in [24], GP has been used to evolve a subsumption architecture for simulated robots capable of performing reactive wall-following and box-moving behaviors in an environment.

A. Characteristics

In commonly-used GP formulations, there is no fixed structure for evolved objects. As a result, the space for searching an optimal program can become enormously large. Another shortcoming that may limit real-world GP applications is that the performance of GP can sometimes be undermined during the evolution of a program as the genetic manipulations are carried out at a *predefined granularity*, *i.e.*, there is no differentiation among the function or terminal sets as far as their complexities are concerned. We believe that in order to make GP most effective, the granularity of function and terminal sets should match the granularity of task constraints. That is, the right GP granularity should match the right problem characteristics (the *characteristics* may be reflected in the sensory measurement of a robot).

In order to speed up the evolutionary process in GP and, at the same time, to enable the dynamic adaptation of GP granularity, we have developed a multi-phase genetic programming (MPGP) approach. In the formulation of MPGP, the evolutionary process varies its granularity in light of the characteristics of a problem on hand. The granularity of MPGP is adjusted by changing the granularity of function and terminal sets through different phases.

III. The Proposed Method of Learning Sumo Maneuvers

In this case study of MPGP, we are interested in evolving effective maneuvers for a sumo robot with respect to certain performance requirements. This problem is in essence a problem of genetic programming that is aimed at synthesizing strategic maneuvers for the robot.

In our implementation, the evolutionary process is operated in two different phases by adopting two action granularities, general and specialized, respectively. During the first phase, the values of units from a terminal set are fixed or limited within a scope. Thus, the fitter functions are evolved first and saved for the next phase evolution. During the second phase, the terminal set is evolved with the functions that are limited in the fitter function set as evolved from the previous phase.

In this section, we will describe the task, the sumo robot, the MPGP formulation of learning, and the detailed algorithm for our case study.

A. The Robot Task

The task addressed in this paper is as follows: A sumo contest is to be played in a closed rectangular environment of size $M \times N$. One robot player of palm size ($PM \times PN$) is required to perform necessary sumo maneuvers that can effectively push its opponent out of the contest arena.

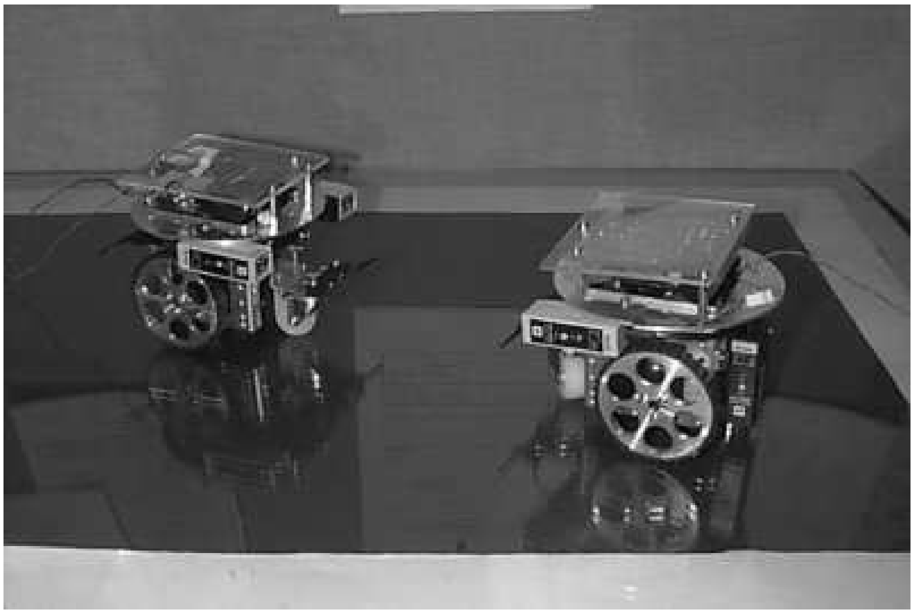

Fig. 1. A rectangular sumo contest arena.

Here, by necessary sumo maneuvers it is meant successful eye-body-coordinated motions against opponents of varying postures and weights.

For the sake of maneuver learning, the player is allowed to communicate with an offboard controller responsible for passing advice to the player in order to improve its performance. Suppose that the opponent to this player is capable of showing different standing postures as well as different degrees of resistance (as if different weights) in an attempt to undermine the strength of the player.

Figure 1 shows a sumo contest arena with two mobile robots. The physical robot can execute a maneuver as evolved by an offboard controller, and measure and communicate the performance of the maneuver back to the controller. The evolution and selection of maneuvers by the controller are based on the above-mentioned multi-phase genetic programming (MPGP) approach.

A.1 The Sumo Agent and its Controller

The sumo player, named Junior, is a physical micro-mobile robot. Its objectives are to follow the maneuver commands from an offboard controller and to try out various strategic maneuvers against its opponent. Junior can readily utilize its two limit-switch arms, two infra-red eyes, an onboard actuator controller with encoder-feedback, a memory board, and a communication device. This offers Junior a number of basic capabilities. For instance, the arms can effectively push and at the same time sense its opponent. The two infra-red sensor-based eyes are capable of detecting the presence of the opponent. The onboard actuator controller with encoder-feedback enables Junior to perform a specific maneuver with its two arms and two wheels (located right below its eyes). The communication infrastructure serves as a channel of information between Junior and an offboard controller. In other words, what Junior sees and feels will be communicated through such a channel back to the offboard controller for maneuver learning. At the same time, the maneuver selected by the offboard controller will also be sent to Junior.

A.2 The Opponents

In order to readily demonstrate as well as evaluate the effectiveness of maneuver learning, in this case study, we will use *dummy players* as opponents. The dummy players will have different sizes and shapes, which correspond to opponents holding different standing postures and different degrees of resistance (as if different weights) in an attempt to undermine the strength of the robot player.

Figure 2 presents the dummy players of different postures, *i.e.*, flat, curved, corner, and circular postures, respectively. More specifically, as shown in the figure, the vertical axis corresponds to the weight of an opponent, whereas the horizontal axis corresponds to three types of engagement contact, *i.e.*, surface, corner, and point contact. The weight difference of an opponent is considered

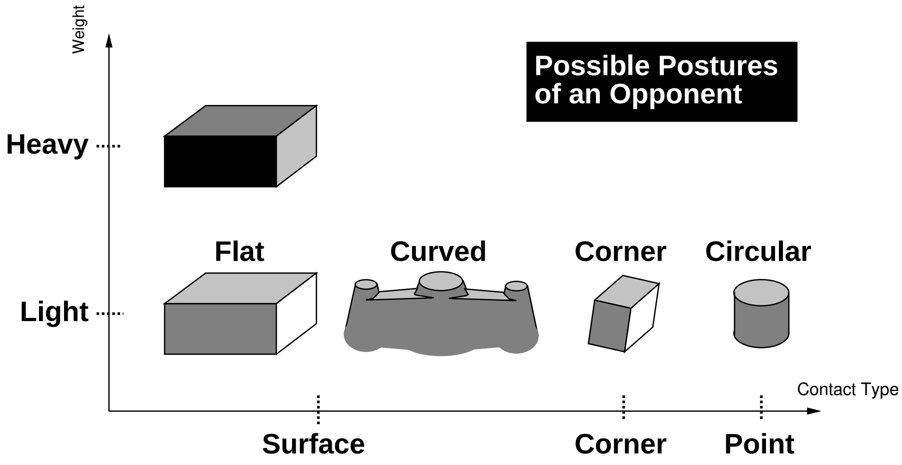

Fig. 2. The possible standing postures and degrees of resistance that an opponent may generate. They are simulated using dummy players of different engagement contact types and/or weights.

only in the case of a flat posture.

B. MPGP-Based Action Learning

In this section, we will provide the detailed MPGP formulation and algorithm for the task of sumo maneuver learning.

B.1 General vs. Specialized Actions

In our case study, we represent the actions of a sumo robot with two different granularities. The optimal maneuvers at a right granularity for a given opponent posture will be evolved using the MPGP approach.

First, we define a set of general actions, *i.e.*, forward move, backward move, left turn, right turn, and stop, that can readily be executed by Junior through controlling the angular displacements and directions of its wheels. These general actions are regarded as low-granularity actions. Next, we add some subtle movements into each general motion and thus obtain a set of high-granularity actions. The high-granularity actions are more specialized and refined than the low-granularity ones, and can be used by Junior to create specialized actions. The general and specialized actions

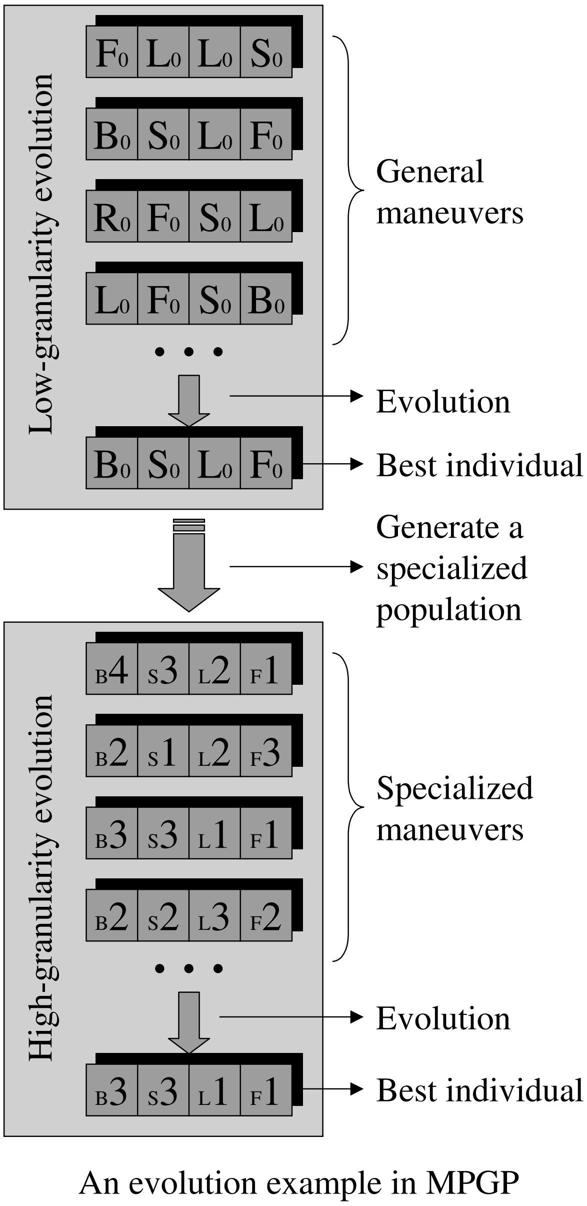

Fig. 3. An illustration of a two-phase MPGP. Here we assume that the length of an action sequence l is 4. The larger font letters or numbers in a chromosome indicates the general or specialized actions to be evolved.

TABLE I
THE SET OF GENERAL AND SPECIALIZED ACTIONS THAT CAN BE EXECUTED BY A SUMO ROBOT.

General Action (low-granularity)	Symbol in MPGP	Specialized Action (high-granularity)	Symbol in MPGP	Displacement	Turning Angle
F: forward move	F_0	FF: fast forward	F_1	$M_d + M_s$	0
		MF: moderate forward	F_2	M_d	0
		SF: slow forward	F_3	$M_d - M_s$	0
B: backward move	B_0	FB: fast backward	B_1	$-M_d - M_s$	0
		MB: moderate backward	B_2	$-M_d$	0
		SB: slow backward	B_3	$-M_d + M_s$	0
L: left turn	L_0	LSRF: l-wheel stall r-wheel forward	L_1	0	T_a
		LBRS: l-wheel backward r-wheel stall	L_2	0	T_a
		LBRF: l-wheel backward r-wheel forward	L_3	0	T_a
R: right turn	R_0	LFRS: l-wheel forward r-wheel stall	R_1	0	T_a
		LSRB: r-wheel backward l-wheel stall	R_2	0	T_a
		LFRB: l-wheel forward r-wheel backward	R_3	0	T_a
S: stop	S_0	S: stop	$S_{(1\ 2\ 3)}$	0	0

are summarized in Table I.

In Table I, M_d denotes the step size of a moderate forward action or a moderate backward action, whereas M_s denotes the discrepancy between a fast or slow action and a moderate action. T_a denotes the angular displacement of turning left or right. It should be pointed out that the turning center is not fixed in each situation. Specifically, when Junior executes the specialized actions of LSRF and LSRB, the turning center will be at its left wheel. When Junior executes the specialized actions of LFRS and LBRS, the turning center will be at its right wheel. And, when Junior executes the specialized actions of LFRB and LBRF, the turning center will be the robot's geometric center.

TABLE II
THE FUNCTION AND TERMINAL SETS AS USED IN THE MPGP CASE STUDY.

Function Set	Terminal Set
F	0
B	1
L	2
R	3
S	

B.2 Maneuvers and their Chromosome Representations

In the formulation of sumo maneuver learning, we represent a sequence of l actions as a single *maneuver*. Based on the preceding section, we can readily define the *function and terminal sets* to be used in the process of MPGP, as given in Table II.

Figure 3 presents an illustrative diagram showing a two-phase MPGP formulation. In the figure, the action types, F, B, L, R, and S, can be viewed as functions, whereas the action degrees, 0, 1, 2, and 3, can be viewed as terminals.

B.3 The Two-Phase Implementation of MPGP

In the MPGP formulation, we split the evolutionary process into two distinct phases[1]. In one *phase*, general maneuver learning is concerned only with general function evolution, *i.e.*, to select a sequence of general actions. In another *phase*, specialized maneuver learning deals with sub-function evolution, *i.e.*, to select a sequence of specialized actions. The objective of this phase is to derive fine-tuned maneuvers. The representations of chromosomes for general and specialized maneuvers are illustrated, respectively, in Figure 3.

[1] This is why this evolutionary approach is called multi-phase genetic programming (MPGP).

B.4 The Fitness Function

The evaluation of individuals in a population is a crucial step in both GP and MPGP. In MPGP, the fitness of a candidate chromosome corresponds to the performance of a maneuver specified in that chromosome. The performance is estimated according to the effectiveness of Junior with respect to its opponent in executing the maneuver. Specifically, the fitness function f of a chromosome C with l number of actions X_i is composed of several sumo performance requirements. Generally speaking, the fitness function measures how Junior positions itself and acts with respect to its opponent. A good maneuver means that Junior can constantly face and see the opponent with both-eye sensing and try to actively engage the opponent with its arms. The fulfillment of these requirements can be measured by using the following fitness function:

$$f = \sum_{i=1}^{l} w_{ai} \times \left(\sum_{j=1}^{n} \alpha^j(X_i) + \sum_{k=1}^{m} (\beta^k(X_i) + \gamma^k(X_i)) \right) \tag{1}$$

where $\alpha^j(X_i)$, $\beta^k(X_i)$, and $\gamma^k(X_i)$ will return 1, if Junior finds the opponent with its eye j, its arm k holds onto the opponent, and the arm is in contact with the opponent during action X_i, respectively. Otherwise, they will return 0. The measurements of $\alpha^j(X_i)$, $\beta^k(X_i)$, and $\gamma^k(X_i)$ can readily be determined by Junior via its sensors. In addition, l denotes the number of actions in a maneuver (which is set to 4 in our experiments). n denotes the number of eyes. m denotes the number of arms. w_{ai} denotes a positive weight for action X_i.

Using the above definition, the fitness of each candidate chromosome can be evaluated once its corresponding maneuver is performed.

In our present case study, Junior will not empirically test all the candidate maneuvers encoded in the population. Instead, we will utilize *a computational model of player-opponent interaction* to predicate the performance of such maneuvers. This computational model is aimed at capturing the effect of a maneuver similar to that of a physical contact pushing situation, and hence reduce

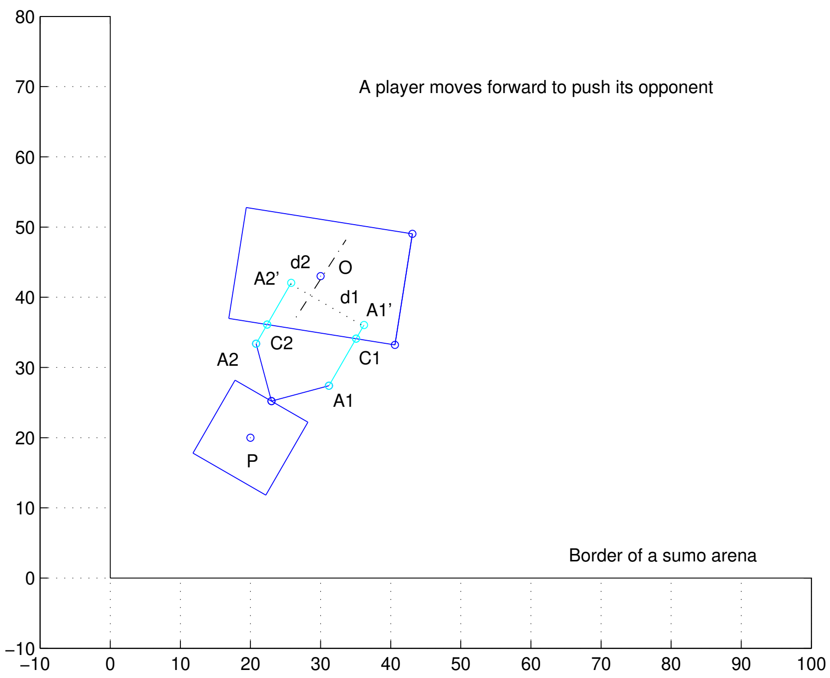

Fig. 4. The calculation of an artificial repulsive force as created by the arms of a moving player against its opponent. In the computational model, the shape of the opponent is rectangular. The player moves forward to push its opponent. In the figure, points P and O denote the centroids of the player and the opponent, respectively. A_1 and A_2 denote the right and left arms of the player, respectively. A_1' and A_2' denote the new locations of the player's arms after forward move, respectively. C_1 and C_2 correspond the intersections between the paths of two arms and the border of the opponent. d_1 and d_2 correspond to the orthogonal distances from the centroid of the opponent, O, to the directions of two arm forces. It is assumed in this illustration that the player moves forward; for other actions the force can be calculated in a slightly different fashion.

the time required for the evaluation. Details of the computational model based evaluation will be given in the following section.

B.5 A Computational Model of Interaction

In our computational model of interaction, we define the dynamics of player-opponent interaction by introducing a notion of *artificial repulsive force*. The model of artificial repulsive forces between a player and an opponent is based on Hook's law, *i.e.*, spring-like model, as follows: When the player executes an action against its opponent with one of its arms near the region of its opponent, it will exert an artificial repulsive force on the opponent. The direction of this force is the same as that of the arm' movement, and the magnitude is proportional to the depth in which

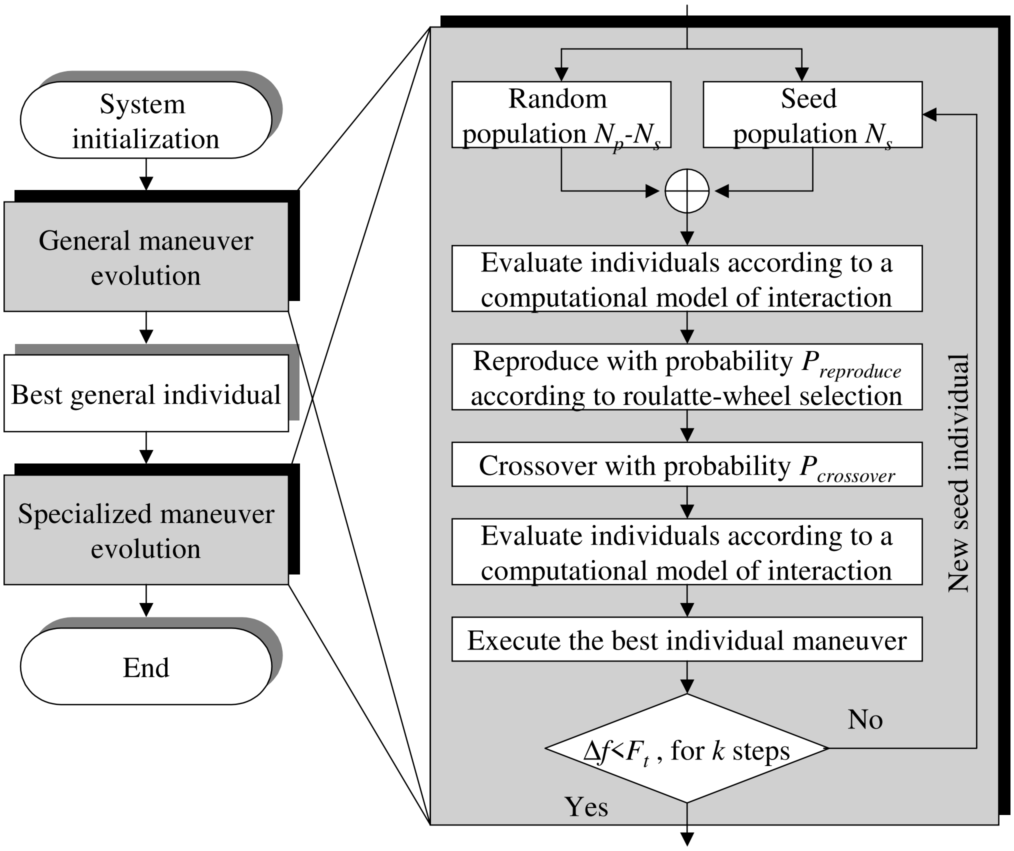

Fig. 5. An outline of the algorithm for the MPGP case study in sumo maneuver learning. Here the conditions are different in different phases. In the general maneuver evolution phase, if there is no large fitness improvement for k steps, the system will switch to a higher-granularity phase in order to acquire finer actions. While in the specialized maneuver learning, the condition will become whether or not the opponent is outside the arena or the number of generations exceeds a predefined threshold.

the arm moves into the original region of the opponent, as illustrated in Figure 4. Therefore, the force with which the player applies on its opponent is the net force from the player's two arms. Similarly, the net torque applied by the player on its opponent is also determined from the forces of the two arms.

Based on the illustration of Figure 4, we can calculate the force that a player exerts on its opponent, *i.e.*, the magnitude and direction of the net force as follows:

$$
\begin{aligned}
&\overrightarrow{F}_1 = C_f \cdot \overrightarrow{C_1 A_1}; \quad \overrightarrow{F}_2 = C_f \cdot \overrightarrow{C_2 A_2}; \quad \overrightarrow{F} = \overrightarrow{F}_1 + \overrightarrow{F}_2; \\
&J_1 = d_1 \cdot |C_1 A_1|; \quad J_2 = -d_2 \cdot |C_2 A_2|; \quad J = J_1 + J_2
\end{aligned} \tag{2}
$$

where C_f is a positive coefficient. F_1 and F_2 denote the forces exerted by two arms on the opponent, respectively. F denotes the net force. J_1 and J_2 denote the torques applied to the opponent. J is the net torque.

From the net force and torque that an opponent receives, its resulting linear and angular displacements can therefore be predicted as follows:

$$
\begin{aligned}
&\overrightarrow{L} = C_d \cdot \overrightarrow{F}; \\
&\theta = C_\theta \cdot J \cdot
\end{aligned} \tag{3}
$$

where C_d and C_θ denote two positive coefficients.

With the above definitions and assumptions, we can efficiently *estimate* the fitness of each chromosome.

C. The Evolutionary Algorithm for MPGP-Based Action Learning

During the process of MPGP-based sumo maneuver learning, the population at each evolutionary step consists of two parts: the *seed population* and the randomly produced population. The best candidate maneuver will be selected by an offboard controller and sent to Junior for execution. After Junior executes the selected sumo maneuver, the actual performance of such a maneuver will be evaluated by Junior using its sensors. This feedback will then be communicated to the controller. Thereafter, the controller will keep this maneuver along with its fitness in the seed population for further maneuver learning.

The seed population will be updated according to the fitness values of maneuvers and the life span of each evolution step. That is, both the lowest-fitness and the oldest maneuver will be eliminated. The rule for updating the seed population of size N_s can be stated as follows:

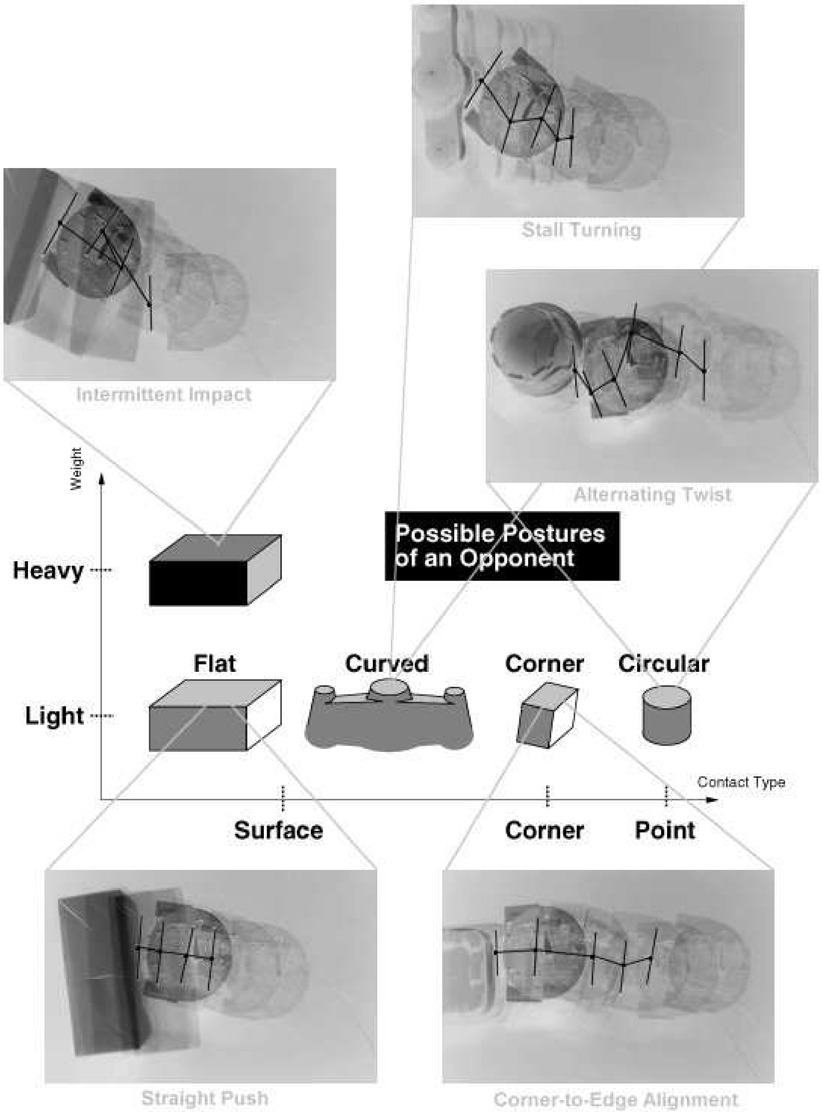

Fig. 6. Traced *trajectories* for the five specialized maneuvers evolved in the presence of five different opponent postures, respectively. In each of the five overlaid maneuver pictures, the equal-length line segments show the front of Junior at different action steps while performing a specialized maneuver. The captured maneuvers are: (i) *straight push* for lightweight flat postures, (ii) *intermittent impact* for heavyweight flat postures, (iii) *stall turning* for lightweight curved postures, (iv) *corner-to-edge alignment* for lightweight corner postures, and (v) *alternating twist* for lightweight circular postures.

$$i \Rightarrow \text{eliminated} \qquad \text{if } f_s^i = \min(f_s^j | f_s^j = y_s^j \cdot f^j,\ \forall j \in N_s) \tag{4}$$

where y_s^j denotes an aging factor of individual j in the seed population.

Figure 5 presents an outline of the algorithm (*i.e.*, the reproduction, crossover, and selection of a population) as used in our MPGP case study.

IV. Experiments and Discussions

In our experiments, we use five dummy objects, simulating five different types of opponent postures and/or weights. As illustrated in Figure 2, the five dummies used represent the opponents with (i) lightweight flat posture, (ii) heavyweight flat posture, (iii) lightweight curved posture, (iv) lightweight corner posture, and (v) lightweight circular posture, respectively.

Figure 6 presents the performance of a sumo learning robot. Specifically, it records the evolved specialized maneuvers of the robot, overlaid on the schematic diagram of five opponent types as given in Figure 2. Each of the overlaid pictures in Figure 6 shows a sequence of actions in a respective maneuver actually performed, in which the equal-length line segments (some are overlapped) indicate the position and orientation of the robot front at consecutive action steps, and the line that connects the centers of those orientation line segments indicates the actual trajectory of the robot in performing a specialized maneuver.

A. *The Fitness Function Revisited*

Let us now take a close look at the changes in fitness values during the course of the MPGP-based maneuver evolution. Specifically, here by *fitness values* (or fitness curves) we mean only those for the early defined fitness function, as it directly concerns the performance of specialized maneuver learning.

We will examine one set of fitness values for each of the five specialized maneuvers. Our objectives are two-fold: (1) to study the effectiveness of the MPGP-based maneuver evolution as well as the difficulty of the problem with respect to a specific posture, and (2) to study the role of sensory feedback (*i.e.*, the infra-red sensors) in the performance of evolution − the question here is whether or not the sensory feedback requirement has anything to do with the uncertainty involved in the problem.

In order to achieve the above objectives, we will examine *three fitness curves for each of the MPGP-based maneuvers*, namely,

1. fitness curve recorded (*i.e.*, the *dotted-circle line* in Figures 7-11) when involving only left infra-red sensor (*i.e.*, *left-eye view or sensing*).
2. fitness curve recorded (*i.e.*, the *dashed-star line* in Figures 7-11) when involving only right infra-red sensor (*i.e.*, *right-eye view or sensing*).
3. fitness curve recorded (*i.e.*, the *solid line* in Figures 7-11) when involving both infra-red sensors (*i.e.*, *both-eye view or sensing*).

Also, it should be mentioned that *the fitness curves being examined here correspond to the measured values for the best individuals from single generations.*

A.1 Lightweight or Heavyweight Flat Posture

As shown in Figures 7 and 8, in the cases of lightweight or heavyweight flat posture opponents, we note that f is very sensitive at the beginning and stabilized after about 15-20 generations. Also, the dotted and dashed fitness lines are shorter than the solid line. In other words, the MPGP-based learning enables Junior to *quickly* find a good way to face to its opponent with a *both-eye view*. Thereafter, it concentrates on the evolution of specialized maneuvers, *straight push* or *intermittent impact*, involving *both arms*.

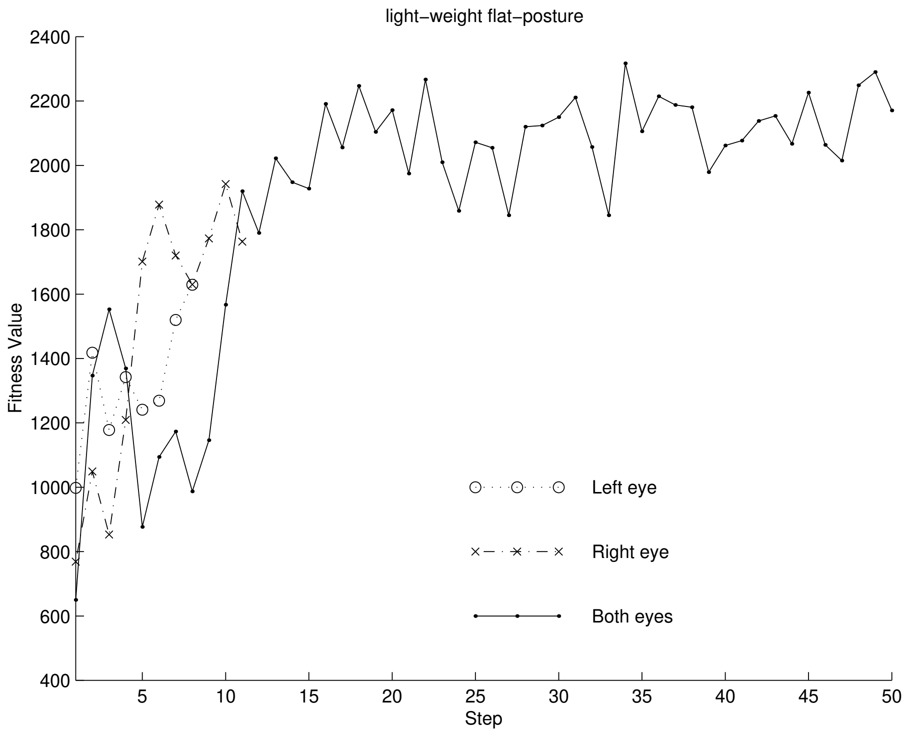

Fig. 7. The changes in the fitness function over 50 generations of the MPGP evolution to deal with *lightweight flat posture opponents.* The *dotted-circle line*, *dashed-star line*, and *solid line* show the values recorded under three conditions of maneuver learning, *left-eye view only*, *right-eye view only*, and *both-eye view*, respectively.

Such performance is in fact quite consistent with the merit requirement as represented by f. As we know in these cases, both-eye sensing and both-arm pushing are not only necessary but also convenient, since the flat posture creates very little uncertainty in terms of sensing and movement.

The lightweight and heavyweight maneuver learning cases are similar, as far as switching from a single-eye view to a both-eye view is concerned. Their key difference is that the maneuvers with a single-eye view in the latter case are not as effective as those in the former case, as we can observe from their fitness curves. In addition, the specialized maneuvers in the latter case utilize a sequence of fine-tuned actions for *high-impact intermittent push.* In other words, the controller using the MPGP approach can find a niche maneuver of intermittent impact for Junior to score high-fitness values when facing a heavyweight opponent.

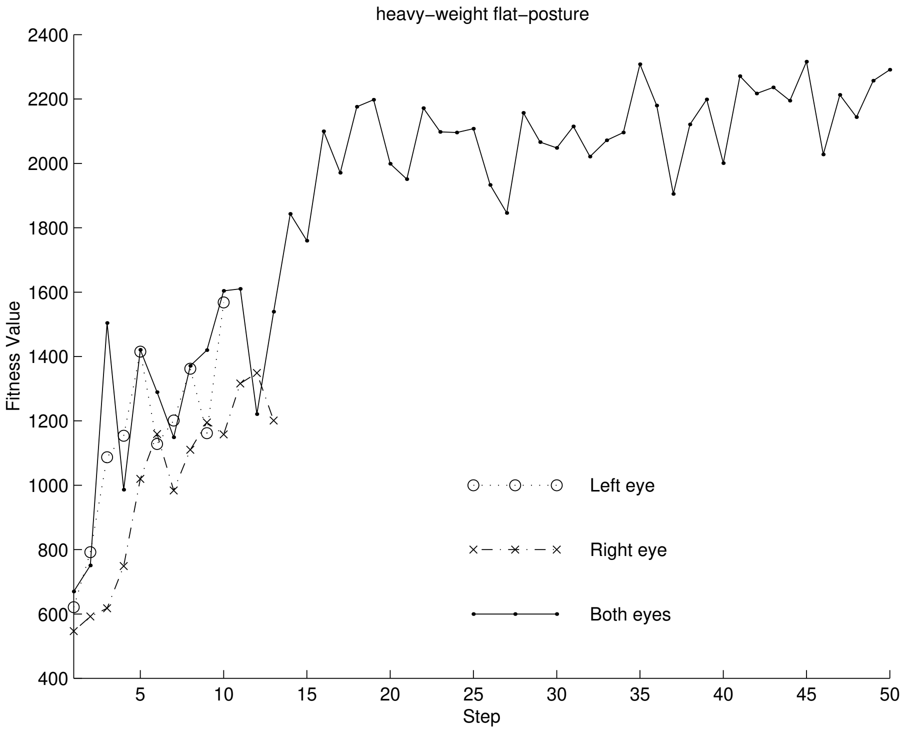

Fig. 8. The changes in the fitness function over 50 generations of the MPGP evolution to deal with *heavyweight flat posture opponents.* The *dotted-circle line*, *dashed-star line*, and *solid line* show the values recorded under three conditions of maneuver learning, *left-eye view only*, *right-eye view only*, and *both-eye view*, respectively.

A.2 Lightweight Curved Posture

In the case of a lightweight curved posture opponent, the f values show a ruggedly increasing trend, in all three sensing modes (*i.e.*, right-eye view only, left-eye view only, and both-eye view).

The lines of Figure 9 differ from those in the preceding two cases. First, the solid line for the both-eye sensing mode increases not as fast as those in the previous cases. This is primarily because the curved posture is relatively *harder* to keep a both-eye view and hence slightly *more difficult* to improve the f values. Secondly, the dotted-circle line for the left-eye sensing mode advances longer and higher than the flat posture cases. This indicates that (1) such a sensing mode can be present for some generations before an effective specialized maneuver is found, and (2) with the feedback information coming from only the left eye, it is still feasible to evolve a specialized maneuver to improve the f values. Similarly, we can also make such observations in the case of

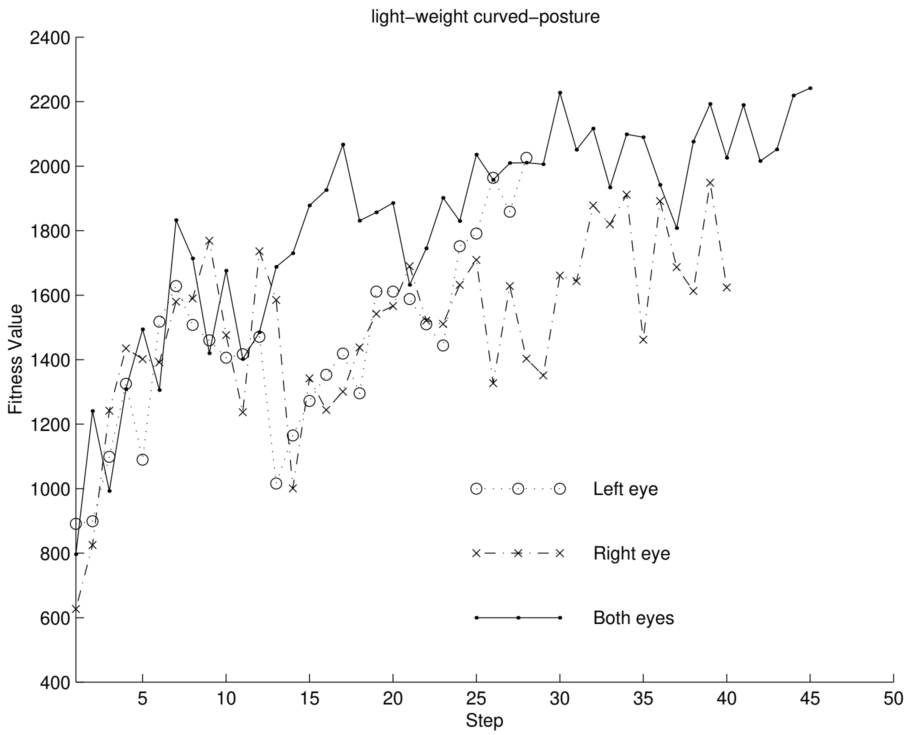

Fig. 9. The changes in the fitness function over 50 generations of the MPGP evolution to deal with *lightweight curved posture opponents.* The *dotted-circle line*, *dashed-star line*, and *solid line* show the values recorded under three conditions of maneuver learning, *left-eye view only*, *right-eye view only*, and *both-eye view*, respectively.

right-eye sensing (*i.e.*, the dashed-star line).

As can readily be noted from the fitness curves in the preceding three cases, a phase transition occurred at about generation 10-13, indicating that the MPGP-based evolution successfully selected a high-fitness general maneuver and then moved onto the next phase of specialized maneuver learning.

A.3 Lightweight Corner Posture

Figure 10 shows the f values in the case of a lightweight corner posture opponent. As can be noted, the both-eye view of the opponent becomes impossible in this case. That is why the solid line is short.

Due to the uncertainty in the environment, *i.e.*, the difficulty in predicting the direction of an

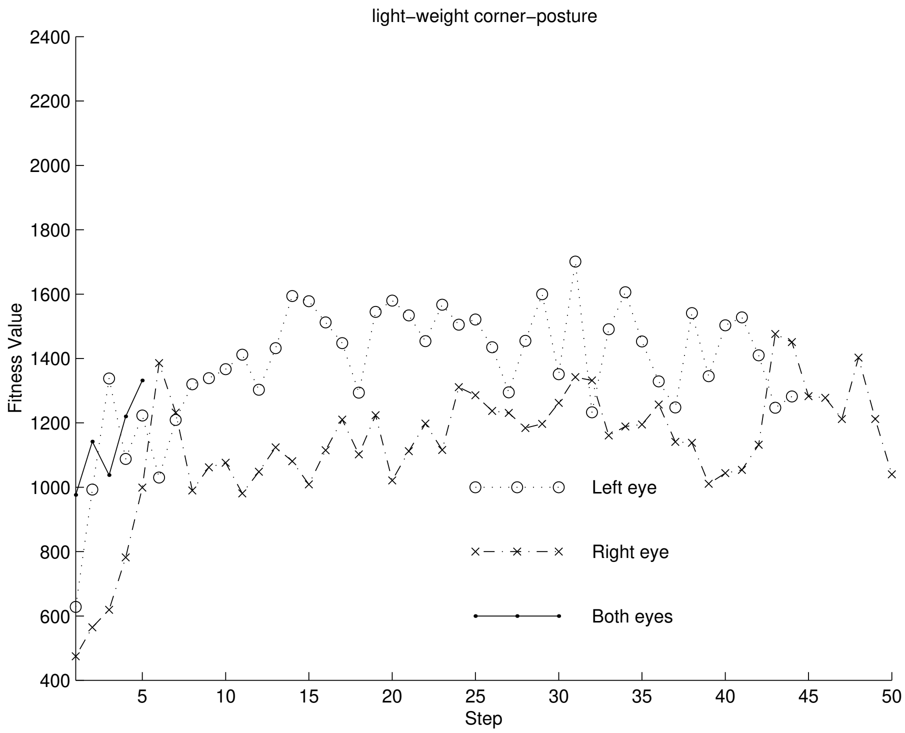

Fig. 10. The changes in the fitness function over 50 generations of the MPGP evolution to deal with *lightweight corner posture opponents.* The *dotted-circle line*, *dashed-star line*, and *solid line* show the values recorded under three conditions of maneuver learning, *left-eye view only*, *right-eye view only*, and *both-eye view*, respectively.

opponent's movement, the evolved maneuvers with a single-eye view are not as effective as those in the preceding three cases. This can also be observed from their f values. In the present case, the fitness function converges to a certain level, revealing that the performance of Junior is still achievable with the evolved general maneuvers.

A.4 Lightweight Point Posture

Figure 11 gives the f values corresponding to the MPGP-based learning of specialized *alternating twist* maneuvers, in the case of a lightweight point posture opponent. It can be noted that the evolved maneuvers are slightly more effective than those in the previous case. This is because the movement uncertainty in the point posture case is less, as it does not involve any engagement contact transitions.

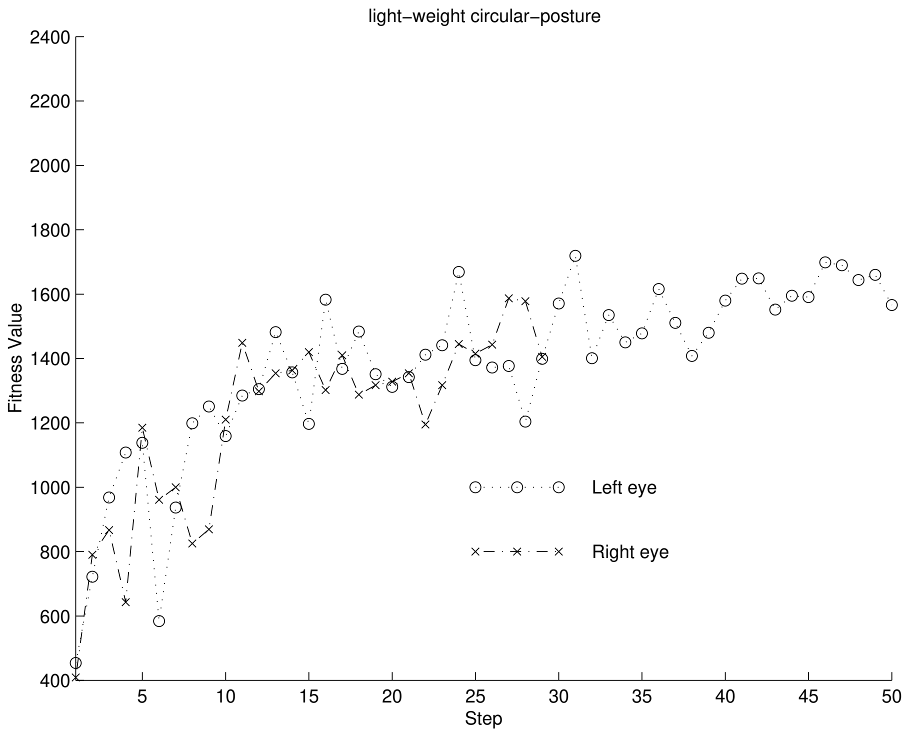

Fig. 11. The changes in the fitness function over 50 generations of the MPGP evolution to deal with *lightweight point posture opponents.* The *dotted-circle line*, *dashed-star line*, and *solid line* show the values recorded under three conditions of maneuver learning, *left-eye view only*, *right-eye view only*, and *both-eye view*, respectively.

Generally speaking, we have found that:

1. f is less sensitive to the above corner and point postures.
2. The role of sensory feedback in the maneuver learning is largely constrained by the uncertainty involved in the problem on hand.

B. Comparison between the MPGP and GP Approaches

In order to compare the maneuver learning performance of the MPGP and GP approaches, we conducted two sets of experiments, one for each approach. Each set of experiments consisted of 30 learning runs, started at the same positions and orientations and terminated once Junior successfully pushed a prototype opponent out of the sumo arena. We recorded the average number of generations for the on-line sumo learning and performance.

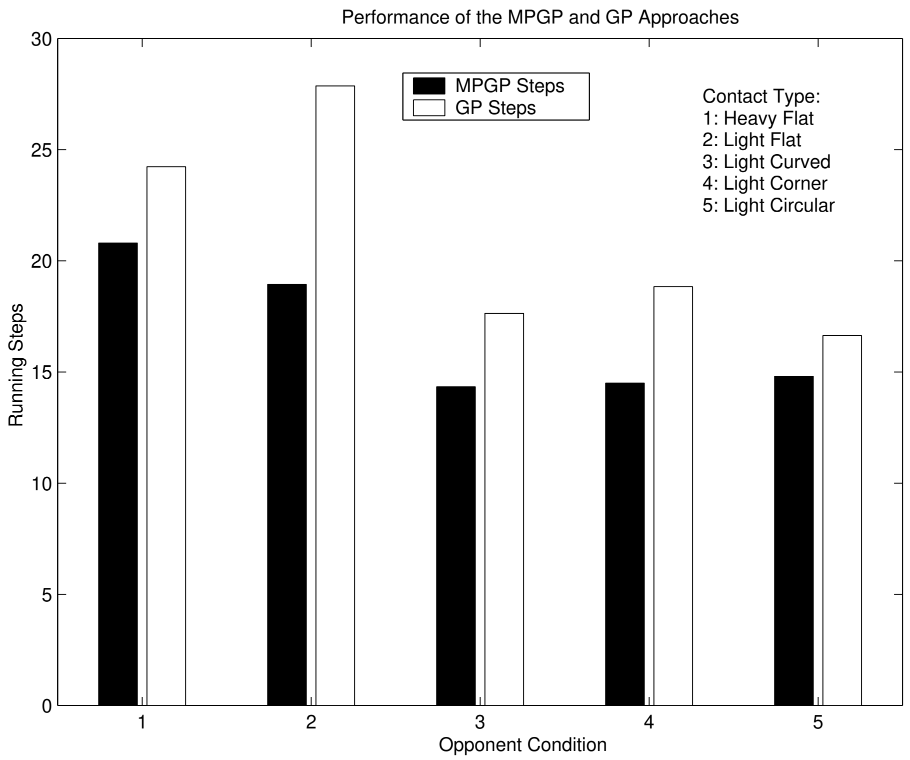

Fig. 12. The average numbers of steps in learning and performing sumo maneuvers using the MPGP and GP algorithms, respectively, in the presence of five different opponent postures. For each case, 30 experiments were carried out.

Figure 12 shows the experimental results of our comparative studies. From the figure, we note that *good-performance maneuvers can more readily be evolved with MPGP than with GP*. We should point out that the parameters for the experiments, including the initial locations of a sumo robot and an opponent as well as their orientations, were all kept the same.

Figure 13 presents the conventional GP algorithm as used in this study.

C. Experimental Settings

Table III summarizes the parameters as used in the above-mentioned MPGP and GP experiments. Also listed in Table III are the geometric dimensions of a sumo arena and a sumo robot.

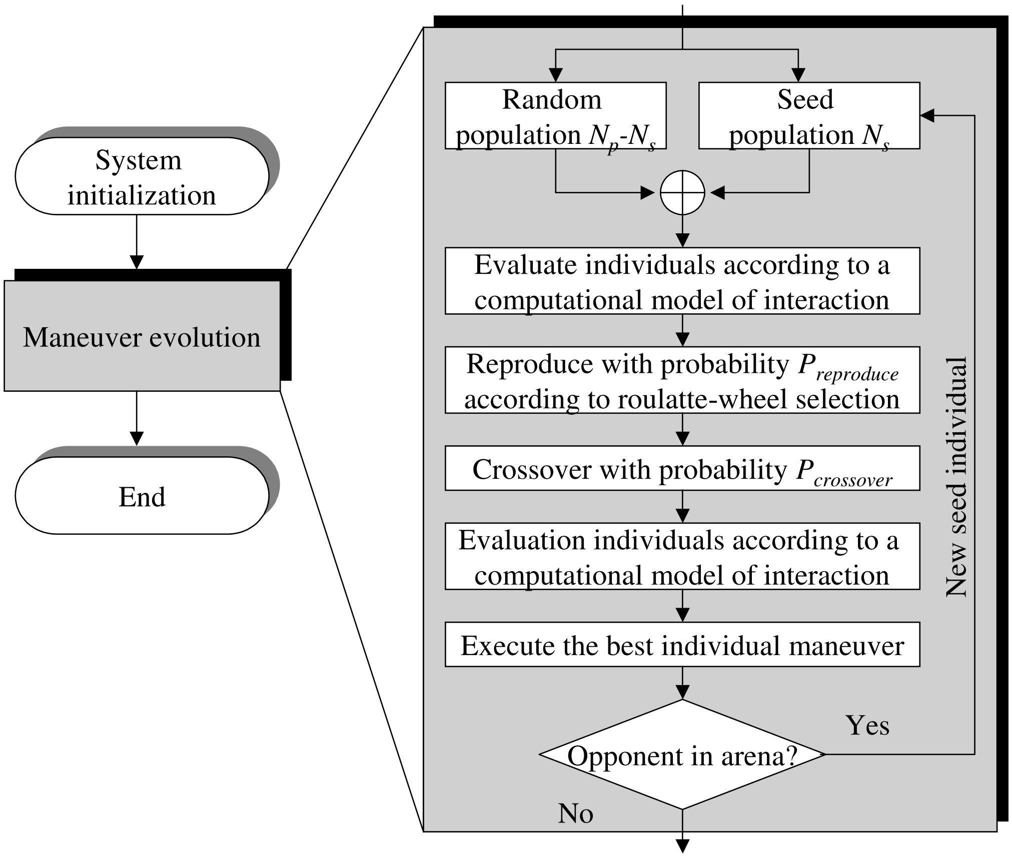

Fig. 13. An outline of the GP algorithm as used in the case study.

TABLE III

THE PARAMETERS AS SET IN THE MPGP-BASED AND GP-BASED SUMO MANEUVER LEARNING.

M	650mm	n	2
N	420mm	l	4
PM	60mm	N_p	8
PN	60mm	N_s	4
C_θ	0.001	$P_{reproduce}$	0.25
C_f	1	$P_{crossover}$	0.75
C_d	1	w_{a1}	1
M_d	40mm	w_{a2}	1
T_a	$\pi/12$	w_{a3}	1
M_s	20mm	k	3
m	2	F_t	200

V. COMPARISON WITH RELATED WORK

In this section, we will compare our MPGP formulation to those of related work on evolutionary robotics and evolutionary algorithms, and discuss the distinct features and advantages of different approaches.

A. *Comparison with Nordin* et al.*'s Work*

Nordin *et al.* developed an evolutionary robotic system that utilized genetic programming to evolve machine codes and to control a miniature robot, *Khepera*, on-line [16]. Their system differs from our MPGP approach in the following ways:

1. In Nordin *et al.*'s work, variable-length machine codes were designed and operated genetically. The advantage of this approach is that the system does not need to translate a conventional program into machine codes, which in turn improves the efficiency of the system. Nordin *et al.* named their genetic programming approach Compiling Genetic Programming System (CGPS).

   In our approach, we divide the actions of a robot into actions of different granularities. Each individual in MPGP, *i.e.*, an action sequence, is composed of general or specialized actions, and is evolved in different phases according to their granularities. Our approach directly evolves, and hence puts more emphasis on, the behavior of an agent. The representation of an individual in this case is straightforward to implement without too much human intervention. In addition, from the results that we have obtained, emergent behavior, *i.e.*, strategic maneuvers, can readily be noted.

2. Nordin *et al.* adopted a Steady-State Genetic Programming (SSGP) to perform genetic operations. In SSGP, each of the selected individuals was evaluated on a physical robot, *Khepera*. This might cause some 'unfair' evaluations in which good individuals may get

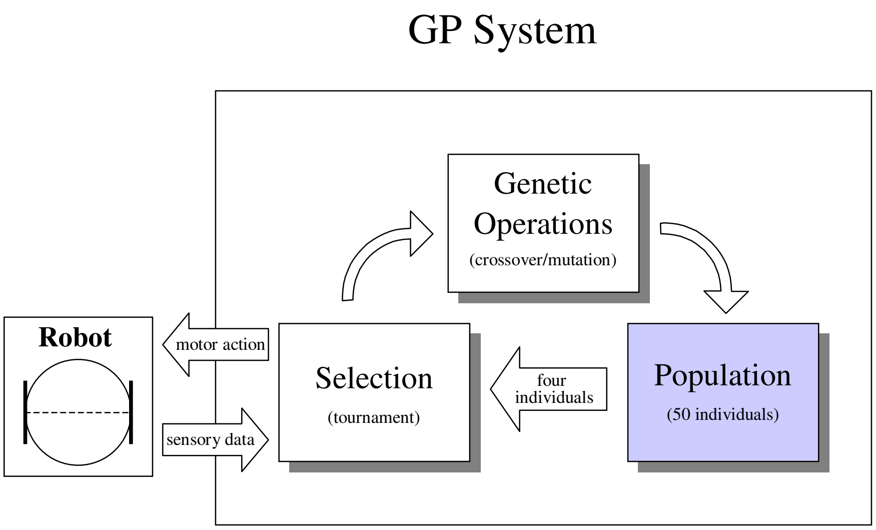

Fig. 14. A schematic diagram of Nordin *et al.*'s on-line GP design.

poor fitness because of the different initial environments that they encountered.

In Nordin *et al.*'s system, all individuals were evaluated on a physical robot, whereas in our approach, only the best individual selected from a computational model at each step is validated on a physical robot. Thus, our system maintains a good-efficiency evolution, as we do not validate every individual on a physical robot.

3. Our system keeps a repository of selected maneuvers. This repository is updated according to the fitness values as well as the ages of the individuals. Thus, our system incorporates a learning capability into the evolutionary process.
4. Last but not the least, the architectures of the two approaches are different, as illustrated in Figures 14 and 15, respectively.

B. Comparison with Floreano et al.*'s Work*

Floreano, Urzelai, and Mondada studied an evolutionary approach to the on-line self-organizing control of mobile robotic systems [13], [25], [26]. In their approach, they utilized a

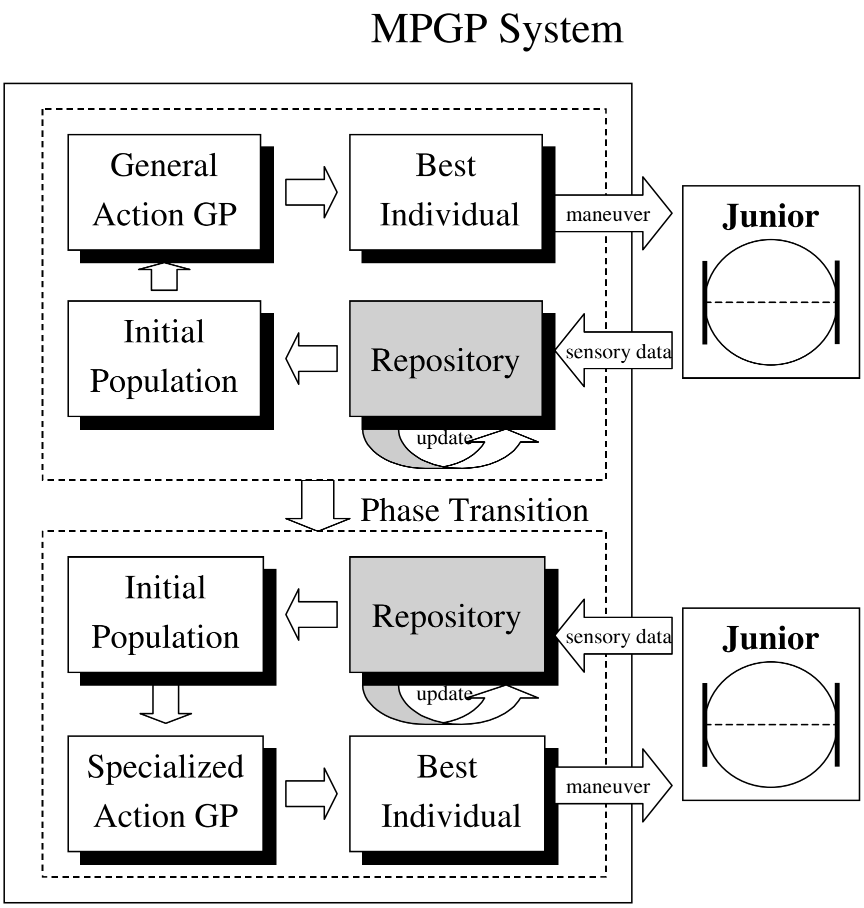

Fig. 15. A schematic diagram of the on-line MPGP design.

discrete-time neural network to evolve and control a physical robot. The differences between their approach and ours can be summarized as follows:

1. They adopted a genetic algorithm to evolve the weights, or rules for defining the weights, of neural networks. This approach is commonly used in evolutionary robotics. The input to neural networks is connected to a robot's sensors, whereas the output unit is used to control a robot's wheels. In our approach, we do not predefine a neuro-controller system, instead the evolved maneuvers are sent directly to control a robot's behavior.
2. Floreano *et al.*'s work adopted an evolutionary process similar to that of Nordin's, where all individuals were evaluated on a physical robot. Although the environmental conditions for

the robot may be different, adaptive behaviors, such as navigation and obstacle avoidance, can be acquired after a number of generations.

In our approach, the individuals of the same generation share the same simulated environment. Only will the best individual selected from the simulated environment be validated on a physical robot, and used to update the behavioral repository of the robot, from which the population, *i.e.*, potential maneuvers, is in turn selected. Thus, our approach can be relatively less time-consuming than Floreano *et al.*'s, *as it does not physically validate all individuals but only the best one from the computational model based evaluations.*

C. Comparison with Harvey's SAGA

Another related approach is Harvey's Species Adaptation Genetic Algorithm (SAGA) [27], [28], [29]. The objective of SAGA was to simulate a long-term evolutionary process in adaptive species. In order to achieve this objective, two issues were addressed: (1) changing the length of a genotype during the evolutionary process, and (2) converging a population while keeping the genotype length constant. The SAGA approach adjusts mutation rates in order to make the evolutionary process converged and to escape from local optima.

In our approach, the populations are also grouped into different 'species', according to different action granularities. The advantage of our approach is that it is relatively *easier to implement*, as it does not require an explicit mechanism for genotype length or mutation rate adaptation.

VI. Concluding Remarks

In this paper, we have demonstrated a multi-phase genetic programming (MPGP) approach to a sumo learning task. From the obtained results of comparison with the conventional GP approach, it was shown that the MPGP approach is relatively more efficient, *as it focuses on general (or grouped) solutions (*e.g.*, general maneuvers) first and then moves onto specialized ones (*e.g.*,* *specialized maneuvers) based on the obtained high-fitness general solutions (*e.g.*, inheriting general sumo expertise).*

REFERENCES

[1] A. Murray and S. J. Louis, "Design strategies for evolutionary robotics," in *Intelligent Systems: Third Golden West International Conference*, pp. 609–616. Kluwer Academic Publishers, 1995.

[2] P. Husbands, "The artificial evolution of robot control systems," in *Adaptive Computing in Engineering Design and Control '96 (ACEDC'96), 2nd International Conference of the Integration of Genetic Algorithms and Neural Network Computing and Related Adaptive Techniques with Current Engineering Practice*, I. Parmee and M. J. Denham, Eds. March 26-28, 1996.

[3] M. Mataric and D. Cliff, "Challenges in evolving controllers for physical robots," *Journal of Robotics and Autonomous Systems*, vol. 19, no. 1, pp. 67–83, 1996.

[4] I. Harvey, P. Husbands, and D. Cliff, "Issues in evolutionary robotics," in *From Animals to Animats 3: Proceedings of the Third International Conference on Simulation of Adaptive Behavior (SAB-94)*, pp. 364–373. MIT Press, Cambridge, MA, 1994.

[5] I. Harvey, "Is there another new factor in evolution?" *Evolutionary Computation*, vol. 4, no. 3, pp. 313–329, 1996.

[6] S. Nolfi and D. Floreano, *Evolutionary Robotics: Biology, Intelligence, and Technology of Self-Organizing Machines*, MIT Press, Cambridge, MA, 2000.

[7] O. Miglino, H. H. Lund, and S. Nolfi, "Evolving mobile robots in simulated and real environments," *Artificial Life*, vol. 2, no. 4, pp. 417–434, 1995.

[8] S. Nolfi and D. Parisi, "Evolving non-trivial behaviors on real robots: an autonomous robot that picks up objects," in *Proceedings of Fourth Congress of Italian Association of Artificial Intelligence*, pp. 243–254. Springer Verlag, 1995.

[9] S. Nolfi, D. Floreano, O. Miglino, and F. Mondada, "How to evolve autonomous robots: Different approaches in evolutionary robotics," in *Artificial Life IV: Proceedings of the Fourth International Workshop on the Synthesis and Simulation of Living Systems*, pp. 190–198. MIT Press, Cambridge, MA, 1994.

[10] D. Floreano and F. Mondada, "Automatic creation of an autonomous agent: Genetic evolution of a neural-network driven robot," in *From Animals to Animats 3: Proceedings of the Third International Conference on Simulation of Adaptive Behavior (SAB-94)*, pp. 421–430. MIT Press, Cambridge, MA, 1994.

[11] F. Gomez and R. Miikkulainen, "Incremental evolution of complex general behavior," *Adaptive Behavior*, vol. 5, no. 3-4, pp. 317–342, 1997.

[12] D. Floreano and J. Urzelai, "Artificial evolution of adaptive software: An application to autonomous robots," *The Journal of Three Dimensional Images*, vol. 14, no. 4, pp. 64–69, 2000.

[13] D. Floreano and J. Urzelai, "Evolutionary robots with on-line self-organization and behavioral fitness," *Neural Networks*, vol. 13, pp. 431–443, 2000.

[14] C. Reynolds, "An evolved, vision-based model of obstacle avoidance behavior," in *Artificial Life III, Santa Fe Institute Studies in the Sciences of Complexity*, C. Langton, Ed., vol. XVI, pp. 327–346. 1993.

[15] C. Reynolds, "Evolution of corridor following in a noisy world," in *From Animals to Animats 3: Proceedings of the Third International Conference on Simulation of Adaptive Behavior (SAB-94)*, pp. 402–410. MIT Press, Cambridge, MA, 1994.

[16] P. Nordin and W. Banzhaf, "An on-line method to evolve behavior and to control a miniature robot in real time with genetic programming," *Adaptive Behavior*, vol. 5, no. 2, pp. 107–140, 1997.

[17] M. C. Martin, "Visual obstacle avoidance using genetic programming: First results," in *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2001)*, pp. 1107–1113. 2001.

[18] G. Hornby, H. Lipson, and J. Pollack, "Evolution of generative design systems for modular physical robots," in *International Conference on Robotics and Automation*, pp. 4146–4151. 2001.

[19] G. Hornby and J. Pollack, "The advantages of generative grammatical encodings for physical design," in *Congress on Evolutionary Computation (CEC)*, pp. 600–607. 2001.

[20] M. Colombetti and M. Dorigo, "Learning to control an autonomous robot by distributed genetic algorithms," in *From Animals to Animats 2: Proceedings of the Second International Conference on Simulation of Adaptive Behavior (SAB-93)*, pp. 305–312. MIT Press, Cambridge, MA, 1993.

[21] L. Steels, "Emergent functionality in robotic agents through on-line evolution," in *Artificial Life IV: Proceedings of the Fourth International Workshop on the Synthesis and Simulation of Living Systems*, pp. 8–16. MIT Press, Cambridge, MA, 1994.

[22] J. Xiao, Z. Michalewicz, L. Zhang, and K. Trojanowski, "Adaptive evolutionary planner/navigator for mobile robots," *IEEE Transaction on Evolutionary Computation*, vol. 1, no. 1, pp. 18–28, 1997.

[23] P. Nordin and W. Banzhaf, "Genetic programming controlling a miniature robot," in *Working Notes for the AAAI-95 Fall Symposium on Genetic Programming*, E. V. Siegel and J. R. Koza, Eds., Cambridge, MA, 1995, pp. 61–67.

[24] J. R. Koza, *Genetic Programming: On the Programming of Computers by Means of Natural Selection*, MIT Press, Cambridge, MA, 1992.

[25] D. Floreano and F. Mondada, "Evolution of homing navigation in a real mobile robot," *IEEE*

Transactions on Systems, Man, and Cybernetics-Part B, vol. 26, no. 3, pp. 396–407, June 1996.

[26] D. Floreano and F. Mondada, “Evolutionary neurocontrollers for autonomous mobile robots,” *Neural Networks*, vol. 11, no. 7-8, pp. 1461–1478, October 1998.

[27] I. Harvey, “Species adaptation genetic algorithms: A basis for a continuing SAGA,” in *Toward a Practice of Autonomous Systems: Proceedings of the First European Conference on Artificial Life*, pp. 346–354. MIT Press, Cambridge, MA, 1992.

[28] I. Harvey, “Evolutionary robotics and SAGA: the case for hill crawing and tournament selection,” *Artificial Life III, Santa Fe Institute Studies in the Sciences of Complexity*, vol. XVI, pp. 299–326, 1993.

[29] I. Harvey, “Artificial evolution: A continuing SAGA,” in *Evolutionary Robotics: From Intelligent Robots to Artificial Life, ER2001*, T. Gomi, Ed., vol. 2217, pp. 94–109. Springer-Verlag, 2001.