Int. J. Appl. Math. Comput. Sci., 2010, Vol. 20, No. 1, 69–84
DOI: 10.2478/v10006-010-0005-7

A BIOLOGICALLY INSPIRED APPROACH TO FEASIBLE GAIT LEARNING FOR A HEXAPOD ROBOT

DOMINIK BELTER, PIOTR SKRZYPCZYŃSKI

Institute of Control and Information Engineering
Poznań University of Technology, ul. Piotrowo 3A, 60–965, Poznań, Poland
e-mail: {dominik.belter,piotr.skrzypczynski}@put.poznan.pl

The objective of this paper is to develop feasible gait patterns that could be used to control a real hexapod walking robot. These gaits should enable the fastest movement that is possible with the given robot's mechanics and drives on a flat terrain. Biological inspirations are commonly used in the design of walking robots and their control algorithms. However, legged robots differ significantly from their biological counterparts. Hence we believe that gait patterns should be learned using the robot or its simulation model rather than copied from insect behaviour. However, as we have found *tahula rasa* learning ineffective in this case due to the large and complicated search space, we adopt a different strategy: in a series of simulations we show how a progressive reduction of the permissible search space for the leg movements leads to the evolution of effective gait patterns. This strategy enables the evolutionary algorithm to discover proper leg co-ordination rules for a hexapod robot, using only simple dependencies between the states of the legs and a simple fitness function. The dependencies used are inspired by typical insect behaviour, although we show that all the introduced rules emerge also naturally in the evolved gait patterns. Finally, the gaits evolved in simulations are shown to be effective in experiments on a real walking robot.

Keywords: evolutionary learning, legged robots, gait generation, model identification, reality gap.

1. Introduction

The generation of gaits—the patterns of moving legs in walking or running—is a fundamental problem for every walking robot. For multi-legged robots, developing walking patterns is a challenging task, because there are a large number of degrees of freedom and therefore the solution must be found in a large, multidimensional search/state space, where useful states are sparsely distributed.

The issues of gait generation have been studied for many years, and there are various approaches to this problem known from the literature (Ridderström, 1999). Although the generation of adaptive gait patterns has already been demonstrated in simulated (Kumar and Waldron, 1989; Yang, 2009) and real (Fukuoka *et al*., 2003; Parker and Mills, 1999) environments, many walking robot controllers use gaits programmed by hand. In most cases (particularly for six- and eight-legged walking machines) these gaits are statically stable, i.e., the gait longitudinal stability margin is positive (Song and Waldron, 1989).

Typically, the synthesis of simple, static gaits is based on a kinematic model of the robot, and the designer's inspiration is taken from biology (Figliolini *et al*., 2007). Studies on insect walking are an obvious source of gait templates for hexapod robots (Beer *et al*., 1997). This is the easiest method to obtain feasible movement of a walking robot. In spite of its implementational simplicity, this method does not guarantee the best use of

Fig. 1. Hexapod robot Ragno.

the robot kinematic and dynamic capabilities. When leg co-ordination patterns known from nature are used, some other gaits are excluded by design, only because they have not been observed in living creatures. Despite the fact that these walking patterns are not used in nature, it is worth verifying whether they are applicable to walking machines. Even though insects are simple animals, they are much more complex than the existing walking robots. An example may be leg specialization found in both insects and vertebrates, which is largely ignored in the design of walking robots due to technological and control issues (Ritzmann *et al*., 2004). Thus, as it seems to be impossible to copy even the simplest insect into a robotic design, also movement strategies, including gaits, may differ. We believe that gaits optimized or learned using an actual robot can perform better than their counterparts directly modelled on biological patterns due to their adjustment to the specific robot hardware.

To find gait patterns for a walking robot from scratch, the multidimensional space of all possible leg states should be examined. The hexapod robot Ragno (Fig. 1) used in this research has 18 degrees of freedom, so in a general case 18 reference values for all the active joints have to be determined in every control cycle. A variety of methods exist for solving such a problem, but an evolutionary algorithm seems to be appropriate to perform this difficult task, due to its ability to provide an approach to the global solution better than gradient-based analytical algorithms, its good explorative properties, and insensitivity to the initial parameters.

However, even with an evolutionary technique learning one big task by the robot at once may be hard. A reasonable solution to this problem is the supervised learning procedure: a human supervisor defines the whole control framework and evaluates the progress of the learning robot. Such an approach is often called 'shaping' (Dorigo and Colombetti, 1997). This paradigm was used successfully in our previous research on GA-based learning of fuzzy reactive behaviour for a wheeled robot (Skrzypczyński, 2004b).

This work approaches the problem of learning gait patterns for a hexapod robot using shaping-type learning, by means of an evolutionary algorithm. Our main idea is to let the system learn a gait starting from as minimal *a priori* knowledge as possible. The leg movement sequences (defined by the reference values for particular joints) are evolved in a realistic simulation (Fig. 2), then transferred to the real mobile robot Ragno for testing. The objective is to develop gaits that enable the robot to move as fast as possible on a flat terrain with the given kinematic and dynamic constraints of its hardware. We do not define the particular gait type that is sought; however, we are looking for gaits that are applicable to the physical robot, therefore they cannot be chaotic, overstress the robot's structure or actuators, or cause the robot to fall down.

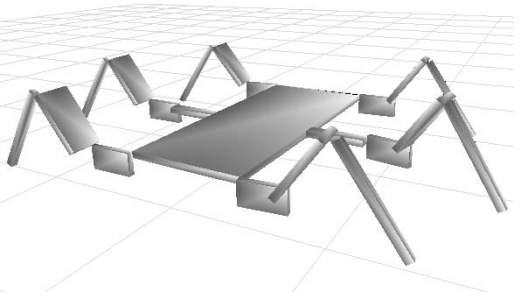

Fig. 2. Simulated hexapod robot.

We show in a series of evolutionary simulations how the implementation of a few, general leg co-ordination rules observed in insects leads to a dramatic reduction of the permissible search space defined by the leg movements. This strategy enables fast and convergent evolution of gaits appropriate for a real hexapod, including patterns significantly different from gaits designed by hand on the basis of their biological counterparts.

2. Related work

Learning-based approaches to gait generation have been broadly studied in the literature. In order to place our work in the context of what has already been done, we study some of the related papers and point out the differences and similarities with regard to our approach.

We try to develop gaits without explicit application of walking patterns observed in animals, because the kinematics and dynamics of hexapod robots are quite different from the respective properties of living creatures. We do not assume that any particular gait type will evolve. Therefore, we evolve whole leg co-ordination patterns instead of just evolutionary optimization of known gait parameters, which is the case in some other works (Hornby *et al*., 2005; Zagal *et al*., 2004).

However, staying away from biological inspirations and starting the search for walking patterns from scratch leads to gaits that are often chaotic and unstable, so they cannot be used to control a physical robot. Busch *et al.* (2002) observed some gaits that cannot found their counterparts in the nature; however, these gaits were slow and not useful for a real robot. This problem is mostly related to a complex state space, in which the search for a solution may be sub-optimal. One of the main questions in this research was how much of the knowledge gained by the observation of animals (insects in particular) should be encoded in the gait controller. We wanted to find a balance between two contradicting aims: enabling the evolution of gaits that are optimal for the specific robot hardware and the chosen fitness function, and avoiding the evolution of unfeasible gaits resulting from convergence with the local optima of a very large search space. A reduction of

this space comes from excluding *a priori* some unwanted solutions (Belter *et al*., 2008). Niching could be additionally incorporated in order to avoid premature convergence (Kowalczuk and Białaszewski, 2006). While it is not optimal to use copied insect gaits with a robot, and it is neither possible nor feasible to copy the Darwinian evolution (Albiez and Berns, 2004), when trying to reduce the search space, it is interesting to take advantage of the rules and/or constraints that proved to lead to a proper solution in natural evolution.

We use an evolutionary algorithm to effectively search for optimal gaits. The term 'evolutionary algorithms' summarizes a family of machine learning and optimization methods inspired by biological evolution (Holland, 1975) that include (among others) genetic algorithms (GAs), genetic programming (GP), and evolution strategies (ESs) (Arabas, 2001). Evolving control systems by using genetic algorithms has become quite popular in the field of robotics (see (Walker *et al.,* 2003) for a survey). Learning and tuning behaviour by means of GA was found useful in developing behaviour-based control systems of wheeled mobile robots (e.g., (Skrzypczyński, 2004a; 2004b).

In the walking machines domain, GA-based methods have been used to evolve gaits for a number of legged robots, including hexapods (Barfoot *et al*., 2006; Gallagher *et al*., 1996; Lewis *et al*., 1994) and octopods (Jakobi, 1998; Luk *et al*., 2001). An evolutionary algorithm has also been successfully applied in the process of developing dynamic gaits for four-legged Sony entertainment robots (Hornby *et al*., 2005). Also other learning algorithms have already been applied to the gait generation problem, most notably different variants of the reinforcement learning (RL) paradigm. For example, six-legged Genghis learned to co-ordinate a simple behaviour set resulting in a tripod gait (Maes and Brooks, 1990).

In (Kirchner, 1998), Q-learning was used to develop swing and stance movements for legs of a hexapod robot. Also Kimura *et al.* (2001) and Svinin *et al.* (2001) used RL to develop motion patterns for a quadruped and an octopod, respectively. We prefer to use an evolutionary algorithm rather than RL, because problems with large state spaces are known to be hard to solve with foundational RL algorithms. While it is possible to decompose *a priori* the search space by enumerating the statically stable robot configurations and eliminating the unstable ones, an explicit description of the stable configurations is not always general and precise enough (Svinin *et al*., 2001). In the system under study such a decomposition might lead to an unnecessary reduction of the admissible search space, and it might impact the ability to learn optimal gaits. Therefore, we prefer to explore the non-decomposed search space, and we require an appropriate learning algorithm to accomplish this. Although several researchers have successfully applied special RL variants to various robot learning problems involving large state spaces (Tuyls *et al*., 2003), we have chosen evolutionary algorithms instead, which can handle complex, non-differentiable search spaces (Goldberg, 1989).

The motivation for using a simulator for learning robot gaits stems primarily from the fact that try-and-error learning using a real robot may be dangerous for the hardware. Also, significant effort (recharging of batteries, repairs) may be necessary to maintain the real robot during continuous testing (Maes and Brooks, 1990). Because of these problems, real-world experiments can impose a prohibitive time overhead. The number of simulated runs performed in a given amount of time can be much higher than that of real experiments. However, any simplification made in a simulator may be exploited by the learning algorithm, resulting in a gait pattern, which cannot be reproduced in the real robot. The problem of transferring the results from a simulation to a real robot has been recognized in the literature (Mataric and Cliff, 1996; Walker *et al*., 2003). The system described here crosses the so-called 'reality gap' (Jakobi *et al*., 1995) by using a simulator that models the physical characteristics of the robot, including its dynamics, and the way a real robot interacts with the environment.

Although there are known results on gait learning using physical robots, we prefer learning in simulation due to reasons related to the aim of this research and the chosen learning method. As stated before, the aim is to evolve gaits that are optimal for the given task and robot hardware. Hence, we try to learn walking patterns from scratch, and we would also like to explore the dynamic states of the robot, beyond the static stability area. Whenever basic motion patterns are acquired through interaction with the environment, the possibility of critical failures increases. The issues of safety during gait learning using an actual robot are considered by Huber and Grupen (1997). One of their safety constraints requires that the walking robot should always remain statically stable. Introducing such a constraint eliminates a large, 'unsafe' portion of the search space but also any possibility to obtain gaits that include dynamic states, which is clearly against our aims. On the other hand, whenever dynamic states are allowed, it is difficult to provide a set of constraints which will ensure robot safety. Such problems obviously do not exist in the simulator, where every collision and every excessive force in a joint is detected by the software.

One can argue that using the actual robot to obtain the reinforcement feedback from the environment should completely remove the reality gap problem. However, in most of the works that use physical robots in gait learning these robots are equipped with special hardware to collect the reinforcement signals (Kimura *et al*., 2001; Maes and Brooks, 1990) or are placed in a completely artificial environment like the treadmill setup designed by Barfoot

et al. (2006) to evolve walking gaits. Hence, robots are not confronted with the real world as it is, because the additional equipment may cause errors or bias in the reinforcement signal (e.g., due to wheel slippage) or even introduce factors that are not present in the natural environment (Barfoot *et al*., 2006). Therefore, real robot-based gait learning systems are not completely free from the reality gap problem. However, in the real world, there are situations that cannot be foreseen, and it is not possible to program or learn an appropriate set of reactions for it in advance. As was observed by Kirchner (1998), such problems should be solved by the robot itself during interaction with the environment, and that is where learning using a real machine should be used. Real robot learning may be also used for the adaptation of an existing control system to particular properties of the environment. An example of such a learning system is shown by Chernova and Veloso (2004), where quadruped robots used for robotic soccer adapt to a new walking surface.

A well-established paradigm in the evolutionary robotics literature is the minimal simulation. The minimal simulation principle was proposed by Jakobi (1995), who applied it to different robots, including an eight-legged walking machine (Jakobi, 1998). The same approach is also used by Svinin *et al.* (2001), where the minimal simulation model of an octopod is explained in detail. In contrast, we use a simulator based on the exact dynamic model of the Ragno robot. Researchers using the minimal simulation approach argue that a learning procedure based on a model of a robot's dynamics can be prohibitively slow (Jakobi *et al*., 1995). However, recent advances in computer speed and the availability of fast, optimized software 'engines' implementing the simulation of real-world physics made such simulators a viable solution for evolutionary robotics. While a minimal simulation may be enough to provide a reinforcement for a wheeled mobile robot performing a simple navigation task, legged robots should explore their dynamics to achieve effective gaits.

The minimal octopod robot model developed in (Svinin *et al.,* 2001) excludes any dynamical effects related to the balancing motions of the body and assumes that the proper control commands lead to stable configurations. Whenever the robot comes into an unstable configuration, the command that led to this configuration immediately gets a negative reinforcement. Obviously, such an approach does not permit the evolution of gaits that include dynamic states. As pointed out by Gallagher *et al.* (1996), who use a simple simulation to evolve gaits for a hexapod robot, an important difference between the simulated and the real robot is that the oversimplified simulated robot falls down as soon as static stability is lost, while the real robot needs time to fall. Observing our results (cf. Fig. 17), one can see that the more elaborated, non-minimal simulator enables the learning system to explore this fact, and to evolve a fast gait that does not preserve the static stability all the time.

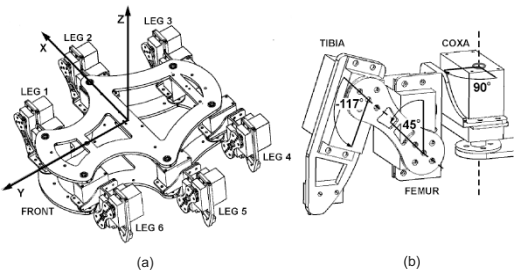

Fig. 3. Robot co-ordinate system and leg numbering (a). Neutral position of a leg (b).

3. Experimental setup

3.1. Hexapod robot. The walking robot Ragno (cf. Fig. 1), which was developed in-house, is used in this research. It has six legs. Each leg has three joints that are driven by integrated servomotors. Figure 3(a) shows a general view of this robot's mechanics, its local co-ordinate frame and the leg numbering convention used. The robot is 33 cm long and 30 cm wide and weights 2.15 kg (without batteries).

The control architecture of the robot is divided into four layers (Walas *et al*., 2008). The first one is placed off-board. It has sufficient resources to compute the appropriate control signal for all joints of each leg. The off-board layer sends control commands to the robot. The commands include 18 reference values for the leg joints. These reference values are determined as a difference between the desired joint positions and the neutral ones. The neutral position for a leg is shown in Fig. 3(b) and is defined by the vector of the joint angles: $[90°, 45°, -117°]^T$. To change the leg position, the difference between the reference and the neutral position has to be sent to the leg joint controllers. Sending the zero vector for a given leg means setting this leg to its neutral position.

The second, on-board control layer interprets these commands and sends them to the appropriate leg controllers. There are six leg controllers that work simultaneously. They produce control inputs for the integrated servomechanisms of the joints. Each joint has a feedback from its angle position. Additionally, the robot has a double axis accelerometer and a gyroscope to measure trunk orientation in a 3D space. The on-board and off-board parts of the control system communicate by means of a Bluetooth connection.

The robot has been programmed by hand to walk with two statically stable gaits: a simple crawl and a tripod-like gait. The crawl is used at slow speeds—at most one leg is transferred at a time. In the tripod gait two sets of three legs each are moved repeatedly. This is the fastest statically stable gait for a hexapod, and according to Wilson (1966) it is one of the standard gaits of insects. The

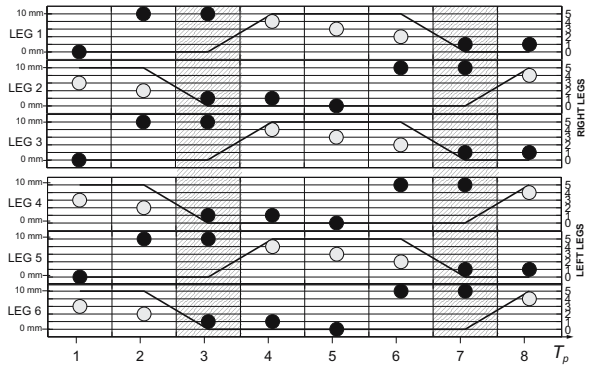

Fig. 4. Tripod-like gait designed by hand.

robot has always at least three legs placed on the ground, and its centre of gravity is located within the support polygon. This is visible on the gait diagram (Fig. 4), where the consecutive columns represent the reference configurations of the robot legs which are issued every time step (leg states are numbered on the right side of each diagram row). The T_p value corresponds to the sampling period of the control signal. The black circles denote legs in the support phase, while the light grey circles denote legs in the transfer phase. To make visual gait analysis easier, a solid line on each part of the diagram shows the height above the ground at which the tip of a particular foot is raised at a given moment in time (the scale is given on the left side of the diagram). With a single step length (the distance between two consecutive footholds of one leg) set to L_s=50 mm, the maximal walking speed of the Ragno robot while using this gait and moving straight forward is $0.09\ \mathrm{m/s}$.

3.2. Robot simulator. To carry on gait synthesis experiments and test new control algorithms, a simulator for the Ragno robot has been built. It includes a robot dynamics simulation based on the Open Dynamics Engine (ODE) (Smith, 2007). The ODE provides tools for simulation of rigid body dynamics and complex structures with defined joints. It also provides an engine to detect collisions and simulate friction between bodies. Each robot link is represented by a cube with the appropriate dimensions and mass. For the sake of simulation speed the geometry of the simulated robot's body and limbs is much simplified. The simulator provides virtual measurements of all the kinematic variables of the robot, as well as the forces/moments in all the joints. The 3D-pose $[x\ y\ z]^T$ of the robot's local co-ordinates (Fig. 3(a)) is computed at every time step of the simulation with regard to the global co-ordinate system, which is equal to the initial position of the robot. Simple visualization of the robot model (cf. Fig. 2) is implemented by using OpenGL routines. The controller code used in the simulator can be directly applied to the real robot. Every actuator is modeled as a proportional controller, with the output y defined as

$$y = k \cdot (\alpha_0 - \alpha), \tag{1}$$

where α_0 is the desired joint position, α is the real joint position, and k is the controller gain. Controller gain values are determined experimentally in such a way that a speed similar to that of the real robot servomotors is achieved. Maximal torque value is limited to 0.9 Nm, as in the actual servomotors.

4. Evolutionary learning system

4.1. Evolutionary algorithm. For the problem of walking robot gait generation, an evolutionary algorithm offers the possibility to select useful gaits by means of reinforcement, and to discover new gaits by means of genetic operators. There are several different approaches exploiting the evolutionary paradigm (GA, GP, ES), which differ with respect to the encoding of the problem as the population of individuals, and with respect to the genetic operators used. To choose the appropriate learning method for the problem of hexapod gait generation, we have to take into account the properties of the problem itself, and parameters of the simulation system we use:

- As the reference values for all the leg joints have to be determined, a large space of possible solutions must be explored.
- We are looking for an optimal sequence of steps (leg movements) over a given time period, which makes the search space even larger.
- There are no heuristics available which could help to define the required population size, number of generations or the particular genetic operators.
- The joint reference angles are real values.
- Most of the simulation time is consumed by the computation of the robot dynamics.

Classic GAs use individuals represented as binary-encoded genotypes and require a mapping from the genotype to the real-valued problem representation. If the problem representation requires many real-valued parameters, as is the case for the gait generation problem, the binary encoded genetic representation can be very long. Therefore, we prefer an evolutionary algorithm that enables direct real-valued representation of the genes. Another issue related to classic GAs is parametrization. Although the GA has proven to be a robust heuristic search technique in many applications (Goldberg, 1989), it is not clear how it should be parametrised and configured to fit a particular problem (Bäck *et al.*, 1991). It would be a non-trivial task to make a proper choice of a selection strategy, a recombination (crossover) operator, and the probability of mutation for the GA-based gait learning system.

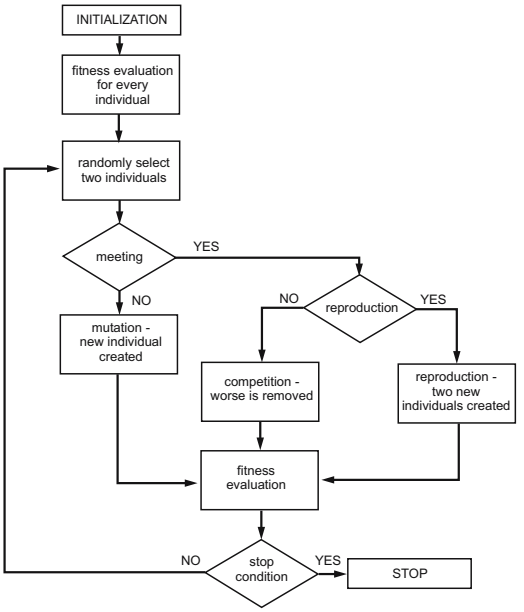

Fig. 5. Block diagram of the evolutionary algorithm.

This could require many simulations to be run and may take a long time, and if poor settings are used, the performance of the evolutionary system can be severely impacted. Therefore, an algorithm that is able to self-adapt its own parameters during the search, as is accomplished in evolution strategies (Bäck *et al*., 1991), is an interesting choice for the problem considered here.

Such a self-adaptive, population level-based evolutionary algorithm for optimization problems is proposed by Annuziato and Pizzuti (2000). The main idea underlying this algorithm is to move the evolutionary metaphor from the genetic level towards that of artificial evolution of societies. The traditional concept of selection in GAs is replaced with a direct competition among individuals, while crossover and mutation are viewed as two different ways of reproduction in a population (Fig. 5).

Each probable solution for the given problem is represented as an individual, which can be encoded directly using real-valued genes. At the start of the algorithm, a population of individuals is randomly generated. The initial size of the population is also a random number, smaller than the maximum population size M_p. The maximum population size is the only user-defined parameter in this algorithm, and it represents the limited resources of the environment. In our case this limit is set with reference to the limited capacities of the simulator. The actual population size is determined by the balance of reproduction and competition among individuals, and the adaptation rules of the genetic operators are driven by the population density.

The selection strategy controls the character of the search and has to balance between the contradicting aims of finding the global optimum and converging quickly (Bäck *et al*., 1991). In the population level-based evolutionary algorithm, the selection is replaced by the meeting concept. In each iteration, two individuals are randomly selected from the population. The probability of a meeting between these two individuals is defined as the population density:

$$P_m = \frac{C_p}{M_p}, \tag{2}$$

where C_p is the actual population size. The denser the population, the higher the probability of meeting another individual. When two individuals meet together, then they can generate offspring (sexual reproduction) or compete (the stronger kills the weaker), otherwise the current individual is mutated.

In the population level-based approach, the typical crossover is replaced by sexual reproduction. The decision whether the individuals that have met will mate depends on the probability of reproduction:

$$P_r = 1 - \frac{C_p}{M_p}. \tag{3}$$

If the population density is low, then the reproduction probability is high, because the individuals do not need to compete for resources. When two individuals meet and mate, they exchange their genes according to a given crossover scheme, but the resulting offsprings, do not replace their parents, and they are added to the population. If two individuals meet but do not mate, then they are compared in order to choose the one with the better fitness value (a metaphor of fight). The worse one is removed and the population size decreases.

In the adopted evolutionary algorithm, mutation is replaced by non-destructive reproduction. It occurs with the probability $1 - P_m$, when the chosen individual does not meet another one. At first the individual is cloned and the copy is added to the population, and then the original is mutated.

According to Annuziato and Pizzuti (2000), the algorithm ends when the maximum number of iterations is reached. In our implementation, an additional stop criterion is introduced—the algorithm ends when the best individual fitness does not improve for a number of iterations.

4.2. Problem representation. In our approach, each individual is represented by a genotype that defines the reference values for all the robot's legs over a given amount of time. Genotype length n is defined as

$$n = \text{ceil}\left(\frac{T_s}{T_p}\right), \tag{4}$$

where T_s is the simulation time, and T_p is the sampling period of the control signal (both in seconds). The value of T_p=0.1s is derived from the maximal control speed—the

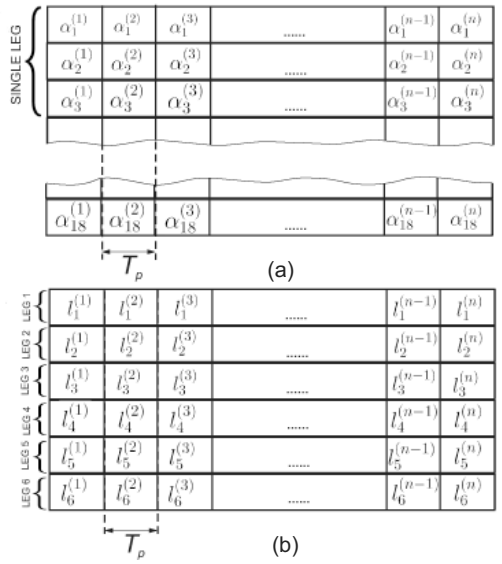

Fig. 6. Alternative structures of the genotype.

robot allows 10 orders and feedback frames to be transmitted per second, while the simulation time is defined by the user. Because the simulation step size is set to 1 ms, each genotype column encodes the reference values for all the legs for 100 simulation steps.

During the simulations presented later in the paper, we used two different methods to set the leg reference values, therefore the structure of the genotype changed accordingly. When we define separate reference values for particular joints of the legs, the genotype represents the reference angle (control input) for each of the 18 servomotors as a discrete function of time, and thus it consists of 18 chromosomes (Fig. 6(a)). Genes store the real angle values $\alpha_i^j, (i = 1, \ldots, 18; j = 1, \ldots, n)$ interpreted as an angular difference from the neutral joint position. These values are bounded by the maximal range of servomotors movement in the real robot.

The alternative problem-encoding scheme is based upon a definition of discrete states for each robot leg. Because individual legs move cyclically during walking, to facilitate gait analysis, two phases are distinguished in the movement of a leg:

- support phase, when a leg supports and propels the robot,
- transfer phase, when a leg moves from one foothold to the next one.

Then, three particular positions of a leg relatively to the robot body can be defined:

- neutral position,
- anterior extreme position, when a leg reaches the foremost point,

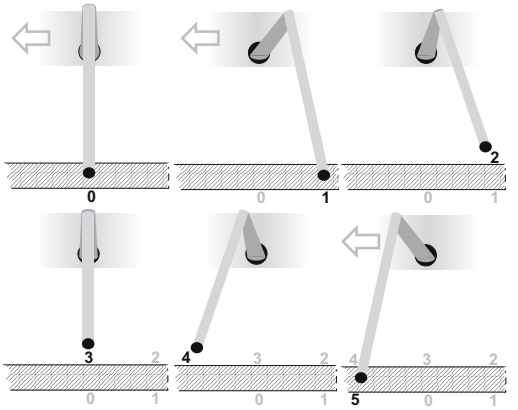

Fig. 7. Possible states of a leg.

- posterior extreme position, when a leg reaches the backmost point.

Using these definitions, six configurations of a leg (Fig. 7, arrows indicate the motion direction) can be defined:

0 : support phase, neutral position: the tip of the leg on the ground,

1 : support phase, posterior extreme position: the tip of the leg on the ground,

2 : transfer phase, posterior extreme position: the tip of the leg above the ground,

3 : transfer phase, neutral position: the tip of the leg above the ground,

4 : transfer phase, anterior extreme position: the tip of the leg above the ground,

5 : support phase, anterior extreme position: the tip of the leg on the ground.

The joint reference values for the above-defined positions of robot legs were computed by using forward and inverse kinematic equations for the leg, resulting in a step length L_s=50 mm. All the positions of the feet lie on the same plane. A discrete state of a leg is recognized by its number and it unambiguously defines all the joint reference angles. Therefore, encoding the individual reference values for each leg joint in the genotype is no longer required. In this case, the genotype consists of six chromosomes, each of them representing the state of a given leg as a discrete function of time (Fig. 6(b)). The genes are encoded as integer values $l_i^j, (i = 1, \ldots, 6; j = 1, \ldots, n)$ bounded to represent the allowed states of a leg, i.e., they can be numbers from 0 to 5.

Regardless of the particular structure of the genotype employed, in all the simulations presented the sexual reproduction uses a two-point crossover to form two new individuals from two parents—we expected it to be more

effective on long chromosomes than the one-point version. The crossover operator uses a whole column of the proposed genotype as a gene while the mutation operator changes only one gene. In most evolutionary systems, a genotype is defined as a string of genes, with particular chromosomes being just parts of this string. We use a two-dimensional genotype, where a chromosome (a row of the genotype) defines the reference states for a given leg over time. On the other hand, a column of the genotype may be considered a definition of a particular 'control step' of the robot, as it represents the reference configuration of all the legs, which is sent to the robot's on-board controller every 0.1 s time step. Therefore, the crossover operation performed on the entire columns is an exchange of control steps between two individuals representing two different gaits. This definition of the crossover operator preserves the already developed control steps from being partitioned and promotes the development of new sequences of control steps, possibly leading to more effective gaits. The mutation operator changes only a single gene at once, thus it affects only the state of one leg in a particular control step. This is the mechanism of discovering new control steps that may be useful in the whole gait.

4.3. Gait learning strategy. The simulations reported in this section show evolutionary learning results while the space of possible solutions is gradually reduced. Simulations start from scratch. Next, the solution space is reduced by implementing selected leg co-ordination rules known from the observations of insects. These rules are applied separately rather than as a complete, biologically inspired walking pattern, in order to allow also the gaits that are not observed in insects to evolve. As a consequence, the genotype size is also reduced gradually, due to the implementation of the knowledge which has been tested and accepted so far. According to the robot shaping concept, the multi-stage learning is supervised by a human coach, who picks up the best solutions and applies them in the next simulations. In research on animal walking, three important rules have been observed:

- Most animals use periodic locomotion patterns when they walk on a flat terrain, in such a gait a short sequence of leg movements is repeated periodically.
- In insect gaits the legs on one side of the body have a phase difference with regard to their contralateral legs, i.e., both laterally paired legs are never raised from the ground at the same time.
- In insect gaits a leg has a phase difference with regard to its anterior leg, i.e., two neighboring legs on the same side are never raised from the ground at the same time.

Using these biological inspirations, and taking into account the already introduced definitions of the genotypes, the following strategy for evolutionary simulations has been proposed:

1. learning from scratch with the search space defined by separate joint angles;
2. learning from scratch with the search space defined by the six discrete configurations for each leg;
3. learning in the discrete search space with the periodic gait property imposed;
4. learning in the discrete search space with the phase difference between contralateral legs imposed;
5. learning in the discrete search space with the phase difference between neighboring legs on the same side imposed;
6. learning in the discrete search space without the periodic gait property imposed, but both rules tested in Simulations 4 and 5 are used;

The fitness function used aims to maximize the forward velocity of the robot and depends only on the forward distance traveled during the time period of the simulation:

$$F = \frac{y(T_s)}{T_s}, \tag{5}$$

where $y(T_s)$ is the robot trunk position along the global y axis at the end of simulation. Simulation time T_s is set to 1.6 s, which is sufficient to calculate a few steps of the robot. Individuals which reach a longer distance are considered to be better adapted to the environment. All the simulations presented were carried out with the same fitness function and the maximal population size set to M_p=500 to examine how the biologically inspired constraints imposed on the search space influence the results.

5. Evolutionary learning in simulation

5.1. Simulation 1. First, we attempted to evolve a hexapod gait with the genotype with the form like that shown in Fig. 6(a). After the first simulation, the best evolved individuals were able to make only an unsystematic movement. There were no cycles observed in this movement. Every leg of the simulated robot worked independently, and the simulated robot used small jumps to move. Some of the legs cooperated to transport the robot trunk in the direction of the movement. Although the robot's average speed was $0.125\ \mathrm{m/s}$, the evolved gait is not applicable to the real robot because it is very chaotic and can overstress the servos.

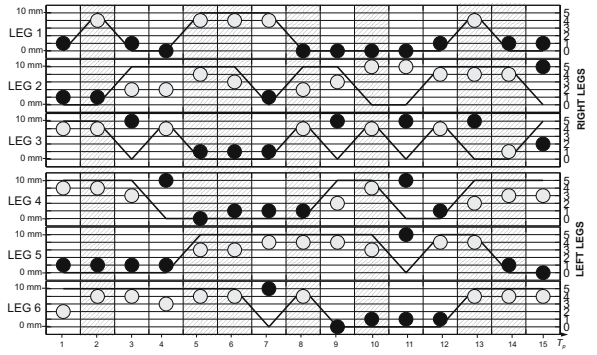

Fig. 8. Irregular gait obtained in the second simulation.

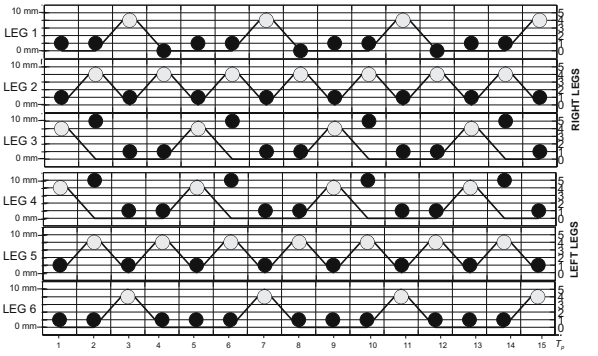

Fig. 9. Periodic gait obtained in the third simulation.

5.2. Simulation 2. Because evolution in the continuous state space had turned out to be ineffective, we approached the problem differently, by allowing for only six discrete states for each robot leg and, thus, by reducing substantially the dimensionality of the search space. However, the resulting number of states, $6^6 = 46656$, is still rather large. The genotype has the form shown in Fig. 6(b).

The gaits obtained in the second simulation were also irregular. No cycles in the movement of the legs were found. Every leg changes its role in walking. Some of the legs have phase difference in relation to the others, as in the tripod gait, but after a while they change their phase, and then, for example, contralateral legs work without a phase difference. The movement of some legs is only the result of the movement of the robot's body, as these legs work to preserve the stability of the trunk, but do not generate movement in the direction of walking. The average speed achieved in simulation with this way of walking was $0.073\ \mathrm{m/s}$, and this gait was still not applicable to the real robot. This is visible on the gait diagram (Fig. 8), which shows some control steps (denoted with a slanted background), where fewer than three legs are placed on the ground, which made this gait statically unstable and caused the robot to fall down.

5.3. Simulation 3. In the next experiment, we retained the discrete state space of the leg movements, but additionally we applied the periodic property of an insect-like gait. The information about the cycle length l_c was added to the genotype as an additional chromosome with only one gene. Now, only the beginning part of each chromosome, with the l_c-genes length, is used to evolve the gait. This sequence is repeated periodically.

Although in this simulation cyclic gaits are produced, a fast and stable gait was not obtained. Most of the resulting gaits were based on small jumps involving a number of various leg combinations. When the periodic property was used, the evolutionary algorithm converged to a solution very quickly. Although the obtained gaits were regular, the covered distance was usually quite short. An example of an evolved gait is presented in Fig. 9. This gait is appropriate for use with a real robot. The leg movement sequence of the Ragno robot during one gait cycle is shown in Fig. 10. The grey dots indicate those legs, whose tips were placed on the ground in due time. The average speed in this simulation was $0.08\ \mathrm{m/s}$.

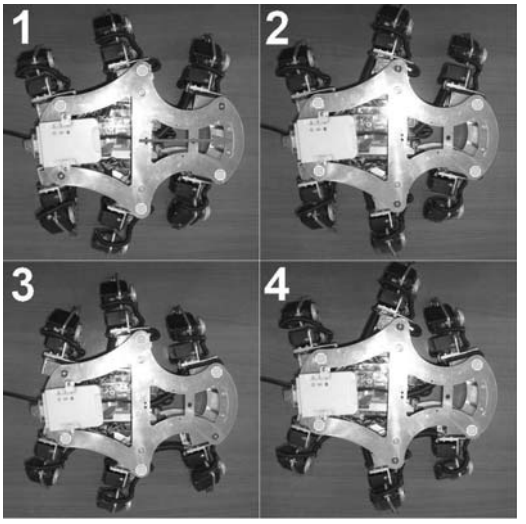

Fig. 10. Real robot using the gait from the third simulation.

5.4. Simulation 4. In research on insect walking it was found that contralateral legs of an insect have 180° phase difference (Wilson, 1966). In a cyclic gait, the 180° phase difference is defined as a state shifted by half of a whole leg movement cycle. Using the previously defined six leg states, this phase difference is defined as follows: if a leg is in state 0, a 180° phase shifted leg is in state 3; if a leg is in state 1, a shifted leg is in state 4; if a leg is in state 2, a shifted leg is in state 5, etc., which can be written as

$$l_k^{+180^\circ} = \begin{cases} l_i + 3, & \text{for } l_i \leq 2, \\ l_i - 3, & \text{for } l_i > 2, \end{cases} \tag{6}$$

where l_i is the state of the i-th leg, and $l_k^{+180^\circ}$ is the state of the k-th leg that has the 180° phase difference to the i-th leg.

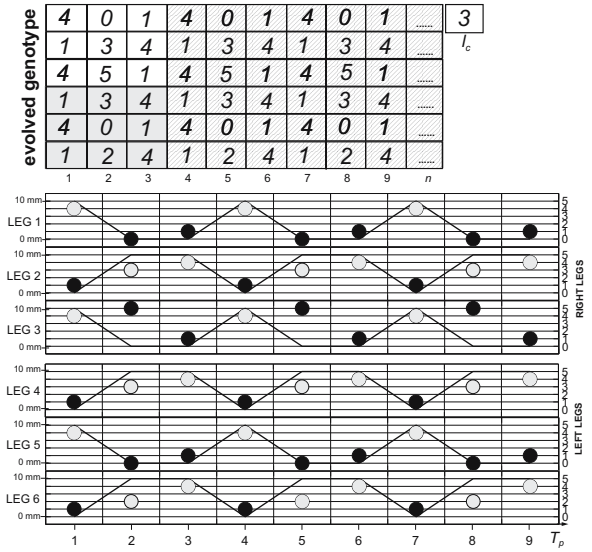

Fig. 11. Genotype and gait resulting from the fourth simulation.

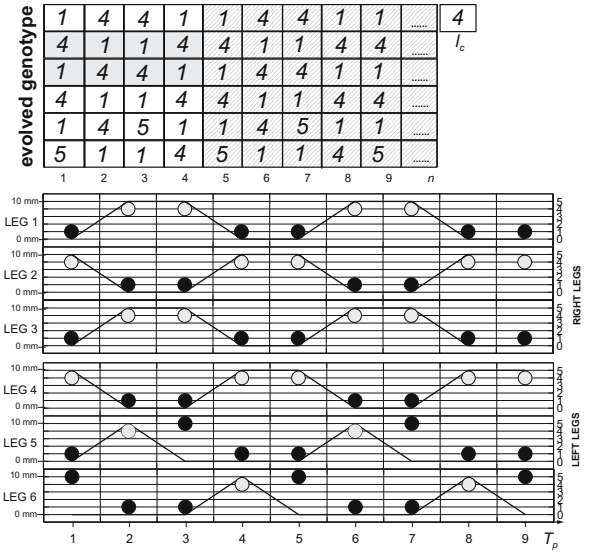

Fig. 12. Genotype and gait resulting from the fifth simulation.

In accordance with that observation, in the next simulation the first three chromosomes were rewritten to the next three by using the following rules: when a right leg is in state 0, a left leg is in state 3; when a right leg is in state 1, a left leg is in state 4, etc. These rules limited even more the search space in the evolutionary learning system (only chromosomes for the legs 1, 2, and 3 are evolved).

The simulation results are shown in Fig. 11. As is visible from the gait diagram, the evolved walking pattern is a tripod, and it is applicable to a real robot. The average robot speed in this simulation was 0.093 $\mathrm{m/s}$, and the single cycle length which evolved was 3 T_p (0.3 s). All reference values for the left side legs are determined from the imposed insect gait rule. In the genotype (Fig. 11), the reference values that come out from the imposed rule are shown on a grey background, while the genes that result from the periodic property of the movement have a slanted background.

The most interesting outcome of this simulation is the observation that the reference values for the first three legs depend on each other. The evolved reference values for the second leg have a 180° phase difference to the first leg, and most of the evolved states of the third leg again have a 180° phase difference to the second leg. This pattern of leg states closely resembles the insect walking rule we plan to use in the next simulation: the phase difference between neighboring legs on the same side of the robot. However, these dependencies are not a result of an imposed rule, but they evolved in the simulation, so there are some 'inaccuracies' in the pattern. According to the rule, the third chromosome should be equal to the first one, but there is a value of 5, where it should be 0. This is caused by the nature of the evolutionary algorithm, which very often finds only a near-optimal solution (in our representation of the problem the reference values 5 and 0 are in the vicinity). In many applications of genetic learning this is not a problem, but here, because of the discrete and cyclic leg movements, the sub-optimal chromosome causes an incorrect leg position in the support phase of the gait. As a result, the robot does not walk straight forward.

5.5. Simulation 5. The next experiment was undertaken in order to verify how the inter-leg dependencies observed in the Simulation 4 work as a rule. The cyclic property of the gait was also imposed. In this step the legs on the right side of the robot had a 180° phase difference with regard to their anterior legs. Thus, on the right side, only the chromosome for the leg 1 was evolved. Chromosomes for the left-side legs were evolved without additional rules, taking into account only the cyclic gait property. The walking pattern, shown in Fig. 12, was similar to a tripod; however, in some control steps more than three legs were placed on the ground simultaneously, which makes this gait less effective than the previous one—the average robot speed achieved during the simulation was 0.086 $\mathrm{m/s}$. The single cycle length which evolved was 4 T_p.

What is particularly interesting is that in the gait diagram (Fig. 12) the rule used in Simulation 4 can be seen as a result of the evolution. The reference values for right-side legs are in opposition to those of left-side legs.

5.6. Simulation 6. Simulations 4 and 5 showed that the application of only one, very simple, biologically inspired rule reduces the search space enough to enable fast evolution of gaits that are feasible for a real robot. However, these results were obtained with the cyclic gait property being imposed. Therefore, the next simulation was

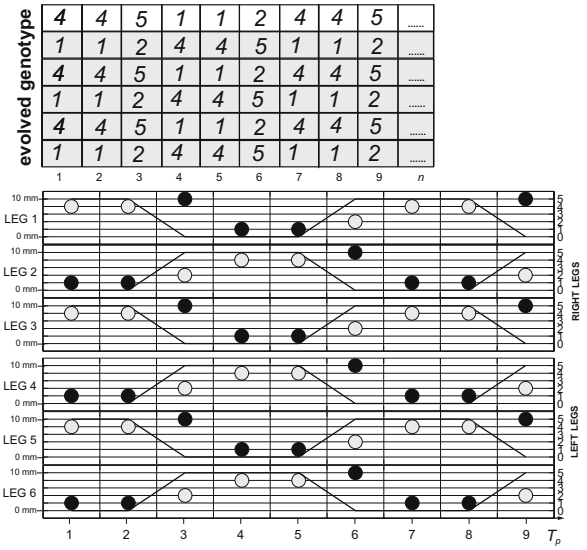

Fig. 13. Genotype and gait resulting from the sixth simulation.

conceived in order to verify the assertion on the cyclic, repeated control steps. In this simulation both rules tested previously were used, apart from the periodic gait condition. Only the chromosome representing the first leg evolved. The third and fifth chromosomes were kept equal to the first one, while the chromosomes for the second, fourth and sixth leg had a 180° phase difference with regard to the first leg.

As a result, a smooth, cyclic tripod gait was obtained very quickly (Fig. 13). As can be observed in the genotype, the single cycle length is 6 T_p. However, there is no separate gene responsible for the evolution of the cyclic property—it emerged naturally as a result of the imposed leg co-ordination rules. The learning system repeats some chromosome parts and achieves a cyclic tripod gait as the best one to perform the task defined by the proposed fitness function. The best individual was able to reach an average speed of 0.098 $\mathrm{m/s}$. This result makes this evolved gait faster (in simulation) than the kinematic tripod-like gait designed by hand.

Having achieved a regular, fast tripod gait in the evolutionary simulations with the discrete search space of the leg movements, we became interested in obtaining a similar result with the search space described by the joint angles, where the evolved reference signals for the servomotors take real values. To achieve this, the genotype structure shown in Fig. 6(a) was used again. The general idea of gait evolution was the same as the one tested before in Simulation 6 with the discrete state space. The evolutionary algorithm made a slower progress than in the previous simulation. The gait obtained was periodic, but not regular. However, the robot achieved a speed of 0.099 $\mathrm{m/s}$.

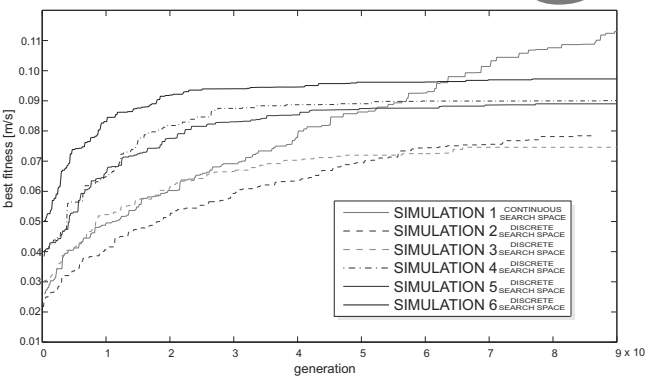

Fig. 14. Learning speed in the simulations.

5.7. Performance of the evolutionary algorithm. Solving high dimensional problems by using an evolutionary algorithm takes a long time. The reduction of the search space helps to find a solution faster. This becomes evident from Fig. 14, where the fitness evolution of the best individual in the population for an average of 10 runs is depicted as a function of the number of iterations for the simulations described in Section 5. When the search space is reduced, the appropriate gait is determined much faster (Simulations 4, 5 and 6). In all simulations the actual population size C_p stabilizes at 75% of M_p.

6. Evolutionary identification of the simulation model

6.1. Reality gap problem in gait learning. Downloading the results from simulation to the robot controller showed that there are still some problems due to the reality gap. The main reasons are the differences between the simulation model dynamics and the real robot, as well as those between the environmental parameters (like the surface friction coefficient) assumed in the simulator and the real-world parameters. This is not a major problem whenever a statically stable walking pattern (e.g., the tripod-like kinematic gait) is used. However, some of the evolved gaits caused the robot to use the dynamic properties of its body to move, dynamically switching the support polygons without much of a static stability margin. In such a case, even small discrepancies between the model and the real robot may cause unwanted robot fall-downs.

To make the results evolved in the simulation easy to be used with a walking robot, the reality gap problem has to be solved completely. This may be accomplished using the evolutionary optimization of parameters of the simulation model (Zagal *et al*., 2004). As a result, the simulated robot should react to the control signals much like the real robot does. An interesting aspect of this approach is that the same evolutionary algorithm which is used for gait learning can also be re-used to identify the simulated robot parameters, making the whole method

more effective from the point of view of the programming effort required to cross the reality gap. Although there are many system identification techniques used in robotics (Kozlowski, 1998), which are based on exact mathematical models of robots, such a model of a multi-legged walking machine, even if identified successfully, would be infeasible for our purposes. We are searching for a set of parameters that enable the much simplified simulated model to mimic the real robot behaviour within the chosen class and range of control signals. Thus, we are facing a search/optimization problem rather than a typical identification problem. Evolutionary and other soft computing techniques are known for being well suited to solve problems belonging to this class, which motivated our choice. In parallel we performed the same robot parameter optimization with a different soft computing technique—particle swarm optimization, which is covered elsewhere (Belter and Skrzypczyński, 2009).

6.2. Parameter identification system. The structure of the Ragno robot simulation model is described by four parameters: the trunk mass, and the masses of the tibia, femur and coxa links constituting a leg (all legs have the same parameters). Two additional parameters are related to the servomotors. One describes the gain of the controller in every joint, and the other defines maximal angular speed of joint movement.

For the identification task, the population level-based evolutionary algorithm described in Section 4.1 was applied, but with different problem encoding and parameters. This time the genotype is one-dimensional, and consists of a single chromosome with six genes directly representing the real-valued model parameters. The maximum size of the population was set to 100 individuals.

The general idea of identification is to find a set of parameters that jointly minimize the discrepancy between the trajectories of some characteristic points of the robot observed in simulation and in the real Ragno walking machine. However, for this task the choice of such points and their trajectories is not straightforward, because the parameter values we search for manifest themselves most prominently in different types of movement, which are often dynamic and hard to track in a physical robot.

Taking this into account, we propose to use a set of reference trajectories registered during eight independent and relatively simple experiments with the physical robot. These experiments were conceived in such a way that in each of them the results depend only on few parameters of the model. These experiments provide sufficient reference data to produce a model of the robot which is general enough for the task under investigation. However, if the number of input trajectories is too small, the obtained robot model will only work properly for exactly the same experiments which were used to obtain the reference data. The following experiments (Fig. 15) were performed, and

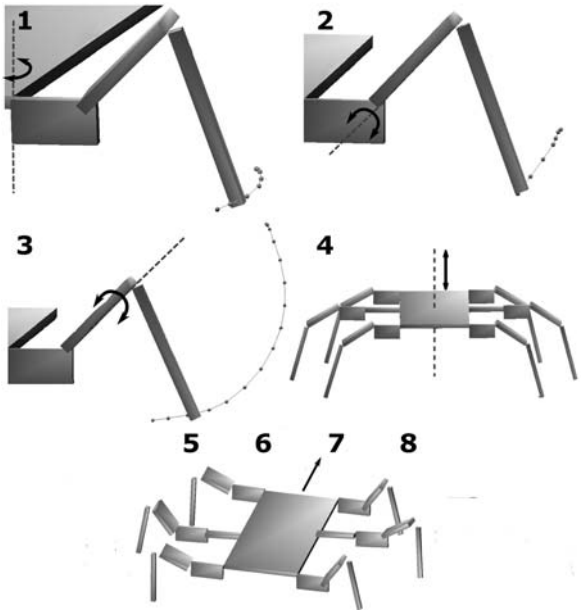

Fig. 15. Robot model identification experiments.

appropriate results were registered:

1. The reference angle in the first joint of a single leg was changed from 0° to -45°. The trajectory of the foot was registered.

2. The reference angle in the second joint of a single leg was changed from 24° to 55°. The trajectory of the foot was registered.

3. The reference angle in the third joint of a single leg was changed from -114° to 0°. The trajectory of the foot was registered.

4. The robot executed an order to move its trunk up, and after reaching the desired position it moved the trunk down. The trajectory of the center point of the robot trunk was registered.

5–8. The robot executed four dynamically stable gaits. They are periodic, but differ in cycle length and average speed. The distance covered was compared to simulation results.

To keep the identification experiments simple, we decided not to include the ground friction coefficient in the optimization procedure. In real robot tests, all of the obtained gaits were tested in a lab, on a surface that almost eliminates leg skidding. To compensate for this, the ground has a high friction coefficient also in the simulator. Therefore, the results of the experiments were determined mostly by the dynamic properties of the robot's body.

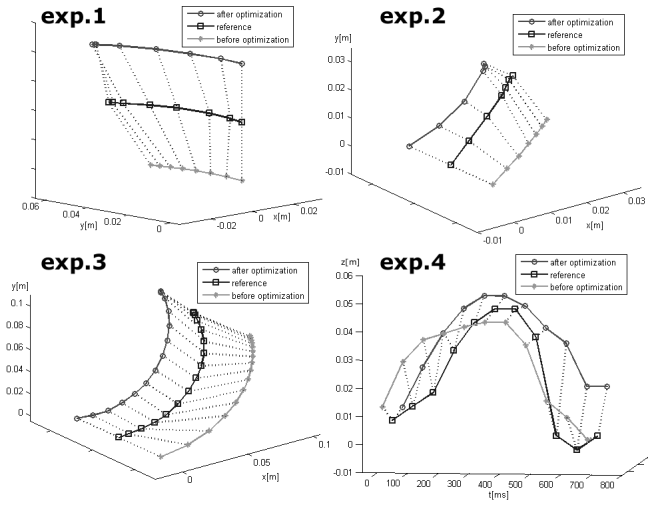

Fig. 16. Test trajectories for the different robot models.

For Experiments 1–4 the fitness function F_j is defined as a sum:

$$F_j = \sum_{i=1}^{N} \left| p_{ji}^{\text{ref}} - p_{ji}^{\text{sim}} \right|. \tag{7}$$

where j is the number of the experiment, N is the number of points of the trajectory, and p_{ji}^{ref} and p_{ji}^{sim} are the corresponding points of the reference trajectory (from the real robot) and the trajectory obtained in the simulation, respectively. The fitness function in the remaining four experiments is simpler; it is defined as the distance between the end position of the real robot and the end position of the simulated one. With the above definitions, the various parts of the fitness function may take quite different values. Therefore, to make all experiments equally important for fitness assessment, the final fitness F is weighted:

$$F = \sum_{j=1}^{8} c_j \cdot F_j, \tag{8}$$

where F_j is the fitness value in the j-th experiment, and c_j is the weighting coefficient set to 1 for experiments $j = 1, \ldots, 4$ and to 6 for experiments $j = 5, \ldots, 8$.

The identification of the robot model parameters was performed in a multi-stage manner. Again, the search space was reduced gradually. However, for the model parameters there are no rules or constraints that we could use to reduce the search space as in the way we used the insect walking rules for the main gait learning task. Because of this, the search space reduction method proposed by Perry *et al.* (2006) was employed. This strategy reduces the search space for parameters which converge quickly. It allows us to save time otherwise wasted on testing solutions, which are far from the optimum. First, the search space was limited to values that guarantee stable operation of the simulator. Next, three identification simulations were run. After considering the obtained results, we set new limits to the search space.

Some parameters, such as the maximal velocity and the servomotor gain, converge quickly. Thus, for the main simulations the search range of these parameters was substantially reduced. For the remaining parameters the search space was only slightly reduced. When the search space had been reduced, the main identification experiment was conducted. The reference trajectories obtained with the Ragno robot and used in Experiments 1–4 are compared in Fig. 16 to their counterparts obtained in simulation, before and after model parameter optimization.

The optimization results show that the simulated robot with the parameters set by hand was slower than the real one. This can bee seen in Fig. 16, where the foot trajectories generated with the simulated robot parameters set by hand differ significantly from the reference trajectories of the real robot. The trajectories obtained in a simulation with the best evolved parameters are considerably more similar to the reference trajectories. Also the forward motion speed of the simulated robot with hand-tuned parameters was much lower than the average speed achieved by the real robot. After the optimization of the model parameters, this discrepancy was much smaller, which is shown in Table 1. This table contains also results of two experiments conducted in order to verify the results on gaits that differ significantly from the gaits included in the optimization procedure. These gaits are not regular tripods and they have increased step length. As can be seen, for the movements/maneuvers that more radically differ from the movements included in the 'training set', the improvement in performance is smaller, but it is still at least 20%. The results shown in Table 1 are averaged over 10 simulations or real robot experiments. The small standard deviation values σ_v suggest that these results are repeatable.

7. Validation experiments

The gaits resulting from the evolutionary learning procedures described here as Simulations 3–6 were implemented and tested on the real Ragno robot in both indoor and outdoor settings[1]. To take practical advantage of this research, we wanted to include the best evolved gait as a part of the standard control software of our robot. The two gaits obtained with the sixth simulation procedure were the fastest. Although the simulated robot was marginally faster with the gait evolved in the continuous state space, in practice there was a problem with this walking pattern. The pace length was not constant, thus the robot moved with a non-uniform speed. Finally, the sixth gait

[1] A video clip is available at
`http://lrm.cie.put.poznan.pl/ragno.avi`.

Table 1. Optimization results—average speed v_m and its standard deviation σ_v.

forward motion speed		exp. 5	exp. 6	exp. 7	exp. 8	verification 1	verification 2
real robot	v_m [m/s]	0.134	0.141	0.125	0.133	0.146	0.171
	σ_v [m/s]	0.003	0.003	0.003	0.004	0.004	0.005
simulator before optimization	v_m [m/s]	0.090	0.089	0.097	0.089	0.077	0.083
	σ_v [m/s]	0.005	0.003	0.002	0.004	0.003	0.005
	error [%]	32.8	36.9	22.4	33	47.2	51.8
simulator after optimization	v_m [m/s]	0.129	0.142	0.107	0.139	0.177	0.220
	σ_v [m/s]	0.004	0.003	0.002	0.003	0.005	0.002
	error [%]	3.7	0.7	14.4	4.5	21.1	29.5

evolved with the discrete search space was selected for permanent use with the real robot. It is compared to the gait designed by hand upon the kinematic description of the robot. The obtained movement is shown in Fig. 17(a), while Fig. 17(b) shows the comparable gait designed by hand.

The kinematic gait transports the robot's trunk at the same height over the ground. In contrast, the gait through evolution generates a movement which maximizes the average speed of the robot. When the robot uses this pattern, its body height over the ground varies. The states shown in Fig. 17(b) (frames 3B and 7B) that are necessary only to preserve robot body height over the terrain are not present at all in the evolved gait. The evolved solution excluded a situation where the even legs wait until the odd legs are put on the ground and vice versa. This difference becomes evident by comparing the gait diagrams of the hand-designed walking pattern and the evolved gait (Figs. 4 and 13, respectively). The evolved gait is a strict tripod, with the movement of two sets of three legs each repeated periodically, while in the standard tripod-like gait of Ragno, designed by hand, there are control steps (shown with a slanted background in Fig. 4) at which all six legs are placed on the ground, which makes this gait more cautious, but slower.

After identifying the simulation model, as described in Section 6, the validation experiments were repeated. The sixth simulation was conducted again, with the discrete search space and the same rules, but this time with the optimized model parameters. The obtained gait was again a cyclic, fast tripod, but this time the cycle of the obtained gait consisted of two leg states only—the gait repeated states 1 and 4 continuously. The evolution of such a gait was possible because the optimized model exhibits more dynamic behaviour than the old one. The simulated robot achieved the speed of $0.144\ \mathrm{m/s}$. When the same gait was downloaded to the real robot's control program, it was able to walk with the speed of $0.137\ \mathrm{m/s}$. This way, we achieved our goals: a gait that is considerably (52 %) faster than the best tripod-like gait we were able to program by hand, and a small discrepancy in performance between the simulation and the real robot.

8. Conclusions

The aim of this research was to develop a system that would learn in the course of simulation gaits optimal for the assumed robot model and fitness function, but feasible for the physical robot. The paper makes the following contributions:

- It demonstrates how the use of a very few, biologically inspired rules enables a genetic search technique to evolve more specific leg co-ordination patterns and leads to an effective gait.
- It shows in the simulations that the inter-leg dependencies—phase difference between contralateral legs, phase difference between neighboring legs, and periodic gait property, closely resembling the insect walking rules—emerge automatically in the proposed gait controller.
- It demonstrates that a dynamics-based walking robot simulator provides an opportunity to evolve dynamic gaits, which cannot emerge in a minimal simulation.
- It proposes a walking robot simulator with integrated robot model identification capabilities and an evolutionary identification procedure for this model.

The proposed approach proved to be effective, as was shown in the simulations, where a stable, cyclic, and fast gait was ultimately obtained. Given these results, we consider this approach a promising one for the evolution of complicated robot behaviour that requires handling large search spaces. As the next step, we will validate this approach using our new, larger and more autonomous hexapod robot Messor.

Acknowledgment

This research was supported by the Polish Ministry of Science and Higher Education under Grant No. N514 294635 for the years 2008–2010, which is gratefully acknowledged. The authors extend their thanks to the anonymous reviewers for their constructive criticism that improved the presentation of this paper.

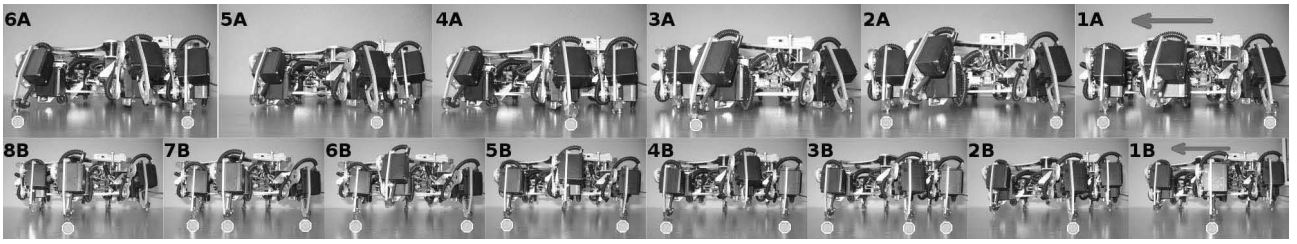

Fig. 17. Obtained evolutionary pattern (a) and gait designed by hand (b).

References

Albiez, J. and Berns, K. (2004). Biological inspired walking—How much nature do we need?, *in* M. A. Armada and P. de González Santos (Eds), *Climbing and Walking Robots. Proceedings of the 7th International Conference CLAWAR 2004*, Springer, Berlin, pp. 357–364.

Annunziato, M. and Pizzuti, S. (2000). Adaptive parameterization of evolutionary algorithms driven by reproduction and competition, *Proceedings of the European Symposium on Intelligent Techniques (ESIT 2000), Aachen, Germany*, pp. 31–35.

Arabas, J. (2001). *Lectures on Evolutionary Algorithms*, WNT, Warsaw, (in Polish).

Bäck, T., Hoffmeister, F. and H.-P. Schwefel (1991). A survey of evolution strategies, *in* R. K. Belew and L. B. Booker (Eds), *Proceedings of the 4th International Conference on Genetic Algorithms, Morgan Kaufmann, San Francisco, CA*, pp. 2–9.

Barfoot, T. D., Earon, E. J. P. and D'Eleuterio, G. M. T. (2006). Experiments in learning distributed control for a hexapod robot, *Robotics and Autonomous Systems* **54**(10): 864–872.

Beer, R. D., Quinn, R. D., Chiel, H. J. and Ritzmann, R. E. (1997). Biologically inspired approaches to robotics: What can we learn from insects?, *Communications of the ACM* **40**(3): 31–38.

Belter, D., Kasiński, A. and Skrzypczyński, P. (2008). Evolving feasible gaits for a hexapod robot by reducing the space of possible solutions, *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, Nice, France*, pp. 2673–2678.

Belter, D. and Skrzypczyński, P. (2009). Population based methods for identification and optimization of a walking robot model, *in* K. Kozlowski (Ed.), *Robot Motion and Control 2009*, Lecture Notes in Control and Information Sciences, Vol. 396, Springer, Berlin, pp. 185–195.

Busch, J., Ziegler, J., Aue, C., Ross, A., Sawitzki, D. and Banzhaf, W. (2002). Automatic generation of control programs for walking robots using genetic programming, *in* J. Foster, E. Lutton, J. Miller, C. Ryan and A. Tettamanzi (Eds), *Genetic Programming, Proceedings of the 5th European Conference EuroGP 2002*, Lecture Notes in Computer Science, Vol. 2278, Springer, Berlin, pp. 258–267.

Chernova, S. and Veloso, M. (2004). An evolutionary approach to gait learning for four-legged robots, *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, New Orleans, LA, USA*, pp. 2562–2567.

Dorigo, M. and Colombetti, M. (1997). *Robot Shaping: An Experiment in Behavior Engineering*, MIT Press, Cambridge, MA.

Figliolini, G., Stan, S.-D. and Rea, P. (2007). Motion analysis of the leg tip of a six-legged walking robot, *Proceedings of the 12th IFToMM World Congress, Besançon, France*, (on CD-ROM).

Fukuoka, Y., Kimura, H. and Cohen, A. H. (2003). Adaptive dynamic walking of a quadruped robot on irregular terrain based on biological concepts, *International Journal on Robotics Research* **22**(4): 187–202.

Gallagher, J., Beer, D. R., Espenschied, K. and Quinn, R. D. (1996). Application of evolved locomotion controllers to a hexapod robot, *Robotics and Autonomous Systems* **19**(1): 95–103.

Goldberg, D. (1989). *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison-Wesley, Reading, MA.

Holland, J. (1975). *Adaptation in Natural and Artificial Systems*, University of Michigan Press, Ann Arbor, MI.

Hornby, G., Takamura, S., Yamamoto, T. and Fujita, M. (2005). Autonomous evolution of dynamic gaits with two quadruped robots, *IEEE Transactions on Robotics* **21**(3): 402–410.

Huber, M. and Grupen, R. A. (1997). A feedback control structure for on-line learning tasks, *Robotics and Autonomous Systems* **22**(3–4): 303–315.

Jakobi, N. (1998). Running across the reality gap: Octopod locomotion evolved in a minimal simulation, *in* P. Husbands and J.-A. Meyer (Eds), *Evolutionary Robotics. Proceedings of the First European Workshop EvoRobot98*, Lecture Notes in Computer Science, Vol. 1468, Springer, Berlin, pp. 39–58.

Jakobi, N., Husbands, P. and Harvey, I. (1995). Noise and the reality gap: The use of simulation in evolutionary robotics, *Proceedings of the 3rd European Conference on Articial Life (ECAL'95), Granada, Spain*, pp. 704–720.

Kimura, H., Yamashita, T. and Kobayashi, S. (2001). Reinforcement learning of walking behavior for a four-legged robot, *Proceedings of the IEEE Conference on Decisions and Control, Orlando, FL, USA*, pp. 411–416.

Kirchner, F. (1998). Q-learning of complex behaviours on a six-legged walking machine, *Robotics and Autonomous Systems* **25**(3–4): 256–263.

Kowalczuk, Z. and Białaszewski, T. (2006). Niching mechanisms in evolutionary computations, *International Journal of Applied Mathematics and Computer Science* **16**(1): 59–84.

Kozlowski, K. (1998). *Modelling and Identification in Robotics*, Springer, Berlin.

Kumar, V. R. and Waldron, K. J. (1989). Adaptive gait control for a walking robot, *Journal of Robotic Systems* **6**(1): 49–76.

Lewis, M., Fagg, A. and Bekey, G. (1994). Genetic algorithms for gait synthesis in a hexapod robot, *in* Y. Zheng (Ed.), *Recent Trends in Mobile Robots*, World Scientific, Singapore, pp. 317–331.

Luk, B. L., Galt, S. and Chen, S. (2001). Using genetic algorithms to establish efficient walking gaits for an eight-legged robot, *International Journal of Systems Science* **32**(6): 703–713.

Maes, P. and Brooks, R. A. (1990). Learning to coordinate behaviors, *Proceedings of the 8th National Conference on Artificial Intelligence (AAAI 1990), Boston, MA, USA,* pp. 796–802.

Mataric, M. and Cliff, D. (1996). Challenges in evolving controllers for physical robots, *Robotics and Autonomous Systems* **19**(1): 67–83.

Parker, G. B. and Mills, J. W. (1999). Adaptive hexapod gait control using anytime learning with fitness biasing, *Proceedings of the Genetic and Evolutionary Computation Conference, Orlando, FL, USA*, pp. 519–524.

Perry, M. J., Koh, C. G. and Choo, Y. S. (2006). Modified genetic algorithm strategy for structural identification, *Automatica* **84**(8–9): 529–540.

Ridderström, C. (1999). Legged locomotion control—A literature survey, *Technical Report TRITA-MMK 1999:27*, Royal Institute of Technology, Stockholm.

Ritzmann, R. E., Quinn, R. D. and Fischer, M. C. (2004). Convergent evolution and locomotion through complex terrain by insects, vertebrates and robots, *Arthropod Structure & Development* **33**(3): 361–379.

Skrzypczyński, P. (2004a). Experimental validation of the fuzzy reactive behaviours evolved in simulation, *in* F. Groen, N. Amato, A. Bonarini, E. Yoshida and B. Kröse (Eds), *Intelligent Autonomous Systems 8*, IOS Press, Amsterdam, pp. 464–471.

Skrzypczyński, P. (2004b). Shaping in a realistic simulation: An approach to learn reactive fuzzy rules, *Preprints of the 5th IFAC/EURON Symposium on Intelligent Autonomous Vehicles, Lisbon, Portugal,* (on CD-ROM).

Smith, R. (2007). Open dynamics engine, `http://www.ode.org`.

Song, S.-M. and Waldron, K. J. (1989). *Machines that Walk: The Adaptive Suspension Vehicle*, MIT Press, Cambridge, MA.

Svinin, M. M., Yamada, K. and Ueda, K. (2001). Emergent synthesis of motion patterns for locomotion robots, *Artificial Intelligence in Engineering* **15**(4): 353–363.

Tuyls, K., Maes, S. and Manderick, B. (2003). Reinforcement learning in large state spaces: Simulated robotic soccer as a testbed, *RoboCup 2002: Robot Soccer World Cup VI*, Lecture Notes in Computer Science, Vol. 2752, Springer, Berlin, pp. 319–326.

Walas, K., Belter, D. and Kasiński, A. (2008). Control and environment sensing system for a six-legged robot, *Journal of Automation, Mobile Robotics and Intelligent Systems* **2**(3): 26–31.

Walker, J., Garrett, S. and Wilson, M. (2003). Evolving controllers for real robots: A survey of the literature, *Adaptive Behavior* **11**(3): 179–203.

Wilson, D. M. (1966). Insect walking, *Annaul Reiew of Entomology* **11**(1): 103–122.

Yang, J.-M. (2009). Fault-tolerant gait planning for a hexapod robot walking over rough terrain, *Journal of Intelligent and Robotic Systems* **54**(4): 613–627.

Zagal, J. C., Ruiz-del-Solar, J. and Vallejos, P. (2004). Back to reality: Crossing the reality gap in evolutionary robotics, *Preprints of the 5th IFAC/EURON Symposium on Intelligent Autonomous Vehicles, Lisbon, Portugal,* (on CD-ROM).

Dominik Belter received the M.Sc. degree in control engineering and robotics from the Poznań University of Technology in 2007. Since then he has been pursuing his Ph.D. in robotics, working as a research assistant at the Institute of Control and Information Engineering, Poznań University of Technology. His research interests include the control of walking robots, machine learning, and soft computing.

.

Piotr Skrzypczyński graduated from the Poznań University of Technology (1993). He received the Ph.D. and D.Sc. degrees in robotics from the same University in 1997 and 2007, respectively. Since 1998 he has been an assistant professor at the Institute of Control and Information Engineering (ICIE) of the Poznań University of Technology, and the head of the Mobile Robotics Laboratory of the ICIE. Dr. Skrzypczyński is the author or co-author of over 90 technical papers in the fields of robotics and computer science. His current research interests include autonomous mobile robots, navigation, multisensor fusion, distributed robotic systems, and computational intelligence methods in robotics.

Received: 16 January 2009
Revised: 20 July 2009