Evolution of Robot Leg Movements in a Physical Simulation

Jens Ziegler, Wolfgang Banzhaf
University of Dortmund, Dept. of Computer Science, D-44227 Dortmund, Germany

Abstract

This paper introduces a Genetic Programming approach to creating patterns of movements for legs of walking robots. It uses a physics-based simulation system to evaluate the fitness of movement patterns, which are emerging from the interpretation of the individuals of the population. Different methods are shown that increase the speed of the evolution by several orders of magnitude.

1 INTRODUCTION

The evolution of robot control programs has been the topic of recent publications (1, 2, 3, 4, 5, 6, 7, 8) and especially the field of walking robots becomes more and more important. Gait patterns of stick insects have been analyzed to gain more detailed information on natural gait coordination Algorithms (9). Many researchers have often been inspired by biology to build legged robots. An overview can be found in (10). Nevertheless, the above-mentioned approaches deal with a special instance of an autonomous robot (or walking agent), on which the architecture of the developed control system heavily depends.

This paper follows a more general approach by abstracting from real robots and their often brittle and time consuming hardware. Instead, a simulated model of arbitrary robot morphologies is used which takes the kinematics of the robot ---given by their geometric and drive specific constraints--- and environmental influence into account. Simulating walking robots allows more flexible architectures and rapid prototyping, which is, compared to experiments with real hardware, less expensive. The evolution of virtual agents in a simulated environment has been successfully demonstrated by Sims and Komosinski (11, 12, 13).

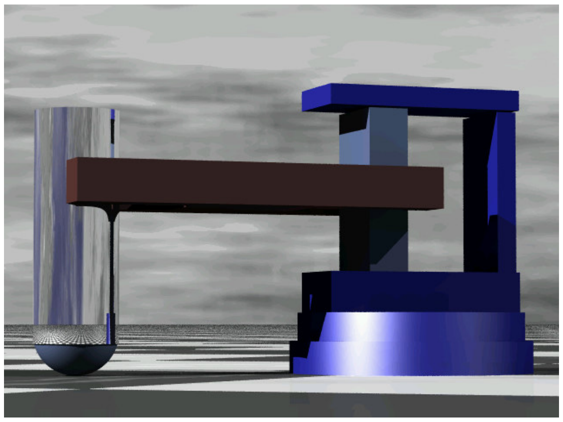
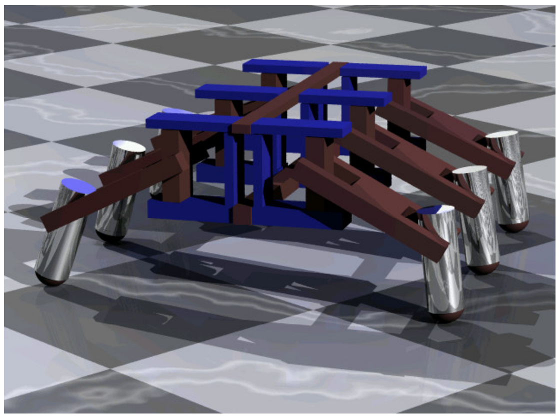
Fig. 1 Left: 3 DOF robot leg, consisting of simple geometric objects. Right: 18 DOF walking robot, whose legs are 'fuzzy' copies of the shown leg. Details see text.

This approach does not imply that the evolved controllers depend on specific information on the robot morphology such as certain lengths or distances. On the contrary, it was one of the main goals of this work to make the evolution of robot controllers as independent as possible from morphology specific information. So morphology-related information, although available, will not be used. If a robot controller uses information of the cinematic model to solve its control task ---e.g. to compute an inverse transformation in order to calculate the joint angles of a robot arm consisting of rotational joints from a given position of the TCP (*tool center point*) in world coordinates--- the correctness of the calculation depends on unchanged parameters of the hardware, either real or simulated. If, for example, a joint looses the ability to reach certain positions, the outcome of the inverse transformation, which depends on the correct working joint, is useless. Additionally, the algorithm for the inverse transformation is correct only for a single robot. A machine-learning algorithm should be able to cope with changing circumstances and recover the former quality.

A first step toward control of movements of a legged robot is to move the single joints according to a desired trajectory. This trajectory depends on the desired behavior (e.g. swing or stance phase of the leg) and requires very well coordinated synchronous movements of all joints involved. The movements of joints are caused by actuators that, basically, apply a force to one or more connected rigid bodies. Thus, describing a movement of a leg of a robot requires to give the time dependent values of the acting forces for each involved joint during the motion. The next sections shall now explain the experimental setup and results from the evolution of leg movements with Genetic Programming.

2 THE PHYSICAL SIMULATION SYSTEM

In order to avoid time consuming experiments with real robots and in order to have the possibility of easy robot morphology configuration, a physical simulation system is used for all experiments. We used the freely available tool AERO[1] as the physical simulation library with our own augmentation for

[1] An introduction to AERO is available via http://www.informatik.uni-stuttgart.de/ipvr/bv/aero/aero.html, the original source code can be downloaded via http://www.ee.uwa.edu.au/~braunl/aero/ftp/. Please note that the original code was modified and expanded.

parameter	range of values
joint no.	integer [0;3]
start	real [0;t]
duration	real [0;t]
force	real [–max;max]

Fig. 1 A single Instruction of the Genetic Programming System

handling of populations, the evaluation of individuals, the variation and selection operators, and the formulation of fitness functions.

2.1 Physics Based Simulation of Legged Robots

The experiments employed regular robot morphologies. These are symmetric multi-legged robots with identical geometry for each leg. A six-legged robot and a detailed view of one of its legs are shown in fig. 1. To be a bit more realistic, our models consist of six variations of a leg mounted on a simple body, so that small differences in the robots morphology mirror the tolerances in the manufacturing process of real robots. The simulation system calculates all variables in the model using gravitation, inertia tensors, torques, accelerations, speed and cinematic constraints (links or joints) -including all motive forces generated by the actuators- to generate a set of differential equations describing the movement of all bodies which are numerically integrated and updated (14).

Coordinating the movement results in the necessity to give a time series of motive forces for each joint of the leg, which sums up to 3 joints for a single leg and 18 joints for a six-legged walker. Most robot languages encapsulate the necessary motive forces and require only a nominal value for either angle or translation length. In the simulator used, however, it is necessary to give the exact amount, direction, time span, and working point of all motive forces. This has some consequences for the layout of the genetic programming system.

3 THE GENETIC PROGRAMMING SYSTEM

Controlling the movement of a robot leg requires a sequence of instructions for each joint of the leg. This sequence has to be coordinated in time to achieve the desired movement in sufficient quality.

3.1 Representation

An individual in the GP system can now be seen as a sequence of instructions of variable length as shown in fig. (2). The order of the sequences in the individual is not important, because the instructions are sorted according to their starting time. The simulator includes the force of each instruction in the set of differential equations exactly at the time the force affects the particular joint. The individuals, consisting of many instructions, are varied with the following operators:

3.1.1 Mutation

- Micro-Mutation
  This operator varies the genome only in one single elementary instruction. It changes one of the four elements of an instruction randomly by adding a Gaussian distributed

random variable with expected value x = 0.0 and a standard variance of $\sigma = 1.0$. The operator ensures valid values.

- Macro-Mutation
  This operator either deletes a single instruction randomly with a uniformly distributed probability for each instruction, or it inserts a totally random instruction. Each element of this new instruction is chosen from a uniform distribution out of the valid range of values.

3.1.2 Crossover

- Micro-Crossover
  Exactly one element of a single instruction is exchanged between two individuals. Which individuals and which parameters are modified is determined randomly with a uniform distributed probability for each individual/parameter.
- Macro-Crossover
  Here, single instructions are exchanged. Which instruction is determined with a uniform distributed probability for each instruction.
- Homologue-Crossover
  This operator exchanges instructions which move the same joint. The entire instruction is exchanged. The instructions are chosen randomly from all valid instructions with a uniform distribution. If no such pair exists, nothing happens.

3.1.3 Fitness

The quality of a movement can be measured as follows,

$$fitness = \int_{0}^{t_{end}} (x(t) - x_s(t))^2 dt \qquad (1)$$

which equals a sum, due to the fact that the time increases with discrete steps in the numerical integration,

$$fitness = \sum_{t=0}^{t_{end}} (x(t) - x_s(t))^2 \qquad (2)$$

The fitness of an individual is the sum of squared differences to an a priori described ideal trajectory. In the following experiments, the TCP of the leg should move linearly from a starting point to a target point where it is supposed to stop. The faster and closer an individual follows the trajectory, the better is its fitness. The trajectory in our experiments is a linear connection of two points. The robot leg is supposed to accelerate, to move along the trajectory, and to decelerate until full stop at the target point. To ensure all phases of the movement, the model was simulated until $t_{end} = 3$ s, long enough for a very slow moving leg to complete the movement. The main parameters of the GP run are shown in tab. 1.

Table 1 Koza tableau with parameter settings for the GP system

Parameters	**Values**
Objective	Evolve movement of leg along a given trajectory
Terminal Set	Integer and real values
Operator Set	Instruction of type shown in fig. 2
Selection Scheme	Subset tournament, size = 4
Population size	600

Crossover probability	0.3 for each variant, adaptive in later experiments
Mutation probability	0.45 for each variant, adaptive in later experiments
Termination criterion	Max. No. of tournaments
Maximum length of programs	Unlimited
Initialisation method	Random init

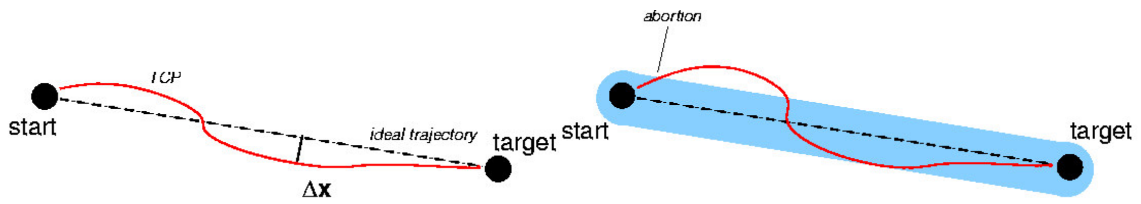

Fig. 3 Left: Distance between ideal trajectory and TCP movement. Right: Premature termination of an evaluation due to exceeding the quality threshold.

4 SPEEDING UP EVOLUTION

Evaluating an individual in a physical simulation takes time. In order to speed up the evolution, parallel evaluation of a subset of the population was implemented. Therefore, the execution of the simulator evaluating the individual is spread over a set of 50 Sun Sparc Ultra workstations, so that every workstation processes one single individual. The fitness values are sent back to a central server, which queues the results in a variable length set. If the queue is full, participants of a tournament are selected from the queue. The offspring individuals are in turn evaluated in parallel. The size of the queue can then be used to configure the algorithm: if the queue is small, the resulting selection scheme is a combination of elitist and tournament selection. If the queue has the size of the population, a pure tournament selection is realized. Experiments showed that a queue size of 80% of the population leads to a good convergence speed. This parallelization reduces the time for an average run (typical parameters see tab. 1) by a factor of 10 to 15 depending on the actual workload of the cluster. It is remarkable that this parallel implementation is robust against possible loss of individuals (provided that the required queue size is less than the population size) and also works within heterogeneous computer networks: Faster computers return their results earlier, slower computers a little later. This has only the effect of increasing the diversity of the population due to delays in evaluations.

4.1 Optimizing Convergence by Premature Termination

The fitness of a movement depends on the correct coordination of joint torques. If in the beginning of a movement a single joint does not fit into the movement pattern of the leg, then the overall quality of the movement will be low, because even very good coordinated joint movements afterwards will not be able to correct the early error. It is therefore possible to decide in an early stage of the evaluation process whether to terminate the simulation of the individual due to the irrevocable error (see fig. 3). This method leads to remarkable saving in simulation time and results in a speedup of the whole algorithm. See fig. 4 for a comparison. The method of premature termination requires checking the development of the movement from time to time. Continuously checking the quality of movements results in an average speedup of 250%.

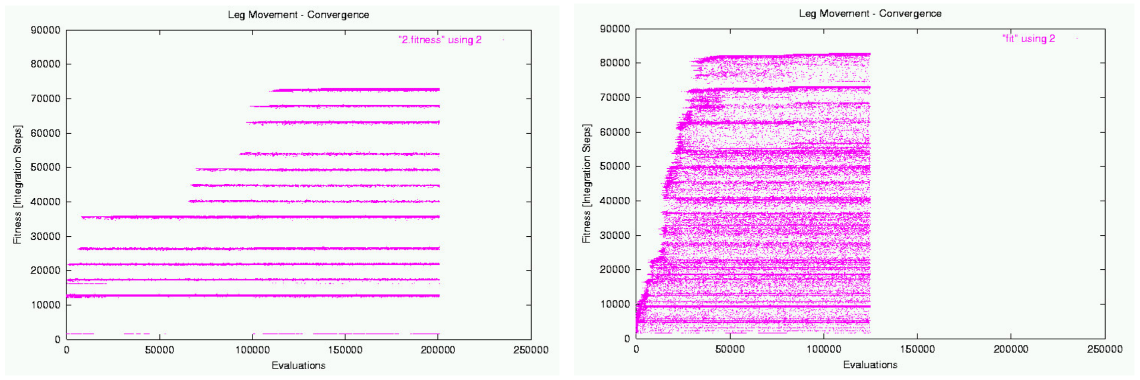

Fig. 4 Left: Experiment with quality control every 1/15 s. Right: Experiment with permanent quality control.

It is remarkable that the evolution is faster at the beginning of the simulation because there are more individuals with a worse fitness whose evaluation is prematurely terminated. Towards the end of the run, when the population has a higher fitness on average, the evaluation of each individual is extended and requires more integration steps, possibly until the end of the evaluation time. As a result, the speed of the whole algorithm decreases during the run. Ongoing experiments investigate the effects of adapting the size of the "fitness channel", i.e. the maximum allowed deviation, during the evolution. With this method, the fraction of prematurely terminated individuals might be held at an almost constant level with the effect that the average time for evaluation may increase less fast, resulting in an additional increase of convergence speed.

4.1 Using a B-Spline Representation

The representation of the experiments shown has a major drawback: the GP system is forced to approximate a probably continuous curve of forces for each joint with discrete and variably long intervals of constant acting forces. The joints may have to accelerate, to move with constant speed, to decelerate, and to stop according to the desired movement of the whole leg. A new and straightforward way of representing continuous force curves with boundary conditions are B-Splines. Splines are piecewise polynomial functions defined by a set of discrete points (control points). There exist a variety of spline functions, but B-splines have some properties, which make them most suitable:

- Locality
  B-spline functions are defined piecewise, and each piece only depends on a limited number of neighboring control points. A global spline function depends at all places on all the control values. A small change at one control point thus changes the function everywhere.
- Smoothing
  B-splines do not interpolate their control points. Instead, the function is smoothed with respect to a linear interpolation of the control points, which gives these splines the benefit of increased continuity.

Locality has influence on the effects of the genetic variation operators, e.g. a point mutation of a single control points does not change the spline everywhere. The smooth shape of the curve has also positive effects on the integration of the differential equations, because sharp transitions cause a decreasing step size of the numerical integration, which in turn slows down the evaluation of an individual. The curves of forces for all three joints of a good individual are shown in fig..6 (left). The algorithm to create the splines is to create partial curves of cubic order.

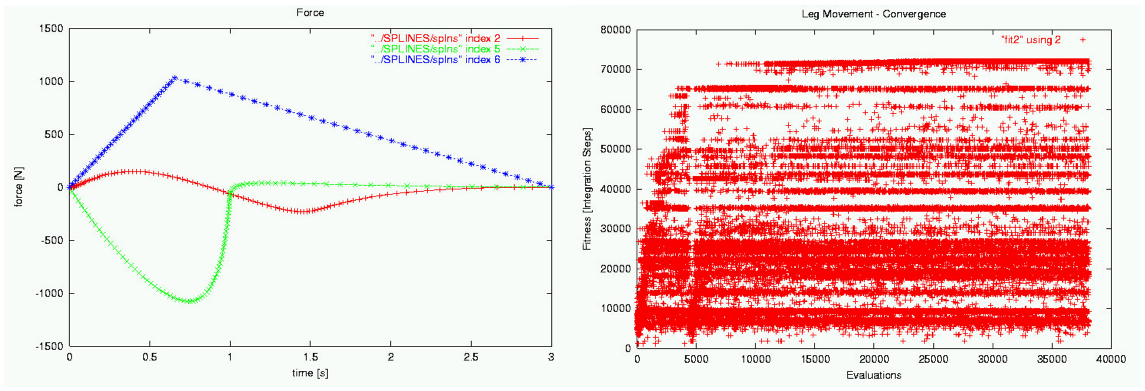

Fig. 6 Left: Force development for each joint. Right: Convergence with new representation. The algorithm converges to a fitness equivalent of the algorith im fig. 4 but with only 10000 evaluations

If the number of control points is small, then the partial curves are of order two ore one. Each point in the curves depends only on the two points bounding the current interval and their direct neighbors. Every joint has two default control points: $t_0 = 0$ and $t_{end} = 0$, to ensure a continuous starting and stopping of the respective movements. If four control points $s_0,\ldots,s_3$ are given, the uniform cubic B-spline function segment between s_1 and s_2, parameterized by t=0,…,1, is

$$f(t) = s_0 b_0(t) + s_1 b_1(t) + s_2 b_2(t) + s_3 b_3(t) \qquad (3)$$

with the following weight functions

$$\begin{aligned} b_0(t) &= -\frac{t^3}{6} + \frac{t^2}{2} - \frac{t}{2} + \frac{1}{6} \\ b_1(t) &= \frac{t^3}{2} - t^2 + \frac{2}{3} \\ b_2(t) &= -\frac{t^3}{6} + \frac{t^2}{2} + \frac{t}{2} + \frac{1}{6} \\ b_3(t) &= -\frac{t^3}{6} \end{aligned} \qquad (4)$$

The uniform cubic B-spline is now a simple sum of the appropriate of these functions times the value at the sample. Fig. 6 shows three B-splines, one for each joint, of the acting forces during the simulation. The splines are of different order, depending on the number of control points. The control points are now encoded in the structure of an individual. The number and values of the points (see fig.. 5) are variable and change during the evolutionary process, until the movements caused by the acting forces described by interpolated force curves based on these control points fit the desired trajectory in an optimal way. The new representation does not change the behavior of the variation operators at all. Preliminary results of the convergence are shown in fig. 6. Why the convergence speed of the spline algorithm is again faster compared to experiments with the old representation needs further analysis and more experiments for statistically significant statements.

5 DISCUSSION AND OUTLOOK

This paper shows a new approach to the evolution of patterns of movement with Genetic Programming. It uses different representations and methods to improve the performance of the algorithm. Preliminary results show that B-spline representation and premature termination along with a robust parallel implementation cause a remarkable speed up of the algorithm. The results need to be further investigated; particularly the effect of self-adaptive control of quality criteria needs more experiments.

ACKNOWLEDGMENTS

This project is supported by the DFG (Deutsche Forschungsgemeinschaft), under grant Ba 1042/6-1.

REFERENCES

1. J. R. Koza. *Genetic Programming*. MIT Press, Cambridge, MA, 1992.
2. P. J. Angeline. Genetic programming and emergent intelligence. In K. E. Kinnear, Jr., editor, *Advances in Genetic Programming,* chapter~4, pages 75--98. MIT Press, 1994
3. W. Banzhaf, P. Nordin, R. E. Keller, and F. D. Francone. *Genetic Programming -- An Introduction; On the Automatic Evolution of Computer Programs and its Applications*. Morgan Kaufmann, dpunkt.verlag, January 1998
4. P. Dittrich, A. Bürgel, and W. Banzhaf. Learning to control a robot with random morphology. *In Proceedings Evo-Robot-98, P. Husbands and J.-A. Meyer, Eds*., pages 165--178, 1998.
5. M. Olmer, W. Banzhaf, and P. Nordin. Evolving real-time behavior modules for a real robot with genetic programming. In *Proceedings of the international symposium on robotics and manufacturing*, Montpellier, France, May 1996.
6. M. A. Lewis, A. H. Fagg, and A Solidum. Genetic programming approach to the construction of a neural network control of a walking robot. *Proceedings of the 1992 IEEE International Conference on Robotics and Automation*, pages 2618--2623, Nice, France, May 1992
7. G. F. Spencer. Automatic generation of programs for crawling and walking. In S. Forrest, editor, *Proceedings of the 5th International Conference on Genetic Algorithms, ICGA-93*, page 654, University of Illinois at Urbana-Champaign, 17-21 July 1993. Morgan Kaufmann
8. L. Gritz and J. K. Hahn. Genetic programming evolution of controllers for 3-D character animation. In J. R. Koza, K. Deb, M. Dorigo, D. B. Fogel, M. Garzon, H. Iba, and R. L. Riolo, editors, *Genetic Programming 1997: Proceedings of the Second Annual Conference*, pages 139--146, Stanford University, CA, USA, 13-16 July 1997. Morgan Kaufmann.
9. H. Cruse. Coordination of leg movement in walking animals. *In From animals to animats. Intl. Conf. on Simulation of Adaptive Behavior*, pages 105--119, 1991
10. K. Kleiner. Look to the insect*. New Scientist, No. 1951, 12 Nov. 1994*, 144:27--29, 1994
11. M. Komosinski and S. Ulatowski. Framsticks - Artificial Life. In *ECML '98 Demonstration and Poster Papers, Chemnitzer Informatik Berichte* , pages 7--9, 1998.
12. K.Sims. Interactive evolution of dynamical systems. In F. J. Varela and P. Bourgine, editors*, Toward a Practice of Autonomous Systems: Proceedings of the First European Conference on Artificial Life*, pages 171--178, Paris, France, 11-13 December 1992. MIT Press.
13. K. Sims. Evolving virtual creatures. In A. Glassner, editor, *Proceedings of SIGGRAPH '94 (Orlando, Florida, July 24--29, 1994*), Computer Graphics Proceedings, Annual Conference Series, pages 15--22. ACM SIGGRAPH, ACM Press, July 1994.
14. R. E. Roberson and R. Schwertassek. *Dynamics of Multibody Systems*. Springer Berlin, 1988