In *Proceedings of ICES2003*
Trondheim, Norway, 17th - 20th March 2003

Co-Evolving Complex Robot Behavior

Esben H. Østergaard and Henrik H. Lund

The Maersk McKinney Moller Institute for Production Technology
University of Southern Denmark
Campusvej 55, 5230 Odense M., Denmark
esben|hhl@mip.sdu.dk

Abstract. Reports on evolutionary robotics systems have so far been on evolving controllers that make simple robots do simple tasks in simple environments. In this paper we try to stress the evolutionary robotics approach by evolving a controller for a more complex task, namely Khepera robot soccer, and evaluate evolved controller performance against hand-coded controllers. We present a system that uses competitive co-evolution to develop robot controllers for the task. The system is described, and performance of the system is documented. Co-evolution is tested against single-population evolution, and it is concluded that co-evolution has the ability to produce more robust individuals with respect to opponent strategies.

1 Introduction

Many of the evolutionary robotics systems that have been reported so far, has documented successful generation of robot controllers for simple robots performing simple tasks in simple environments. In this paper we will try to go one step further and apply the method to a more complex and competitive task, namely the Khepera Robot Soccer task. Using this task as a testbed has two advantages: 1) No known optimal strategy exists. 2) Real-world competitions exist, so that evolved controllers can be tested against hand-coded controllers in a neutral competitive scenario. Given the competitive nature of the task, the task lends itself to artificial co-evolution. The emphasis of this paper is on co-evolution applied to this task, and on the viability of the evolutionary robotics approach in general.

1.1 The Khepera Robot Soccer Task

In a Khepera robot soccer match two Khepera robots are pitted against each other in an arena. The arena is 105cm long and 68cm wide with cut off corners, as shown in figure 1. The Khepera robot is a widely used miniature battery-driven robot with on-board processing capabilities from K-Team. The Khepera used for soccer playing is equipped with 8 IR-sensors, wheel encoders and a camera.

A match consists of five rounds. A round lasts until a goal is scored, until the ball has not moved for thirty seconds, or until four minutes have elapsed. At

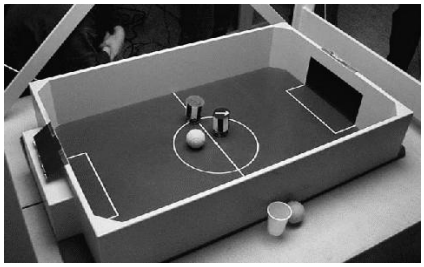

Fig. 1. *A picture of the football arena.*

the beginning of each round, the players are positioned on random symmetric starting points. The ball starts at the center of the pitch. Each player starts facing the opponent's goal line at random, symmetric starting positions.

2 Approach

We decided to use an evolutionary robotics approach, and to construct a simulator to speed up evolution. The implemented system is shown in figure 2. After a pre-specified number of generations in the simulator, the best performing controller is translated into a binary file that can be executed directly on a Khepera robot. The following sections will go into more detail about each of the components of the system.

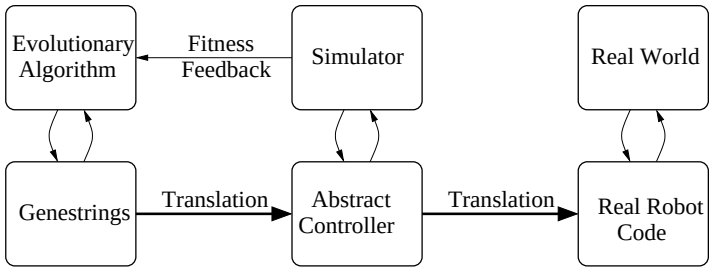

Fig. 2. *The implemented system is an evolutionary robotics system using a simulator to speed up evolution.*

3 Co-evolution

Since D. Hills, [Hil92], showed that better sorting networks could be produced when using co-evolution, co-evolutionary methods have spread to other fields of research where Evolutionary Algorithms are being used, [AP93], [CM96] and

[BLP96]. The task of evolving a robot football player intuitively lends itself to co-evolutionary methods because of the competitive nature of the problem. Given a specific opponent, it might be possible to find the best strategy, but there is no single strategy that is best against all opponents. The task is thus to find a player that performs well against a large number of opponent strategies.

An attempt to illustrate competitive co-evolution for two populations can be seen in figure 3. When one population (A) finds a good locally optimal strategy, the competing population (B) is encouraged to find a solution that performs especially well against the other populations strategy. This changes the fitness landscape for population (A), forcing evolution to find another solution to the problem.

The effect of one population changing the fitness landscape of the other population is called the *Red Queen Effect*, described in [CM95]. The Red Queen Effect has been examined by co-evolving pursuit and evasion behavior, both in simulation [CM96] and on real robots [FNM01]. In [FNM98] and [FN98], D. Floreano et al. investigate the dynamics of competitive co-evolutionary systems, and show that the Red Queen Effect can give rise to "cycles" between alternative classes of strategies and can make it difficult to monitor evolutionary progress. They also state that co-evolution can be used with benefit when the task is such, that the best strategy of one population is dependent on the strategy of the other population.

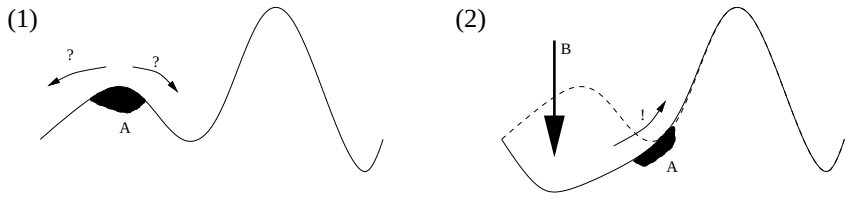

Fig. 3. *Illustration of the idea behind co-evolution.*

3.1 The evolutionary algorithm

The co-evolutionary algorithm was implemented as two separate GA's. There is no exchange of gene material between the two populations. After much trial and error, good evolution parameters were found to be a population size of 50, an elite of 10, and these 10 were also used for reproduction. The probability of mutation was 5% per gene (byte), and there was no cross-over.

For evaluation, individuals played 25 simulated rounds against individuals from the opponent population (every second player, alternating). The following fitness fuction was used;

– 10.000 per goal, -10.000 for own goal[1].

[1] The player getting (mis)credit for the goal, is the player that most recently touched the ball.

- 2.5 per second remaining of the round when scored a goal.
- -0.01 per cm to the ball when the round ends.

The fitness for one round was the sum of the three components. The first component dominates the fitness. The second and third components are only relevant in cases where an equal number of goals have been scored. The second component favors quick goals, and the third component favors robots that stay close to the ball. The third component is meant for bootstrapping the evolutionary process.

Design of the fitness function was a guess based on intuition. It turned out to serve it's purpose, and no further experimentations were performed.

4 The Simulator

A central issue in any evolutionary robotics systems is the evaluation of candidate controllers during evolution. It was decided to perform the evaluations in a simulation modelling the Khperea soccer world for a number of practical reasons. Using a table based model [MNL95] was rejected due to the large state-space of the world. Instead a hybrid between the minimal simulation approach, suggested by Nick Jacobi [Jac98a], and a geometric model is used. Basically everything that easily could be modeled geometrical is modeled geometrically, and then the rest is made *unreliable* by adding noise[2].

The simulator was constructed using iterative refinements, where each version of the simulator was tested by observing differences in the simulated robot behavior and the real world robot behavior.

4.1 The Use of Minimal Simulation

The interaction between the different world objects in the soccer arena is very difficult to model accurately [Smi98]. A minimal simulation approach could be to make the contact between world objects unreliable for the robot controllers, as described in [Jac98b]. By making the simulated environment more unreliable than the real environment, evolved controllers would be forced to be robust, thus increasing the chance that the controller will perform well when transferred to a real robot. In the simulation, interaction between world objects is split up into cases, and specialized code deals with each specific case. For each cases, interaction was modeled according to observation as closely as possible and then noise was applied both to robot and ball movement.

As stated in [HCH92] and [MC96], producing simulations of visual sensing is a very time-consuming task, both for the programmer when building the simulator, and for the computer during simulation. Instead of simulating the Khepera camera pixel by pixel, the output from the pre-processors were simulated.

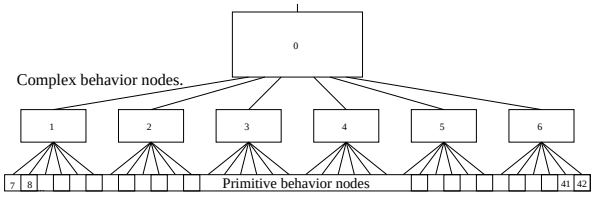

Fig. 4. *Architecture of the controller. A tree of behavior modules.*

5 Behavior Representation

The robot controller is a fixed-structure tree of arbitrators, as shown in figure 4. Modules 0-6 are arbitrator modules, and modules 7-42 are primitive actions. At each 100 ms time step, control is propagated down through the tree to one of the 36 primitives, which then has full control of the robot for the duration of the time step. The architecture is based on the task decomposition approach described by W. Lee, J. Hallam and H. H. Lund [LHL97].

Two types of arbitrators were used, implementing sequential and reactive arbitration, both illustrated in figure 5. Both arbitrators uses a set of conditions to arbitrate between six sub modules, and transfer control to only one of them. The conditions are implemented as fixed size boolean expressions. Figure 7 shows the structure of the condition trees (a), and an example (b). The bottom four nodes of the condition trees can be constant values or current sensor reading of the robot.

The contents of the nodes in the behavior and condition trees was coded by the gene strings. A genotype is a string of 638 bytes, corresponding to a search space of about 10^{1500}. 36bytes code for the primitve actions, whereas the rest code for the conditions.

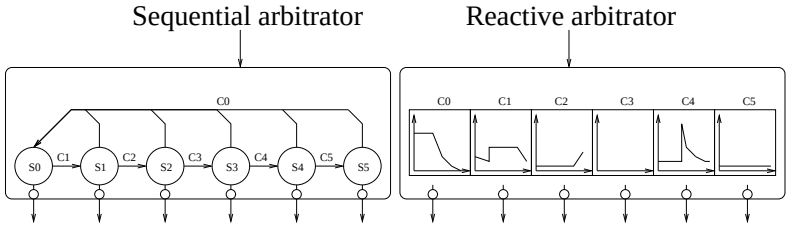

Fig. 5. *The two types of arbitrators. Control is propagated down to a sub module. In the sequential arbitrator, control is transferred to the sub module corresponding to the state of a finite state automaton. In the "reactive" arbitrator, control is transferred to the sub module corresponding to condition with the highest activation. The activation goes high when the condition is true, and then decays exponentially when the condition is false. The height and slope is determined by genes.*

[2] An applet can be seen at http://www.mip.sdu.dk/~esben/EvoRobSoc/applet/

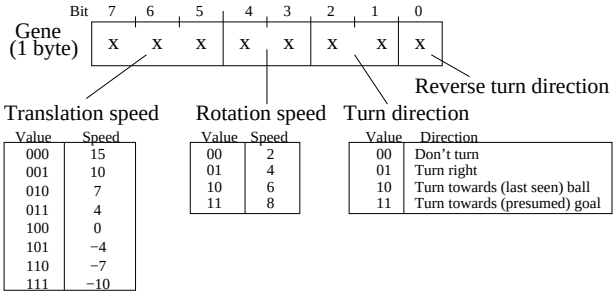

Fig. 6. *A gene coding for a primitive action. All speeds are in Khepera speed units. Rotation speed is the difference in speeds for the two motors.*

5.1 The behavior primitives

Generalizations of the behavior primitives used in a successful hand-coded robot football player are used as building blocks for the evolved controllers. The motor primitive are combinations of three parameters; translation speed, rotation speed, and rotation direction, as shown in figure 6. Note that for example action number $001-01-10-0$ will make the robot move forward while turning toward the ball.

The sensor primitive are output from preprocessors, not the raw Khepera sensor data.

- The proximity sensors, filtered so the returned value is the minimum of the last three measurements, and then paired two and two, as in [VPB99]
- The output of the image processing algorithm. The algorithm returns the centroid and width of the ball-blob, if the ball is in the image.
- A "stuck sensor" that uses the position counters to determine whether the robot is stuck or not.
- A sense of direction, that uses the incremental encoders to give an approximated heading of the robot.

These motor and sensor primitives are quite high level compared to what is described in other systems in the studied literature. M. Matarić and D. Cliff wrote: "This abstraction of representation allows for significantly reducing the search space and can greatly accelerate the evolutionary process" [MC96]. A counter argument is, that using higher level primitives reduces the search space so fewer potential solutions are available to evolution. Also, higher level primitives require more work for the programmer.

6 Tests & Results

The system was tested both on performance in the simulation, an on the performance in the real world. Also, a number of tests were performed to evaluate the use of co-evolution.

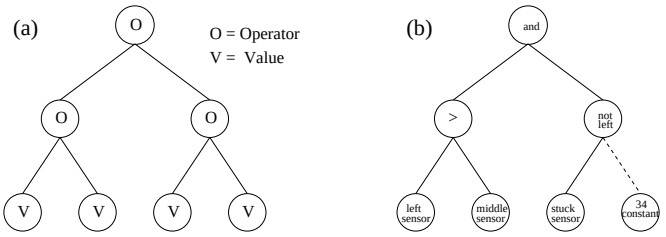

Fig. 7. *The structure of a condition. (a) the general structure, (b) an example.*

6.1 Behavior in Simulator

The first test was to test weather the given architecture and primitives were sufficient to make a simulated robots play soccer. Initial runs revealed that after about 250 generations the best individuals of both populations were able to score about 80% of the time in the simulator, when alone in the arena. This certainly shows that the set of primitives used is sufficiently expressive, but it also gives rise to some concern with respect to the realism of the simulator, since 80% scoring rate is very good, even for a hand coded robot.

Evolved Strategy Two distinct strategies for approaching the ball were observed in the evolved robots. The two types are shown in figure 8 (a) and (b). The figure shows one of the difficult situations, where the robot has to go back and get behind the ball to avoid scoring an own goal. Neither of these strategies resemble the typical hand-coded strategies.

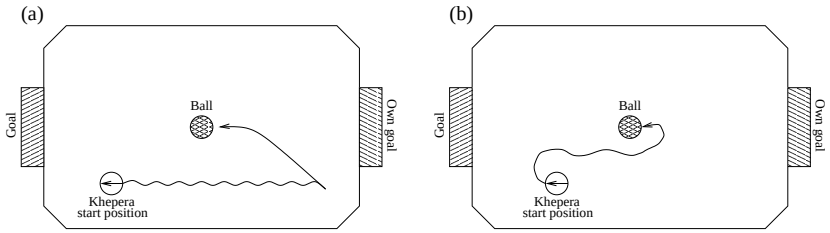

Fig. 8. *Different evolved football strategies.*

6.2 Testing real-world performance

A severe drop in fitness was observed when robots are transferred to reality. In the real world, the evolved robots would typically get a fitness of about 1000, which is quite far from the about 8000 typically obtained in simulation. This difference, known as the *reality gap*, is in this application probably mainly due to imperfect modelling of skidding and world object interactions.

The system was really put to the test, when an evolved robot controller participated in the Danish Robot Football Competition, Dec. 1999 under the name

Brute Force. Videos can be found on `http://www.mip.sdu.dk/~esben/EvoRobSoc`, and is the main source of documentation for the real robot performance. In the preliminary matches, *Brute Force* won two times and had one draw, which was enough to qualify for the semi finals. In the semi final *Brute Force* won 1-0 without much difficulties, but lost in the final against *KITT*, partly due to a dead battery in the second round (The evening before the final, *Brute Force* won 3-0 against *KITT* in a test match). An other robot soccer player evolved with the system participated in the FIRA2002 WorldCup KheperaSot tournament, where it won the second place.

The behavior of *Brute Force* is quite different from the behavior of the hand-coded competitors, being more in direct contact with the ball. Even though *Brute Force* was moving slower than its opponents, it still won many games by better maneuvering. *Brute Force*'s behavior in the simulator is shown in figure 9. By observation, this behavior corresponds to the behavior in the real world. An applet showing more *Brute Force* behavior can be seen on the papers home page [3].

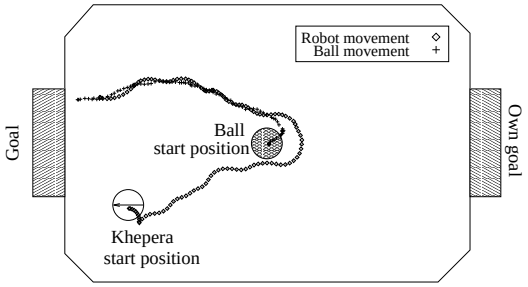

Fig. 9. *The plotted position of the robot Brute Force and the ball in the simulator. The ball starts to move when Brute Force pushes the ball.*

6.3 Testing Co-evolution, Chasing the Red Queen

Two different approaches were tried to track the Red Queen. One approach is evaluating individuals against reference players. The other approach is the Master Tournament method, described in [FN98]. Four tests were set up to test the effect of co-evolution.

Is co-evolution more efficient than single population evolution with respect to use of computing power? Two evolutionary setups were implemented. One with co-evolution of 2×50 individuals in the populations, and one single population evolution with 100 individuals in a population (players are siblings). In both cases, performance was measured a-posteriori against a reference player.

[3] http://www.mip.sdu.dk/~esben/EvoRobSoc/

Five runs were started with each setup, results are shown in figure 10. The single population evolutionary algorithm seems to climb faster, and seems to reach higher values. To test whether there is a statistical significant difference between the performance when using evolution or co-evolution, the average fitness of the last 100 generations was calculated for each population. From these data, statistics show that co-evolution performs significantly worse than single population evolution.

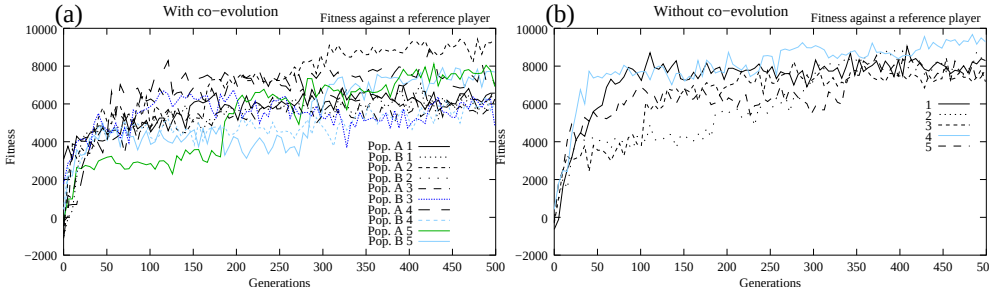

Fig. 10. *Co-evolution vs. single population evolution, test 1. Performance against a reference player of the co-evolution with two populations of 50 robots (a), and a single population of 100 robots (b).*

Does co-evolution hurt evolution? Another test was designed to test whether co-evolution made the evolutionary algorithm worse. In this test, the population size for the single population evolutionary algorithm was decreased to 50 and compared to 2×50 individuals in the co-evolutionary algorithm, so that the single population evolutionary algorithm runs with half as many fitness evaluations as the co-evolutionary algorithm.

Six runs, each 1000 generations, were made with the single population algorithm, and compared to five co-evolution runs. Results can be seen on the left half of figure 11.

The data shows that the difference between the two algorithms is not significant. The co-evolutionary algorithm shows no advantage over the single population algorithm in our data. Since the co-evolutionary algorithm uses twice as many fitness evaluations, the single population algorithm is to be preferred.

Does co-evolution win in the long run? Due to arms-race dynamics, co-evolution might have an advantage after some time, when the single-population evolution reaches a plateau. A test was made in which evolutions from the left graph of figure 11 were continued for another 2000 generations. The result of this run can be seen on the right graph in figure 11, and shows no significant difference in the two methods (at 5% confidence level). Over the last 100 generations, the single population evolutionary algorithm has a higher average value than the co-evolutionary, but the difference is not significant.

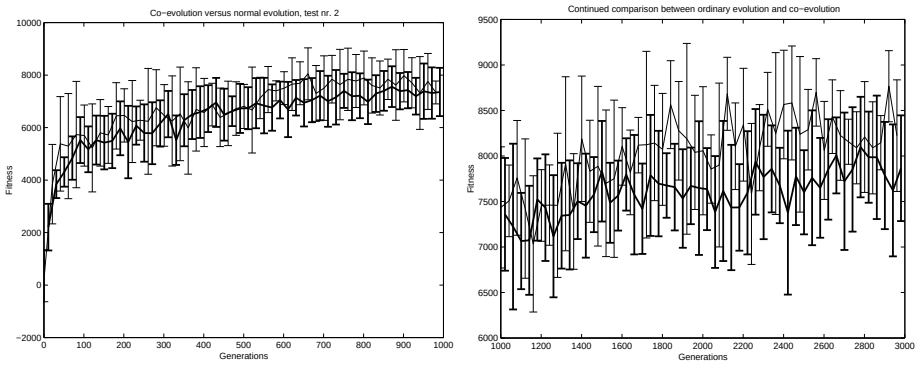

Fig. 11. Left: *Co-evolution vs. single population evolution. The graph shows the average and the* 95% *confidence interval of the elite fitness-evaluations averaged over both populations. The thin line is the single population evolutionary algorithm (6 runs), the thicker line is the co-evolution one (5 runs).* **Right:** *A continuation of the graph shown on the left.*

Does co-evolution produce more robust results? A fourth test was run to test whether the robots evolved with co-evolution were more robust with respect to handling different opponent strategies than the ones evolved with single population evolution, as argued by Stefano Nolfi and Dario Floreano in [NF98]. An experiment was set up, in which 10 co-evolved individuals played against 6 single population evolved individuals. The co-evolved individuals were chosen to be the best individuals from each population from the previous experiment, according to the master tournament method. For the 6 single population runs, the chosen individuals were the best individuals from the final generation of each run. Each of the co-evolved individuals was set to play 100 rounds against each of the individuals evolved with single population evolution. Each co-evolved individual thus played 600 rounds, while the other individuals played 1000 rounds. The average fitnesses obtained from these rounds are shown in the table in figure 12. The table shows a significantly higher average fitness for co-evolved individuals.

	Average fitness for individuals		
Player No.	*Without co-evolution*	*With co-evolution*	
		Pop. A	Pop. B
1	2306.11	3298.83	1481.15
2	2277.18	5349.24	4278.84
3	2772.82	3200.50	3432.08
4	1110.28	3011.15	6223.48
5	3964.32	4182.49	4476.53
6	245.54		
Average	**2112.71**	**3893.43**	
Std.Dev.	1297.68	1323.49	

Fig. 12. *Single population evolved players playing against co-evolved players.*

7 Discussion & Conclusion

A system has been implemented, that successfully uses evolution to generates controllers for the Khepera Robot Soccer task. Using this system, co-evolution is compared to single population evolution through four tests. The first three tests showed no benefit from using co-evolution. Single population evolution seems to converge faster to good solutions than co-evolutionary algorithms. However, the fourth test revealed, that the co-evolved individuals have developed more robust strategies than the single population evolved individuals. This supports D. Floreano and S. Nolfi's hypothesis, that "co-evolution can have a higher adaptive power than evolution" [FN98]. This series of experiments also shows, that the effect of co-evolution can be very difficult to measure. Under what prerequisites the positive effect of co-evolution will take place is more or less unexplored territory.

The objective of the work was to explore the co-evolutionary robotics approach and to test whether it could be used to evolve behavior for the Khepera robot soccer task. This seems to be the case. However, existing theories still need further development in several areas to reduce the amount of intuition required to build such a system. The problems of constructing simulators of a sufficiently high fidelity seems to be the major obstacle for the viability of the approach taken in this paper. Other problems are the heuristics involved in deciding behavior architecture and sensor and motor primitives.

In 1992, R. Brooks wrote "To compete with hand coding techniques it will be necessary to automatically evolve programs that are one to two orders of magnitude more complex than those previously reported in any domain." [Bro92]. This goal seems to have been reached for the present task, since the controllers evolved with this system plays football at least as well as most of the hand coded controllers in the Khepera robot soccer competitions.

References

[AP93] Peter J. Angeline and Jordan B. Pollack. Competitive Environments Evolve Better Solutions for Complex Tasks. In Stephanie Forrest, editor, *Genetic Algorithm: Proceedings of the Fifth International Conference (GA93)*, 1993.

[BLP96] Alan D. Blair, Mark Land, and Jordan B. Pollack. Coevolution of a backgammon player. In *Proceedings of the Fifth Artificial Life Conference*, 1996.

[Bro92] R. Brooks. Artificial life and real robots. In *European Conference on Artificial Life*, pages 3–10, 1992.

[CM95] Dave Cliff and Geoffrey F. Miller. Tracking the red queen: Measurements of adaptive progress in co-evolutionary simulations. In F. Morán, A. Moreno, J. J. Merelo, and P. Chacón, editors, *Proceedings of the Third European Conference on Artificial Life : Advances in Artificial Life*, volume 929 of *LNAI*, pages 200–218, Berlin, June 1995. Springer Verlag.

[CM96] Dave Cliff and George Miller. Co-evolution of Pursuit and Evasion II: Simulation Methods and Results. In Maes, P., Mataric̀, M., Meyer, J.-A., Pollack, J., and Wilson, S. W., editors, *From Animals to Animats 4: Proceedings of the*

fourth Internation Conference on Simulation of Adaptive Behavior (SAB96). MIT Press Bradford Books, 1996.

[FN98] Dario Floreano and Stefano Nolfi. Co-evolving predator and prey robots: Do 'arms races' arise in artificial evolution? *Artificial Life, 4 (4)*, pages 311–335, 1998.

[FNM98] Dario Floreano, Stefano Nolfi, and Francesco Mondada. Competitive co-evolutionary robotics: From theory to practice. In R. Pfeifer, B. Blumberg, J-A. Meyer, and S.W. Wilson, editors, *Animals to Animats V*, pages 512–524. Cambridge, MA: MIT Press, 1998.

[FNM01] Dario Floreano, Stefano Nolfi, and Francesco Mondada. Co-evolution and ontogenetic change in competing robots. In M. Patel, V. Honavarand, and K. Balakrishnan, editors, *Advances in the Evolutionary Synthesis of Intelligent Agents, Cambridge (MA)*. MIT Press, 2001.

[HCH92] I. Harvey, D. Cliff, and P. Husbands. Issues in evolutioanry robotics. In Roitblat, H. Meyer, J.-A. and Wilson, S., editors, *Proceedings of SAB92*. MIT Press Bradford Books, Cambridge, MA, jul 1992.

[Hil92] Daniel W. Hillis. Co-evolving parasites improve simulated evolution as an optimization procedure. In Chris Langton et al., editor, *Artificial Life II*, pages 313–324, 1992.

[Jac98a] Nick Jacobi. The minimal simulation approach to evolutionary robotics. In Takashi Gomi, editor, *Evolutionary Robotics, Volume II*, 1998.

[Jac98b] Nick Jacobi. Running across the reality gap: Octopod locomotion evolved in a minimal simulation. *Lecture Notes in Computer Science*, 1468:39–??, 1998.

[LHL97] Wei-Po Lee, John Hallam, and Henrik Hautop Lund. Learning complex robot behaviours by evolutionary approaches. *6th European Workshop on Learning Robots, EWLR-6*, aug 1997.

[MC96] Maja Matarič and Dave Cliff. Challenges in evolving controllers for physical robots. In *Proceedings, AAAI-92*, 1996.

[MNL95] Orazio Miglino, Stefano Nolfi, and Henrik Hautop Lund. Evolving mobile robots in simulated and real environments. *Artificial Life 2*, pages 417–434, 1995.

[NF98] Stefano Nolfi and Dario Floreano. How co-evolution can enhance the adaptive power of artificial evolution: Implications for evolutionary robotics. In Philip Husbands and Jean-Arcady Meyer, editors, *Lecture Notes in Computer Science*, volume 1468, 1998.

[Smi98] T.M.C. Smith. Blurred vision: Simulation-reality transfer of a visually guided robot. In P. Husbands and J.-A. Meyer, editors, *Evolutionary Robotics, First European Workshop: EvoRobot98*, Lecture Notes in Computer Science 1468, pages 152–164. Springer-Verlag, 1998.

[VPB99] M.M.B.R. Vellasco, M.A.C. Pchesco, and I.L Brasil. Mobile robot control using fuzzy logic. In Mondada, F. Löffler, A. and Rückert, U., editors, *Experiments with the Mini-Robot Khepera*, 1999.