Layered Learning for a Soccer Legged Robot Helped with a 3D Simulator

Andrea Cherubini, F Giannone, L Iocchi

▶ **To cite this version:**

Andrea Cherubini, F Giannone, L Iocchi. Layered Learning for a Soccer Legged Robot Helped with a 3D Simulator. Int. Robocup Symposium, 2007, 2007, Atlanta, United States. 10.1007/978-3-540-68847-1_39 . hal-01247242

HAL Id: hal-01247242
https://hal.science/hal-01247242

Submitted on 21 Dec 2015

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Layered learning for a soccer legged robot helped with a 3D simulator

A. Cherubini F. Giannone and L. Iocchi

Dipartimento di Informatica e Sistemistica
Università di Roma "La Sapienza"
Via Eudossiana 18, 00184 Roma, Italy
{cherubini, iocchi}@dis.uniroma1.it

Abstract. In the robotic soccer domain, many factors, such as the speed of individual robots, the effectiveness of kicks, and the choice of the appropriate attacking strategy, determine the success of a team. Consequently, soccer robots, and in particular legged ones, require fine tuning of the parameters not only for the vision processes, but also in the implementation of behaviors and basic control actions and in the strategic decisional processes. In recent years, machine learning techniques have been used to find optimal parameter sets for different behaviors. In particular, layered learning has been proposed to improve learning rate in robot learning tasks. In this paper we consider a layered learning approach for learning optimal parameters of basic control routines, behaviours and strategy selection. We compare three different methods in the different layers: genetic algorithm, Nelder-Mead, and policy gradient. Moreover, we study how to use a 3D simulator for speeding up robot learning. The results of our experimental work on AIBO robots are useful not only to state differences and similarities between different robot learning approaches used within the layered learning framework, but also to evaluate a more effective learning methodology that makes use of a simulator.

1 Introduction

In order for robots to be useful for many real-world applications, they must be able to adapt to novel and changing environments. For this purpose, a popular research field is the annual RoboCup soccer competition[1]. In this domain, many factors, such as the speed of individual robots, the effectiveness of kicks, and the choice of the appropriate attacking strategy, determine the success of a team. The robot should be able to respond to changes in its surroundings by adapting both its low-level skills (e.g., the walking style) and the higher-level skills (e.g., the behaviour) which depend on them.

Firstly, creating effective motion for walking and kicking the ball is a challenging task, since there is a large number of parameters to be set, and since successful motions strongly depend on many factors, including: playing surface, robot hardware, and game situation. In recent years, machine learning techniques have been used to find optimal parameter sets. Secondly, in Robocup matches, the correct choice of the best behaviours required to accomplish a certain task

[1] In this work, we focus on AIBO robots and RoboCup Four-Legged league (see www.tzi.de/4legged/)

(e.g., to score a goal) is fundamental for success. Machine learning techniques have been used in this field as well, in order to adapt the behaviours to the given game situation. In fact, machine learning generates solutions with little human interaction, and explores the search space of possible solutions in a systematic way, whereas humans are often biased towards exploring a small part of the space. An analysis of the methods used to learn optimal motion parameters and optimal behaviours, and comparisons with the approach presented in this paper, are reported in Section 2.

Following up on these works, we have analyzed different learning techniques for a AIBO soccer robot. In particular, we have considered a layered-like learning approach [14] that is suitable with the large dimensions of the search space we are considering. Besides, we have compared three different methods in the different layers: genetic algorithm (GA), Nelder-Mead (NM), and policy gradient (PG). Finally, we have studied how to use a 3D simulator for speeding up robot learning. The results of our experimental work can be summarized as follows: 1) layered learning is very effective in the complex scenario considered in this paper; 2) using a 3D simulator can actually speed up the learning process on real robots; 3) the learned low-level parameters are strongly related to the desired behavior. We have successfully experimented the presented learning methodology in preparation of RoboCup 2006, showing a notable improvement of performance of the basic behaviours of the robots in our team.

2 Related Work

Robot learning is a growing area of research at the intersection of robotics and machine learning. It has been used both at low level, for sensing and control issues, and at high level, for cognitive and behavior issues. We include in the first class – *parameter learning* – all problems where learning is aimed at fine-tuning of the parameters used by the low level algorithms. In the second class – *behavior learning* – we consider problems where learning is aimed at finding the optimal composition of simple behaviors for accomplishing a certain task.

In the first class of problems, one of the primary application areas is robot motion control, in all cases where the complete mathematical model is not known, and traditional optimization methods cannot be used. Gait optimization for legged robots is one of such fields. A Genetic Algorithm has been used for optimization of the vector of quadruped walk parameters, as shown in [4]. Algorithms based on an evolutionary approach achieved four-legged walks of high quality avoiding the need for gradient approximation [5, 13]. Kohl and Stone, from the University of Texas at Austin [8], empirically compared four different machine algorithms for quadruped walk optimization. Parameter learning has proved very effective for improving other motion control tasks, such as robot grasping. This task is achieved in [7] by applying the *layered learning* paradigm [14]: grasping parameters rely on previously learned walk parameters. Another interesting application area for parameter learning is robot vision. Bredeche and others [3], for instance, have shown the interest of using biologically inspired perceptual learning mechanisms to improve object identification in real-world environments. Researchers also proved the utility of behavior learning for improving some high level robot tasks, i.e. navigation, exploration, path-planning. Genetic Programming (GP, [9]) and Reinforcement Learning (RL) are often used for learning behaviors at the cognitive layer. The adaptation mechanism of GP is similar to

that of GAs, except that it is based on symbolic, rather than bit string, representation. In [6], RL is efficiently used for concept and rule acquisition in high level navigation in an unknown environment. Behavior learning is also a popular method for solving complex problems in multi-robot systems [11].

To our knowledge, the aforementioned classes of learning (parameter and behaviour learning) have rarely been joined in a single framework. In [1], this possibility has been explored: cognitive capabilities are developmentally created, starting from abilities for detecting, segmenting and recognizing objects up to task execution. In such approach, learning can be accelerated by increasing the complexity of the environment while the robot develops [12]. Moreover, experimental comparisons of different learning methodologies have been rarely addressed. In this work, we focus our attention on the definition of a learning task in which behavior learning and parameter learning are integrated and present an experimental evaluation of a layered learning approach using three different learning techniques: Genetic Algorithm, Policy Gradient and Nelder-Mead Simplex Method.

3 Problem definition

In this paper we consider learning a complex task as a composition of different behaviors. More specifically, we consider situations in which a task $\mathcal{T}$ can be accomplished by applying different strategies, where each strategy is a composition of different behaviors. Each behavior B is characterized by a set of parameters $\Theta_B = \{\theta_1, ..., \theta_{k_B}\}$. Notice that a behavior can be present in different strategies and possibly requires the definition of different parameters depending on the situation in which it is used. The learning problem is thus twofold: from one hand, behavior learning is needed to select the best strategy, i.e., the best composition of basic behaviors; from the other hand, each behavior needs a fine tuning of each parameter. In the next section we present a learning methodology that integrates behavior and parameter learning for a complex robot task.

To make this problem more clear we will present the application example in which we have tested our method. Consider a robot playing soccer within the RoboCup Four-Legged league competitions. One of the main tasks to be accomplished is to approach the ball and kick it to the opponent goal. This is a complex task that requires the integration of different behaviors in different ways. Many strategies can be defined to accomplish this task, but a winning strategy is difficult to identify since it depends on many factors: position of the robot and of the ball in the field, position of the other robots, etc. Besides, each behavior has several parameters to be tuned: walking gait parameters, kick parameters, etc. Also, these values should depend on the situation at hand: for example, we may prefer a fast but imprecise walk when the robot is far from the ball and an opponent robot is closer, while a slower but precise walk may be better when the robot is close to the ball and no other robots are around. Learning such a complex task requires to define a strategy (as a combination of behaviors) and tune the parameters of the behaviors involved in such a strategy.

4 System description

As aleady mentioned, we focus on learning the *attacking* task for a soccer robot in the Four-Legged League, i.e. learning to approach the ball and kick it to the opponent goal in the 'best' way. In the rest of this Section, we will describe our

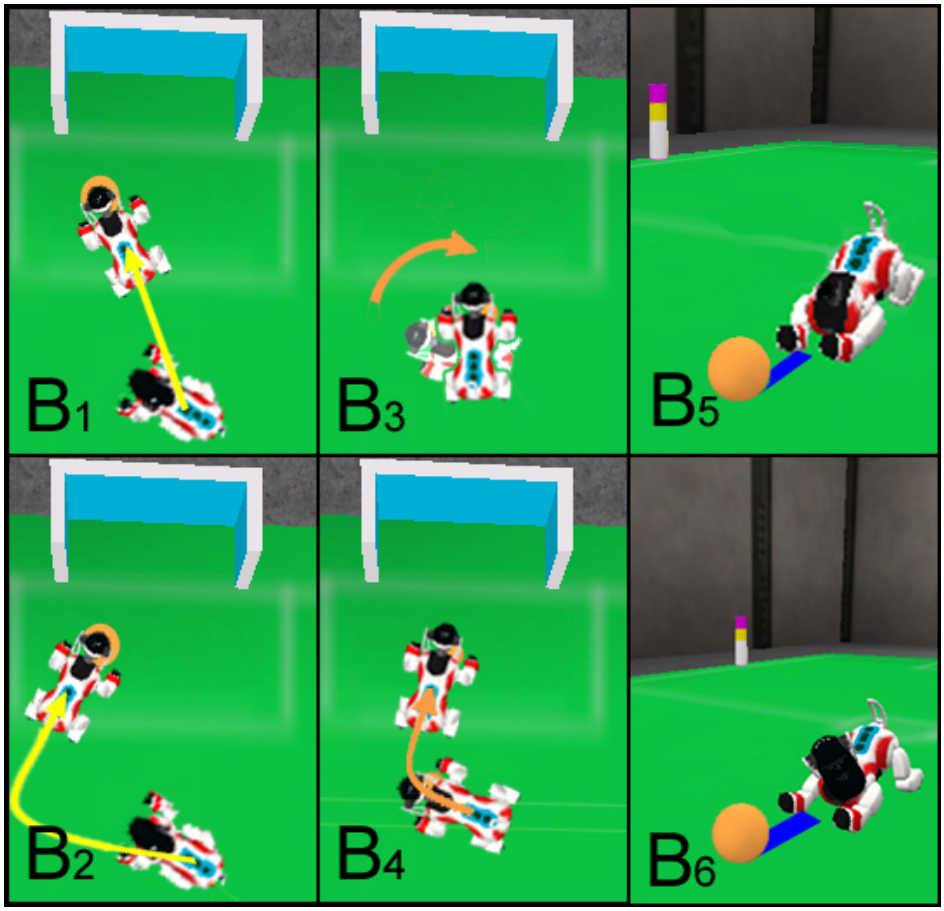

Fig. 1. The six behaviours that can be used in the attacking task.

task learning approach, by focusing on: the description of the system (behaviors and parameters of the attacking robot), the definition of learning approach, and the algorithms used for optimization of the robot performance.

4.1 Behaviors

For learning the *attacking* task, a set of six behaviors $\mathcal{B}_P = \{B_1, \ldots, B_6\}$ have been considered. These behaviors (see Figure 1) can be classified in three subsets:

- *Ball approaching* behaviors: B_1: *fast ball approach* (the path length is minimized by an omnidirectional walk), B_2: *aligning ball approach* (while the robot approaches the ball, its heading is concurrently oriented towards the opponent goal)
- *Ball carrying* behaviors (aimed at grasping the ball with the robot chin, and orienting the robot heading towards the opponent goal): B_3: *rotational ball carrying* (alignment is achieved by pure rotation), B_4: *rototranslational ball carrying* (alignment is achieved by a rototranslational movement towards the goal)

– *Ball kicking* behaviors (realized by direct kinematics control of the 15 joint positions): B_5: *head straight kick* (the robot 'dives' on the ball and hits it with the head), B_6: *head spanning kick* (the robot 'dives' on the ball and hits it with varying head pan).

A strategy is a combination of such behaviors. In this paper we consider only a simple form of behavior composition: chaining. Thus we consider *strategies* as sequences of behaviors. For example, $\{B_1; B_4; B_5\}$ is a possible strategy for the attacking task.

4.2 Parameters

Each behavior B is characterized by a set of parameters Θ_B.

Speed and stability of the ball approach are mainly characterized by the *walking gait parameters*, which define the kinematic characteristics of the walk. The **eleven** walking gait parameters $\Theta_{WG} = \{\theta_{WG,1}, \ldots, \theta_{WG,11}\}$ are: the relative positions of the locus center for fore/hind feet as (x, y, z) offsets (6 parameters), the height of the steps performed by the fore/hind legs (2 parameters), the step duration (1 parameter), and the maximum amplitude of the locus base for forward and lateral motions (2 parameters).

Walking gait parameters have little influence on the quality of ball carrying, which relies mainly on the way the robot slows down and eventually stops near the ball (i.e. on the *ball carrying parameters*). The **four** ball carrying parameters $\Theta_{BC} = \{\theta_{BC,1}, \ldots, \theta_{BC,4}\}$, which control the transition from approaching the ball to grasping it, are: the ball distance at which linear translational slowing down begins, the distance at which the robot stops translating, and similarly the relative robot-ball angle at which linear rotational slowing down begins, and the angle at which it terminates. These parameters also influence the quality of ball control in attacking strategies with no ball carrying (e.g., strategy $\{B_1; B_5\}$).

The robot kicks are generated by a sequence of fixed joint positions, related to a timing information, stating for how long the joint values must be kept (i.e. the kick 'speed'). The **nine** *ball kicking parameters* $\Theta_{BK} = \{\theta_{BK,1}, ..., \theta_{BK,9}\}$ are: the front legs initial and final angular positions (6 parameters), the robot head tilt initial and final angular positions (2 parameters) and the kick speed (1 parameter).

4.3 Learning to attack

Here, we present the optimization problem that must be solved in order to improve the attacker performance, based on the above system characteristics. As aforementioned, we present a learning approach that allows for the concurrent search of optimal behaviors and optimal parameters. The first issue deserves some clarification. We represent the attacker strategy P_i with a string of three integers, instead of symbolic expressions. We use the coding function:

$$c : P_i \rightarrow \{\phi_1\ \phi_2\ \phi_3\} \in \mathbf{Z}^3 \qquad\qquad i = 1, \ldots, 12$$

such that: ϕ_1 indicates the *ball approaching* behaviour used in strategy P_i (1 for B_1, or 2 for B_2), ϕ_2 the *ball carrying* behaviour (1 for B_3, 2 for B_4, 0 for no ball carrying), and ϕ_3 the *ball kicking* behaviour (1 for B_5, 2 for B_6). For example, the strategy $\{B_1; B_4; B_5\}$ is coded with the string $\{122\}$.

With this approach, the search for the optimal composition of simple behaviors amounts to a parameter optimization problem in the discrete set $S_\phi \subset \mathbf{Z}^3$.

In practice, since we consider parameter and behavior learning together, a candidate solution of the optimization problem is:

$$\underline{X} = [\theta_{WG,1} \ \ldots \ \theta_{WG,11} \ \theta_{BC,1} \ \ldots \ \theta_{BC,4} \ \theta_{BK,1} \ \ldots \ \theta_{BK,9} \ \phi_1 \ \phi_2 \ \phi_3]^T \in \mathbf{R}^{24} \times S_\phi$$

The very large dimensions of the search space, and the system characteristics, suggest the use of the *layered learning* paradigm [14] for optimizing this problem. In fact, given a hierarchical task decomposition, layered learning allows for learning at each level of the hierarchy, with learning at each level directly affecting learning at the next higher level. The *incremental learning* approach that we use is inspired by the layered learning paradigm; however, in contrast with classic layered learning, we utilize the same learning method for each layer of the hierarchy. Specifically, we can decompose optimization of the attacker task in the following four optimization subtasks (layers):

- L_1: find $\underline{X}_{1,opt} = [\theta_{WG,1} \ \ldots \ \theta_{WG,11}]^T_{opt} \in \mathbf{R}^{11}$ for 'best' walk;
- L_2: find $\underline{X}_{2,opt} = [\theta_{BC,1} \ \ldots \ \theta_{BC,4}]^T_{opt} \in \mathbf{R}^4$ for 'best' ball carrying, given $\underline{X}_{1,opt}$ found by L_1;
- L_3: find $\underline{X}_{3,opt} = [\theta_{BK,1} \ \ldots \ \theta_{BK,9}]^T_{opt} \in \mathbf{R}^9$ for 'best' ball kicking, given $\underline{X}_{1,opt}$ and $\underline{X}_{2,opt}$ found by L_2 and by L_3;
- L_4: find $\underline{X}_{4,opt} = [\phi_1 \ \ldots \ \phi_3]^T_{opt} \in \mathbf{Z}^3$ for 'best' attacking strategy, given $\underline{X}_{1,opt}$, $\underline{X}_{2,opt}$, and $\underline{X}_{3,opt}$, found by the three previous layers.

We note k_i the dimension of $\underline{X}_i$ at each level L_i ($k_1 = 11$, $k_2 = 4$, $k_3 = 9$, $k_4 = 3$). The solution can be obtained as:

$$\underline{X}_{opt} = [\underline{X}_{1,opt} \ \ \underline{X}_{2,opt} \ \ \underline{X}_{3,opt} \ \ \underline{X}_{4,opt}]^T$$

Irrespective of whether we use incremental learning or not, the appropriate choice of the objective function for optimization is fundamental. The function for evaluating the quality of an *attacking performance*, must take into account: the quality (speed and precision) of the robot motion for approaching the ball, the quality of ball carrying, and the quality (power and precision) of the kick. Hence, we adopt the following objective function:

$$F(\underline{X}) = k_{WG} f_{WG}(\underline{X}) + k_{BC} f_{BC}(\underline{X}) + k_{BK} f_{BK}(\underline{X})$$

where k_{WG}, k_{BC}, k_{BK} are positive weights indicating the significance desired for each of the three aspects in the learning process, and f_{WG}, f_{BC}, f_{BK} indicate respectively the quality of the walking gait, of ball carrying, and of ball kicking. These functions are derived with heuristics on the robot and ball positions at various stages of the attacking task.

4.4 Learning techniques

Our objective is to maximize $F(\underline{X})$ in a space of dimension k. This is not trivial, since $F(\underline{X})$ is 'black box', and although it can be computed on the real system as well as in the simulated environment, analytical computation of its derivatives is impossible, and all the parameters are box-constrained due to the physical characteristics of the system (we note : Δ_j the range size for each θ_j). Hence, conventional derivative-based optimization methods (e.g. Gradient or Newton

methods) cannot be utilized, and convergence analysis of a method is impossible. The selected approach must handle non differentiable search space, have high convergence rate, and be resistent to noise in $F(\underline{X})$. A variety of algorithms possess these characteristics. In particular, we will focus on three different machine learning algorithms: Genetic Algorithms, Nelder-Mead Simplex Method, and Policy Gradient. Characteristics of the three methods are briefly presented below.

Genetic algorithms have shown very interesting results in many low level problems [2]. A *population* of q parameter sets (*individuals*) is used to find a solution for the optimization problem. To evolve a new population from the tested one, the q_e best individuals are preserved (*elitism*) and the remaining individuals are generated by applying *crossover* (q_c individuals) and *mutation* (q_m individuals) operators.

The *Nelder-Mead simplex algorithm* is a very popular direct search method for multidimensional unconstrained minimization [10]. In a search space of dimension k, a *simplex* of $v = k + 1$ vertices is tested, and subsequently moved through the search space via four possible geometrical transformations: reflection, expansion, contraction, and shrinkage. From an initial parameter set ${}^0\underline{X}$, the other ${}^l\underline{X}$ vertices ($l = 1, \ldots, k$) of the first simplex are generated as: ${}^l\underline{X} = {}^0\underline{X} + [0, \ldots, \pm\zeta_l, \ldots, 0]^T$.

The *Policy Gradient algorithm* has been successfully used for robot learning [8]. From an initial parameter set ${}^0\underline{X}$, p randomly generated *policies* ${}^m\underline{X}$ ($m = 1, \ldots, p$), near ${}^0\underline{X}$, are evaluated, such that: ${}^m\underline{X} = {}^0\underline{X} + [\rho_1, \ldots, \rho_k]^T$, and each ρ_j is chosen randomly in the set $\{+\epsilon_j, 0, -\epsilon_j\}$. Each ${}^m\underline{X}$ is grouped into one of three sets for each j: $S_{+\epsilon,j}$, $S_{0,j}$ or $S_{-\epsilon,j}$ depending whether its j^{th} parameter was obtained by adding $+\epsilon_j$, 0 or $-\epsilon_j$. After evaluating the objective function at each ${}^m\underline{X}$, average scores $\bar{F}_{+\epsilon,j}$, $\bar{F}_{0,j}$, and $\bar{F}_{-\epsilon,j}$ are computed for $S_{+\epsilon,j}$, $S_{0,j}$ and $S_{-\epsilon,j}$, respectively. These scores are used to construct an estimate of the *gradient*, which is then normalized, multiplied by a scalar *step-size* η and added to ${}^0\underline{X}$, to begin the next iteration.

5 Experimental results

In this section we report the experimental results that we obtained by applying the proposed learning methodology in the described robot soccer scenario. More specifically, we comment on three results: 1) the effectiveness of the incremental approach; 2) the comparison among the three learning methods implemented: Genetic Algorithms, Nelder-Mead, and Policy Gradient; 3) the effectiveness of using a 3D simulator for speeding up the learning process.

We have executed the incremental learning approach presented in the previous section, but without considering the fourth layer L_4, since this would require additional input to the system. In fact, selecting the best strategy depends on other variables, such as the position of the robot and of the ball with respect to the target goal, the position of other robots in the field, etc. For layer L_4 it is necessary to learn a function that maps the current situation with a suitable strategy for that situation. This aspect needs to be further investigated and it is thus beyond the scope of the present paper.

In practice, we applied incremental learning, focused on layers L_1 to L_3, for optimizing the strategy $P_1 = \{B_1, B_5\}$, i.e., the robot approaches the ball as fast as possible, and kicks it forward, without grasping it, with fixed head pan.

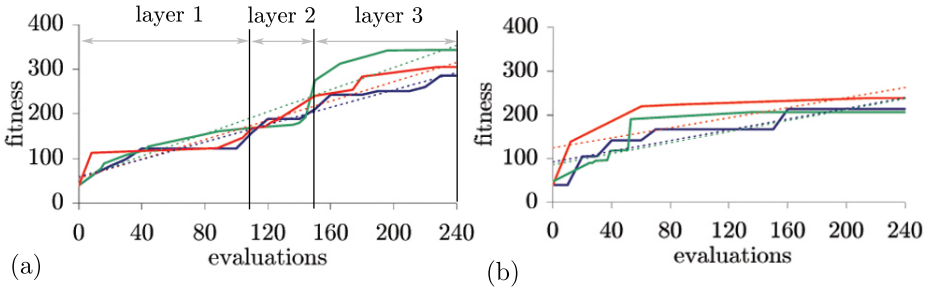

Fig. 2. Fitness values at each evaluation during USARSim training (solid lines: GA in blue, NM in green and PG in red) and linear interpolations (dashed lines): (a) incremental learning, (b) learning all parameters.

Learning this task has been initially developed and configured within the 3D AIBO simulator [15] embedded in USARSim (Urban Search and Rescue Simulator), before experimentation on the real robot. In fact, one of the objectives of this work is to show that the learning configurations tuned in the simulated environment can be successfully ported for learning on the real robot.

Here, we briefly outline the configurations of the three learning algorithms. For the genetic algorithm, we chose: $q = 10$, $q_e = 1$, $q_c = 6$, $q_m = 3$. *Selection* of individuals from the original population is based on the popular roulette wheel scheme, and mutation is obtained by altering the j^{th} parameter with an offset chosen randomly in the set $[-0.2\,\Delta_j, +0.2\,\Delta_j]$. For the Nelder-Mead algorithm, at each layer L_i: $v = k_i + 1$, and $\zeta_l = \pm 0.2\,\Delta_j$, with the sign of ζ_l chosen randomly. For the policy gradient, we use for the three layers: $p_1 = 8$, $p_2 = 4$, $p_3 = 6$, $\epsilon_j = 0.1\,\Delta_j$, $\eta = 3$.

The same initial parameter set ${}^0\underline{X}$ is used for starting each learning technique. Since there is significant noise in each experiment, each set of parameters is evaluated three times, and the resulting fitness *evaluated* for that set, is computed by averaging over the three experiments. For each layer, and each learning technique, we terminate learning after a number of *evaluations* equal to: $10k_i$ has been performed (e.g. for L_1, 110 evaluations, i.e., 330 experiments). We choose to use the same amount of learning time, since this is usually a given specification in learning problems. The results of the incremental learning algorithm are shown in Fig. 2(a), where the best fitness value found by the learning algorithms is plotted over the number of evaluations. A similar plot is shown in Fig. 2(b), where we ran the same number of evaluations by learning all 24 parameters at the same time Comparison between the two plots confirms the qualitative properties of the incremental approach: for instance, for the GA, the final fitness obtained by the incremental approach is 34% higher than that obtained by learning all parameters together, and the slope of the interpolating line is greater (0.97 versus 0.60) in Fig. 2(a) than in Fig. 2(b). Nelder-Mead slightly outperforms Genetic Algorithm and Policy Gradient.

Other experiments were carried out to show how the optimal low-level parameters are strongly related to the desired behavior. To emphasize this aspect, we used the same learning configuration used for optimizing strategy $P_1 = \{B_1, B_5\}$, to optimize strategy $P_2 = \{B_1, B_3, B_5\}$ (the robot approaches the ball as fast as possible, rotates while grasping it, and kicks it forward with fixed head pan). This experiment was carried out with the genetic algorithm, in USARSim: starting

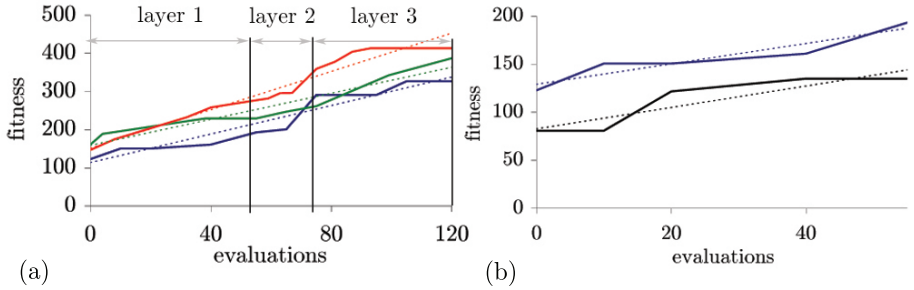

Fig. 3. Fitness values during AIBO training: (a) incremental learning (solid lines: GA in blue, NM in green, PG in red) with start point derived from USARSim, and linear interpolations (dashed lines), (b) comparing GA for L_1: with initial population derived from USARSim (solid blue line), and with random initial population (solid black line), and linear interpolations (dashed lines).

from the optimal walking gait parameters $\underline{X}_{1,opt}$ learned for the previous strategy, we proceeded with the other two levels to derive $\underline{X}_{2,opt}$ and $\underline{X}_{3,opt}$. Comparison between the optima learned for P_1 and P_2 showed major differences. In fact, for P_1, the optimal parameters are:

$$\underline{X}_{2,opt} = [148\ 5\ 50\ 6]^T \qquad \underline{X}_{3,opt} = [21\ 44\ 969\ 1611\ 30\ 0\ -684\ -350\ 26]^T$$

whereas for P_2, the optimal parameters are:

$$\underline{X}_{2,opt} = [300\ 100\ 100\ 7]^T \qquad \underline{X}_{3,opt} = [19\ 71\ 908\ 1654\ 10\ 0\ -879\ -247\ 17]^T$$

The differences are reasonable: for P_2, the robot must slow down farther away from the ball (parameters $\theta_{BC,1}$ and $\theta_{BC,2}$ are different in the two cases) and the head movement for kicking the ball (which in P_2 has been grasped) is different and depends on parameters $\theta_{BK,7}$ and $\theta_{BK,8}$.

After having configured the algorithm for learning strategy P_1 in the 3D simulated environment, we ported it on the real robot. The initial parameter set for each technique is the set derived at the end of simulator optimization. This time, for each layer, and each learning technique, we terminate learning after a number of *evaluations* equal to $5\ k_i$ has been performed. The results of the incremental learning algorithm are shown in Fig. 3(a), where the best fitness value is plotted over the number of evaluations. On AIBO, policy gradient slightly outperforms the two other approaches.

To emphasize how the use of a 3D simulator speeds up the learning process on real robots, we ran another experiment where the GA was used to learn L_1 for the same strategy P_1, starting from a random population, different from the one derived at the end of USARSim optimization. The results of this experiment are shown in Fig. 3(b), where the fitness of GA walking gait learning starting from different populations are plotted: the figure clearly shows the advantage of the simulator for deriving the GA initial population.

6 Conclusions

In this paper we presented a layered-like approach for learning optimal parameters, and strategy selection. We compared three different methods in the different layers: genetic algorithm, Nelder-Mead, and policy gradient. Moreover, we showed how the use of a 3D simulator speeds up robot learning.

The proposed learning methodology has been applied to the soccer attacking task. It has been implemented in USARSim and extensively experimented in laboratory on AIBO, showing a notable improvement in the performance of the robot basic behaviours. The main results of the experiments are: the utility of the layered approach for this complex scenario, and the effectiveness of the 3D simulator for configuring the learning algorithms before porting on the robot.

Incremental learning has been executed without considering the strategy selection layer. In fact, this depends on the 'game situation' (e.g., positions of robot and ball with respect to the goal). Learning a function that maps the game situation with a suitable strategy will be the object of further work.

References

1. A.M. Arsenio. Developmental learning on a humanoid robot. In *IEEE International Joint Conference on Neural Networks, Budapest*, 2004.
2. T. Back. Optimization by means of genetic algorithms. In *36th International Scientific Colloquium*, pages 163–169, 1991.
3. N. Bredeche, Z. Shi, and J.-D. Zucker. Perceptual learning and abstraction in machine learning: an application to autonomous robots. *Systems, Man and Cybernetics, Part C, IEEE Transactions on*, 36(2):172–181, March 2006.
4. S.K. Chalup, C.L. Murch, and M.J. Quinlan. Machine learning with AIBO robots in the Four-Legged League of Robocup. *Systems, Man and Cybernetics, Part C, IEEE Transactions on*, 2006.
5. S. Chernova and M. Veloso. An evolutionary approach to gait learning for four-legged robots. In *IEEE/RSJ International Conference on Intelligent Robots and Systems 2004, IROS 2004*, 2004.
6. J. del R. Millan. Rapid, safe, and incremental learning of navigation strategies. *Systems, Man and Cybernetics, Part B, IEEE Transactions on*, 26(3):408–420, June 1996.
7. P. Fidelman and P. Stone. The chin pinch: A case study in skill learning on a legged robot. In *Proceedings of 10th International Robocup 2006 Symposium*, 2006.
8. N. Kohl and P. Stone. Machine learning for fast quadrupedal locomotion. In *The Nineteenth National Conference on Artificial Intelligence*, 2004.
9. J.R. Koza. *Genetic Programming: on the programming of computers by means of natural selection*. Cambridge, Mass., 1992.
10. J. C. Lagarias, J. A. Reeds, M. H. Wright, and P. E. Wright. Convergence properties of the Nelder Mead simplex algorithm in low dimensions. *SIAM Journal on Optimization*, 9:112–147, 1998.
11. E. Martinson and R.C. Arkin. Learning to role-switch in multi-robot systems. In *Proceedings of International Conference on Robotics and Automation*, 2003.
12. Y. Nagai, M. Asada, and K. Hosoda. A developmental approach accelerates learning of joint attention. In *2nd International Conference on Development and Learning*, 2003.
13. T. Rofer. Evolutionary gait-optimization using a fitness function based on proprioception. In Springer, editor, *ser. LNCS, D. Nardi, M. Riedmiller, C. Sammut and J. Santos-Victor*, volume 3276, 2004.
14. P. Stone and M. Veloso. Layered learning. In R. L. De Mántaras and E. Plaza, editors, *Machine Learning: ECML 2000 (Proceedings of the Eleventh European Conference on Machine Learning*, pages 369–381. Springer Verlag, Barcelona, Spain, May/June 2000.
15. M. Zaratti, M. Fratarcangeli, and L. Iocchi. A 3D simulator of multiple legged robots based on USARSim. In *Proceedings of 10th International Robocup 2006 Symposium*, 2006.