A robot behavior-learning experiment using Particle Swarm Optimization for training a neural-based Animat

Fabien Moutarde
Robotics laboratory (CAOR)
Mines ParisTech
60 Bd St Michel, F-75006 Paris, FRANCE
Fabien.Moutarde@ensmp.fr

***Abstract*— We investigate the use of Particle Swarm Optimization (PSO), and compare with Genetic Algorithms (GA), for a particular robot behavior-learning task: the training of an animat behavior totally determined by a fully-recurrent neural network, and with which we try to fulfill a simple exploration and food foraging task. The target behavior is simple, but the learning task is challenging because of the dynamic complexity of fully-recurrent neural networks. We show that standard PSO yield very good results for this learning problem, and appears to be much more effective than simple GA.**

***Keywords*— animat, behavior-learning, genetic algorithms, particle swarm optimization, recurrent neural network.**

I. INTRODUCTION

Animats are simulated animals or robots with behaviors inspired by those of real animals, for instance particular locomotion modes, or more abstract tasks such as exploration for food foraging [1]. One of the aims of animat research is to explore means of "naturally" producing (as opposed to "hand-crafting") autonomous and adaptive behaviors for simulated animals or biologically inspired robots. A very fruitful part of this research field has focused on evolutionary robotics, where behaviors are obtained by evolutionary algorithms and self-organization [2].

Genetic Algorithms (GA), originally developed in the 1960s by John Holland, are an optimization method inspired by evolution where natural selection ensures that fittest members of a population tend to reproduce more often than others. In order to search a good solution to a problem, a population of solutions is first generated, each one being encoded as a genome, i.e. a set of genes (each of them corresponding either to a single bit or to one elementary information such as an integer or floating-point number) [3]. The genetic algorithm then proceeds by evolving this population with reproduction biased by selection according to fitness evaluation of individuals, followed by crossover (hybridization of genomes from 2 previous solutions) and mutations [4]. Evolutionary algorithms have been extensively used for learning of robot behaviours, and standard GA have already been shown to be effective in evolving simple robotic controllers [5].

The Particle Swarm Optimization (PSO) algorithm, originally introduced by Kennedy & Eberhart [6], is a kind of population-based cooperative-search inspired by swarm intelligence such as bird flocking and fish schooling. The basic PSO model consists in a swarm of particles moving in an n-dimensional search space, in which a certain objective function is to be optimized. Each particle has a position `x` in the optimization space, and a velocity vector `v` defining its moving speed in this space; it also remembers its own best past position `bp`. Furthermore, each particle has a set of "informants" (defined by a neighbourhood relation in the swarm), which provide a best group-explored position `bg`. At each iteration step, the velocity of each particle is updated according to equation (1), in which `I(t)`, `N(t)` and `F(t)` are three time-dependant coefficients described below. Once the new velocity `v(t+1)` has been computed, the particle is moved to its new position according to its velocity by the following equation: `x(t+1)=x(t)+v(t+1)`.

$$v(t+1) = I(t)v(t) + N(t)(bp(t) - x(t)) + F(t)(bg(t) - x(t)) \quad (1)$$

As seen on equation (1), the velocity update is a compromise between inertia (keeping the same velocity vector) `I(t)`, nostalgia (trying to go back to best past position) `N(t)`, and following (going to best known group position) `F(t)`. Each of these three coefficients is usually itself the product of a uniform random variable in [0;1] by specific fixed parameters `I`, `N` and `F`. According to theoretical and empirical studies (see e.g. [7], [8] and [9]), best convergence is obtained with values `I~0.7` and `N~S~1.5`.

PSO has proven to be a very efficient optimization method, with successful applications in many fields, as reported for instance in [10]. PSO has been shown to perform as well as, or better than, Genetic Algorithms (GA) for several optimization problems (see for instance [6]).

One of the fields it has been fruitfully applied to, is the training of neural network weights (see eg [11]), including (more recently) unsupervised training of recurrent networks (see eg [12]). Several authors, among which Pugh et al. in [12] have already compared the performance of PSO with that of

978-1-4244-2287-6/08/$25.00 ©2008 IEEE

classical Genetic Algorithm evolution for unsupervised robotics learning tasks, but with no clear and definite conclusion about the superiority of one or the other method.

In this paper, we investigate the use of PSO algorithm for the behaviour-training of an animat's recurrent neural network "brain", in order to obtain a simple exploration and food-foraging behaviour. We also compare with results obtained for the same task with GA.

II. ANIMAT MODEL

Our animat model is very similar to that proposed in [13]. The animat behavior is defined by a fully recurrent neural network with 5 sensory inputs, and a total of N fully-connected neurons among which 4 are motor outputs, as illustrated on figure 1. A *recurrent* neural network is used in order to allow the animat's behavior not to be purely reactive on sensor inputs, but to depend on some internal state so as to be able to exhibit complex behaviors.

The first 4 sensory inputs provide ternary information on presence/absence of "something" (+1, -1 or 0 respectively for presence of food or poison, presence of wall, or nothing) on the animat current position and on 3 adjacent cells (in front of the animat's current orientation, on the left and on the right). The fifth sensory input is a "smell" binary input indicating if food (rather than poison) is present on the animat's current position, but its value is random when there is nothing on the current cell. As exposed in [13], the idea behind this last "smell" input is that, because the "presence sensor" does *not* discriminate between food and poison, the eating decision has to be made by fusion of 2 inputs: presence of something on current place, simultaneously with food smell.

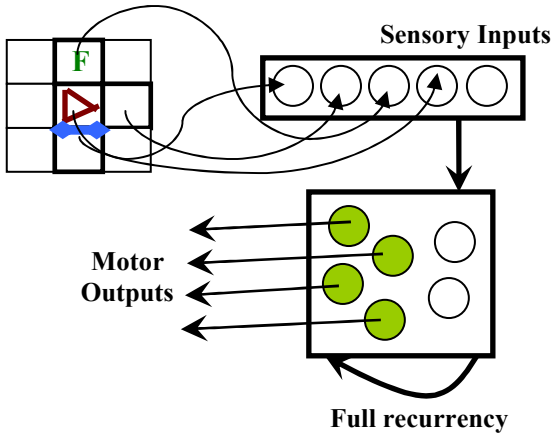

Figure 1. Schematic view of neural "brain" of the animat. It senses only its immediate environment, with 4 input giving information on presence of something (food, poison or wall on current and 3 adjacent cells in front/left/right, and a last input giving a local smell (random in case nothing is present). The "brain" is a fully recurrent network with N≥4 neurons, including the 4 motor output neurons.

The 4 motor outputs independently suggest left-rotation, right-rotation, forward-move, and eating. The actual action taken by the animat is the one corresponding to the "active" output if only one is activated, or nothing if several motor output are simultaneously active. This forces the animat to learn to produce a single-action decision.

The animat behavior is therefore totally defined by its brain size (number of neurons) and the values of the $5N+N(N+1)=N^2+6N$ weights and biases of the recurrent neural network. In practice, all results presented in this paper use N=4.

III. BEHAVIOUR-TRAINING EXPERIMENT

The behavior to be learnt by the animat is to forage a grid-structured space where a fixed amount of "Food" and "Poison" are placed at random locations, and to "eat" as much food as possible but no poison. The grid size is fixed and equal to 7x7 cells, and the food and poison quantity are set respectively to 6 and 11, as illustrated on figure 2. Note that for the behavior visualization we have used and modified an old version of Becker's implementation of "Karel the Robot" ([14]).

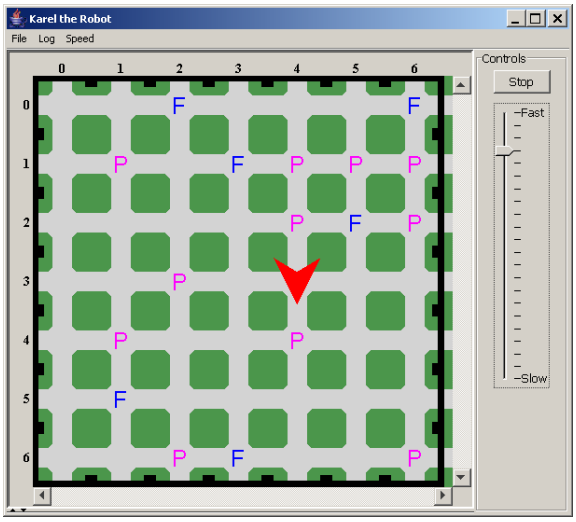

Figure 2. Typical grid and animat initial state before a behavior evaluation. The animat position and orientation is given by the red arrow. Positions of "food" and "poison" positions are respectively given by the F and P letters randomly scattered on the grid.

The evaluation of a given neural brain is done by generating G random grids (with random food and poison positions). For each grid, the animat has a random initial position and orientation, its brain neuron *states* are all zero-initialized, and the animat is allowed a fixed number of 150 sensory-motor cycles. The animat evaluation is done by a combination of 2 criteria:

- the food-collecting aptitude obtained by just counting the number of food eaten decremented by the number of eaten poison, and divided by the total food available;
- the exploration efficiency measured by the proportion of the total grid space actually explored by the animat during its 150 sensory-motor cycles.

Those two criteria are averaged over the G different random

grids, and combined in one single "score" for each animat brain, given in equation 2 below. For this evaluation to be really representative of the behavior in any grid configuration, G must be sufficiently high. We typically use G~300.

score=(evalFood+ratioExplo*evalExplo)/(1+ratioExplo) (2)

The `ratioExplo` parameter allows to tune the respective weights in the animat's score, of "food-collecting aptitude" and "exploration efficiency".

IV. TRAINING WITH PSO

We first present results of behavior-training with Particle Swarm Optimization (PSO) for the task described in §III.

The PSO-based behavior-training is done by applying standard PSO as described in §I, to a swarm of animats with uniform brain size N. Each particle is therefore simply the vector of $d=N^2+6N$ weights and biases values of the recurrent neural network, where N is the brain size. Particles are initialized inside the $[-10.;10]^d$ area.

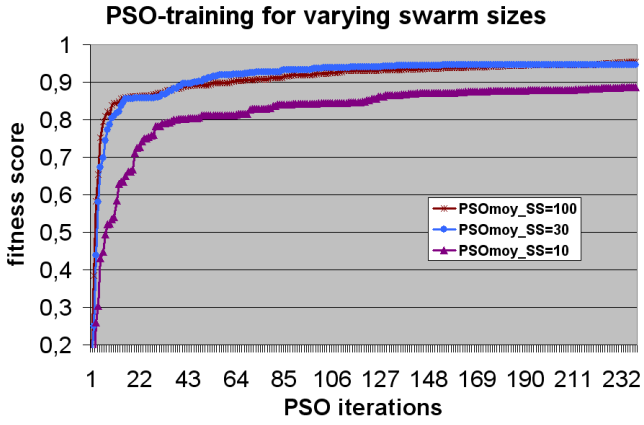

Figure 3. PSO-training results for varying swarm sizes (SS) (averages over 10 independent training tests, with fitness score defined by equation 2 with `ratioExplo=0`); no significant improvement appears over SS>30.

We have tried and compared several swarm sizes (SS), from SS=10 to SS=100. We use a "complete neighborhood" topology in which all particles belong to the same information group. The PSO parameters are fixed to I=0.729 and N=S=1.429 (a set of values suggested in [7]). We first considered only the food collecting capability, i.e. we set `ratioExplo` to zero in equation 2. As can be seen on figure 3, PSO training yields very good results in small number of iterations. It also appears that no significant improvement is obtained by growing swarm size over SS=30.

We also tried using fitness function with `ratioExplo>0` in equation (2) of §III. An interesting finding is that beginning training with `ratioExplo>0` (for instance 10 first iterations with `ratioExplo=1`) seems to accelerate significantly the later convergence to optimum fitness score with `ratioExplo=0`. In other terms, optimizing simultaneously the exploration criterium at the beginning of training apparently allows faster subsequent training based only on food collection, as illustrated on figure 4.

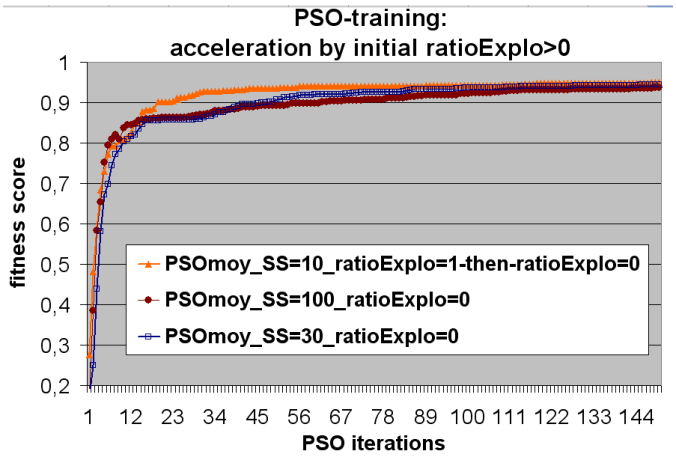

Figure 4. Acceleration of PSO training by an initial short period of training with ratioExplo=1 in fitness function, appearing on upper curve (averages over 10 independent training tests)

V. TRAINING WITH GA

We now present training results obtained with Genetic Algorithms (GA) on exactly the same task described in §III.

The GA-based training uses genomes of length $L=N^2+6N$ consisting of double values (each gene corresponding to one weight or bias value of the recurrent neural network, so that each genome corresponds to one possible animat "brain" with N neurons). For the GA evolution operators, we used fixed-point middle cross-over, and uniform randomization in [-10;10] interval as gene mutation, so that explored space is the same as in the PSO training. Several mutations policies and rates were tried. We report here only on the mutation scheme which gave best result: nearly systematic (0.9 probability) mutation of only one single gene. Several population sizes were also tested and compared, as reported on figure 5.

Obviously, the GA "flavour" we are using is not the most sophisticated possible. However, we also used the most standard version of PSO in §IV, and our aim is to compare what can be achieved for our behaviour-training task by applying techniques as straightforwardly as possible, rather than focusing on fine-tuning the algorithms used.

As can be seen on figure 5, GA training can also provide animat "brains" with quite good performance on the assigned task. It also seems that maximal performance can be attained on this problem with a population size PS=100, with only slight acceleration of evolution for bigger sizes.

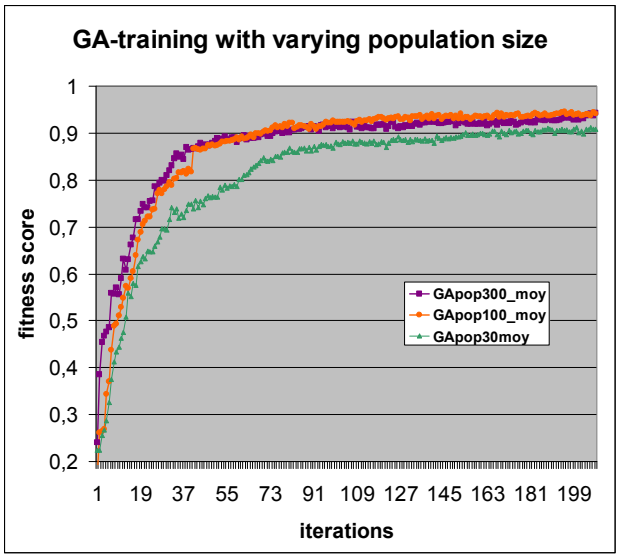

Figure 5. GA-training results for varying population sizes (PS) (averages over 10 independent training tests); it seems no further improvement can be obtained for PS>100.

VI. COMPARING PSO-BASED AND GA-BASED TRAINING

We can now compare, on the exact same task described in §III, unsupervised learning with Particle Swarm Optimization (PSO), and using Genetic Algorithms (GA). Both algorithms are purposely used in their most standard variant, as our aim is rather to check which simple technique can yield good results even when applied without tedious fine-tuning.

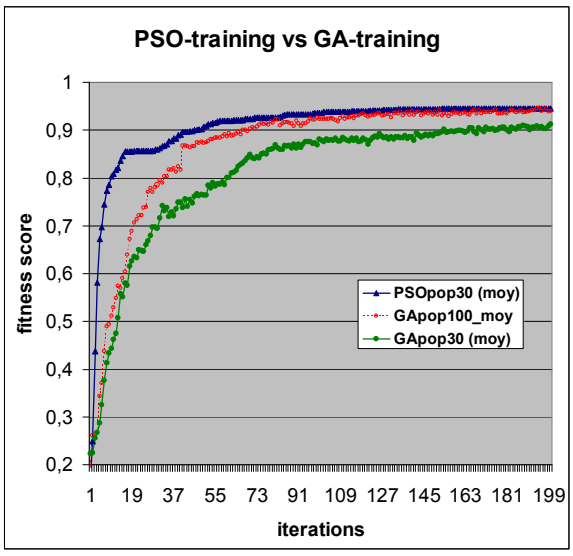

Figure 6. Comparison of PSO-training and GA-training (averages over 10 independent training tests). For the same population size of 30, PSO training is much quicker and reaches a better final fitness. Note also that even with a 3 times larger population, GA training requires more iterations to attain nearly the same final fitness.

In our problem, the computation time consists essentially of the particles/genomes evaluation on a large set of 300 random grids, and the particles updating or genome evolution computation is negligible compared to it. The total computing time for either PSO or GA is therefore basically proportional to the population size and the number of required iterations.

Standard PSO turns out to be much more efficient than simple GA for training, as illustrated on figure 6. Not only the typical maximum fitness score is slightly higher (0.95 instead of 0.9) with a population size of 30, but the PSO training is definitely much quicker, with fitness score reaching 0.9 in less than 50 iterations. Even when comparing 30-particles PSO with 100-genomes GA, it can be seen that PSO requires much less iterations to attain the same animat fitness level: about 200 iterations instead of 120 to reach maximum fitness. Considering that computing time (largely dominated by evaluation of particles/genomes) is proportional to population size and number of iterations, this means that for our unsupervised learning task PSO appears to be roughly (200x100)/(120x30)~5 times faster for a same optimization level of solution.

VII. CONCLUSIONS AND FUTURE WORK

We have presented a robot behavior-learning experiment in which an animat's "brain" consisting of a fully recurrent neural network has to be trained to produce a foraging behavior. We have shown that standard Particle Swarm Optimization (PSO) seems particularly effective for this kind of behavior-learning task. According to our experiments, standard PSO clearly outperforms simple Genetic Algorithms (GA) on our particular neural-animat behavior-learning setup.

Our results tend to confirm earlier results by Pugh et al [12] in similar context. This reinforces the presumption that PSO algorithm is particularly efficient, and a very appealing alternative to GA evolution, for these kinds of behavior learning tasks.

Further work planned includes similar systematic investigation for learning of more complex behaviors, in particular *cooperative* behavior of several animats.

REFERENCES

[1] Meyer, J.-A. and Guillot, A. Biologically-inspired robots. In Siciliano, B. and Khatib, O., editors, *Handbook of Robotics*. Springer-Verlag. (2008).

[2] Floreano, D., Husbands, P. and Nolfi, S., Evolutionary Robotics. in Handbook of Robotics, Berlin : Springer Verlag, 2008.

[3] Goldberg, D. E. Genetic Algorithms in Search, Optimization & Machine Learning. Addison-Wesley, Reading, MA, 1989.

[4] Mitchell, M. An Introduction to Genetic Algorithms. MIT Press, Cambridge, MA, 1996.

[5] Floreano, D. & Mondada, F. "Evolution of Homing Navigation in a Real Mobile Robot" Systems, Man and Cybernetics, Part B, IEEE Transactions on, Vol. 26, No. 3, Jun 1996, pp. 396-407.

[6] Kennedy J. & Eberhart R., "Particle swarm optimization", 1995. Proc. IEEE Int. Conf. on Neural Networks, Nov/Dec 1995, pp. 1942-1948.

[7] M. Clerc, The swarm and the queen: towards a deterministic and adaptive particle swarm optimization, in: Proc. ICEC, Washington, DC, 1999, pp. 1951–1957.

[8] Clerc, M. and Kennedy, J. The particle swarm: explosion stability and convergence in a multidimensional complex space. IEEE Transactions on Evolutionary Computation 6(1), 2002, pp. 58-73

[9] Trelea I.C., The particle swarm optimization algorithm: convergence analysis and parameter selection. Information Processing Letters, 85, pp. 317-325 (2003).

[10] Eberhart; Yuhui Shi, "Particle swarm optimization: developments, applications and resources," Evolutionary Computation, 2001. Proceedings of the 2001 Congress on , vol.1, no., pp.81-86 vol. 1, 2001

[11] Hong-Bo Liu; Yi-Yuan Tang; Jun Meng; Ye Ji, "Neural networks learning using vbest model particle swarm optimisation," Machine Learning and Cybernetics, 2004. Proceedings of 2004 International Conference on , vol.5, no., pp. 3157-3159 vol.5, 26-29 Aug. 2004

[12] Pugh, J., Zhang, Y. & Martinoli, A. "Particle swarm optimization for unsupervised robotic learning" Swarm Intelligence Symposium, Pasadena, CA, June 2005, pp. 92-99.

[13] R. Aharonov-Barki, T Beker & E. Ruppin , "Emergence of memory-driven command neurons in evolved artificial agents" , Neural Computation, 13(3), 2001, pp 691-716

[14] B. W. Becker, "Java: Learning to Program with Robots", Course Technology, 2006. [http://www.learningwithrobots.com]