An Evolutionary Algorithm for Multi-Robot Unsupervised Learning

Philippe Lucidarme
ISI/AIST - STIC/CNRS Joint Robotics Laboratory
ISI, AIST, Central 2, 1-1-1 Umezono, Tsukuba,
305-8568 JAPAN
philippe.lucidarme@aist.go.jp

Abstract - **Based on evolutionary computation principles, an algorithm is presented for learning safe navigation of multiple robot systems. It is a basic step towards automatic generation of sensorimotor control architectures for completing complex cooperative tasks while using simple reactive mobile robots. Each individual estimates its own performance, without requiring any supervision. When two robots meet each other, the proposed crossover mechanism allows them to improve the mean performance index. In order to accelerate the evolution and prevent the population from staying in a local maximum, an adaptive self-mutation is added: the mutation rate is made dependent on the individual performance. Computer simulations and experiments using a team of real mobile robots have demonstrated the rapidity of convergence to the best-expected solution.**

I. Introduction

Cooperation of multiple mobile "autonomous" robots is a growing field of interest for many applications, mainly in industry and in hostile environments such as planet exploration and sample return. Theoretical studies, simulations and laboratory experiments have demonstrated that intelligent, robust and fault-tolerant collective behaviors can emerge from colonies of simple automata. This tendency is an alternative to the all-programmed and supervised learning and operation used so far. The "animats" concept thus joins the pioneering works on "Cybernetics" published in the middle of the previous century.

Although human supervision would obviously remain necessary to expensive missions, long and tedious programming tasks would be cut out if designing robots capable of self-learning, self-organization and adaptation to unexpected environmental changes was made possible. Previous works have shown many advantages for self-learning robots:

1. At the lowest level, complex legged robots can learn how to stand up and to walk [1]
2. A mobile robot can learn how to avoid obstacles [2] and to plan a safe route towards a given goal [3 and 4]
3. A pair of different mobile robots can learn to cooperate in a box-pushing task [5]
4. Efficient global behaviors can emerge in groups of robots [6]

The bottom-up approach for building architectures of robotic multi-agent systems automatically acquiring distributed intelligence appeared to be simple and efficient. However, even if we do not ignore the needs, for some applications, for communicating indirectly information, by letting the robots deposit beacons for example, direct modes are of prime interest. They use exteroceptive sensors: force sensing, "vision" and message passing. It has been demonstrated that even very simple information sharing induces a significant enhancement of both the individual and group performance [6, 7 and 8].

To obtain learning and adaptive abilities, it seemed natural to call for the genetic algorithm principles [9], which transform populations of "individuals" into new ones abler to perform given tasks in an uncertain and varying environment [10,11,12 and 13]. Some of our previous works, ranging from optimal allocation of robots and machines in factories [14] to multi-path generation of mobile robots on uneven natural terrain [15 and 16], have led to adapting such principles to robotics. In this paper, we rather focus on cooperative basic learning of multi-robot systems, aiming at genetic programming for tasks that are more complex.

Section 2 describes the proposed evolutionary algorithm and demonstrates the convergence to the best behavior. Simulations are given in section 3 that allows testing the performance in exploring the environment. The convergence time is presented as function of the number of robots. Section 4 presents experiments using a group of small robots and discusses the practical results obtained. Section 5 concludes the paper and presents some further issues.

II. Description of the Algorithm

A. Hypotheses

In this paper, a homogeneous population of agents is considered, i.e. all the robots have the same capabilities of sensing, acting, communicating and processing.

Moreover, the considered task is safe and robust reactive navigation in a clustered environment for exploration purposes. The robots are programmed a priori neither for obstacle avoidance nor for enlarging the explored area, and nor for executing more complex actions like

- Finding a sample
- Picking up a sample
- Returning to the home base
- Dropping the sample into an analyzer

On the contrary, the agents have to find by themselves an efficient policy for performing the complex tasks. The on-line self-learning procedure is based upon the principles of genetic algorithms that allow a faster convergence rate than the classical learning algorithms, as it is shown in what follows. The algorithm's operation is then illustrated by considering the exploration problem, which is simple to evaluate and to implement on real mobile robots. The population size is constant; i.e. no robot will disappear.

B. The Chromosome Encoding

The inputs of the sensorimotor system are composed of five states listed in Table 1. These states can easily be recognized by the proximity sensors of any robot. Of course, more inputs would be available if fuzzy processing was applied. On the other hand of the control system are the elementary behaviors of the robots are given in Table 2. They actuate immediately and properly the wheel motors. An individual of the population considered for the evolutionary learning is the list of connections between inputs and outputs. It encodes thus the "synapses" of the robot's sensorimotor control system. Such states and elementary actions has been choose to guarantee that the behavior is learnable during the battery life.

TABLE 1: DIFFERENT STATES OF AN AGENT'S SENSORY SYSTEM

State 1	No obstacle
State 2	Left obstacle
State 3	Right obstacle
State 4	Front obstacle
State 5	Robot is jammed

TABLE 2: THE SET OF ELEMENTARY REACTIONS

Behavior 1	Go forward
Behavior 2	Turn right
Behavior 3	Turn left
Behavior 4	Go backward

The chromosome of a robot is the concatenation (a string) of N words in a subspace of the {0,1}M space. N is the number of inputs, and M the number of outputs. Of course, a single bit is "1" in each word (Figure 1). The population of robots will evolve following the evolution of the embedded chromosomes under the action of genetic operators. This technique belongs to the genetic connectionism class.

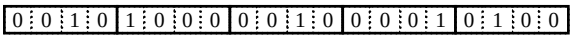

Figure 1: An example of chromosomal string

C. The Operators

Initialization. At the beginning, the various chromosomal strings may be filled either at random or given the same behavior, like a string of "go ahead". For the simulations and experiments reported in this paper, the strings have been randomly generated**.**

Individual Performance Index. In classical genetic algorithms and many other learning techniques, some kind of supervisor exhibits templates, and gives a rating to the agent's resulting behavior. This upper level may also evaluate the "fitness" of each individual of the population with respect to a required performance. The individuals are then ranked (the selection operation) before applying the genetic operators. On the contrary, as we are looking here for a fully autonomous evolution, a capability of local self-evaluation is given to each robot, but it is never conscious of the global population efficiency.
However, following the general principles of self-learning algorithms, each individual (here the agent numbered i) computes its own current reward as follows:

$$R_{N(i)}(i) = (1-\alpha(i))R_{N(i)-1}(i) + \alpha(i)F_{N(i)}(i) \qquad (1)$$

Where

$$\alpha(i) = \frac{1}{1+N(i)} \qquad (2)$$

$N(i)$ is the number of time steps since the beginning of the estimation by agent i

$R_{N(i)}(i)$ is the estimated reward at time $N(i)$

$F_{N(i)}(i)$ is the instantaneous reward at time $N(i)$

In the exploration problem, the instantaneous reward is computed as being the forward distance traveled during an elementary time step. Thus, the current reward resulting from applying the current policy (the chromosome) is the average distance since initialization or since the last change of the agent's chromosome by crossover or mutation. Such a computation automatically penalizes too numerous turns and reverse motions.

Crossover. Most of the solutions proposed by the others investigators call for global communication between all the agents. By this way, the classical genetic algorithm

technique is used: selection by considering the whole population, then crossover of two chromosomes at a periodic average rate. This method suffers from the lack of parallelism since only two agents can be concerned at the same time. It even may require a slackening of the robot moves to allow the complex communication, agent recognition and signal processing.

In order to avoid these major drawbacks, the solution proposed in this paper uses a local and simple communication. This way, any pair of robots meeting each other may perform crossover. The formal conditions are the following:

- The robot-to-robot distance is short
- Both robots have not recently performed a crossover or mutation operation

The first condition ensures the possible parallelism, while the second prevents any robot from changing its current policy before having evaluated it over a significant number of steps.

When crossover is completed, agents i and j have new chromosomes, i.e. policies, according to the probabilities:

$$P(i) = \frac{R_{N(i)}(i)}{R_{N(i)}(i) + R_{N(j)}(j)} \qquad (3)$$

Crossovers are not sufficient to ensure the convergence of the system towards the best solution, especially when the population size is small. In that case, the agents may be trapped by a local maximum, and the population no longer evolves from that common state. As it is usual, mutations are necessary to escape from local extrema and to explore a wide domain of the chromosomal state space.

Mutation. In the classical genetic algorithms, mutations are performed at random. In our application, this could be an important drawback since we are looking for on-line learning. It cannot be admitted that a robot can change its policy to a far less efficient one and waits for a long time before re-improving it by a new mutation or by crossover.

To solve these problems, the following mutation strategy is adopted:

- The agent has not performed a previous mutation recently
- The agent's fitness is low

Notice that again such a method allows multiple simultaneous mutations of several agents.

As for crossover, the first condition ensures that the robot has had enough time for computing a good estimation of its own fitness. The second condition prevents from altering good policies into worse ones. We adopt a probability of mutation that depends on the individual performance index. It ensures that the worst policies are more likely to mutate, while the best ones possess a non-zero probability for escaping a possible local maximum.

D. Demonstration of Convergence

The architecture proposed here aimed at obtaining an on-line unsupervised learning procedure working faster than the other methods. Among these, are Q-learning and artificial neural network learning, and this is not an exhaustive list.

In Q-learning, an equation similar to (1) is used to update the expected reward, which is an unknown stochastic function. The γ coefficient is not evaluated like in equation (2), but is rather selected as a relaxation coefficient $0<\gamma<1$. This way, the system "forgets" old experiences and gives more importance to the recent ones. This is a first drawback: Q-learning acts like a greedy algorithm instead of integrating previous experience, and thus being more robust to unlikely situations. Secondly, this method is very time-consuming since almost all the possible states of an agent with respect to the others and to its position in the universe must be explored.

Using artificial neural networks for designing a mobile robot planner/navigator is very appealing. Once programmed, this sort of architecture provides the right behavior corresponding to each environmental situation detected by the sensors. It acts like an associative memory. However, a problem still appears in the learning, which can be very long. Each agent must have explored almost all the possible situations, thus requiring some kind of simulated annealing to be added to escape from the local maxima.

In the scheme proposed here, the mean performance index of each robot, its fitness, is a non-decreasing function thanks to the rules adopted for crossover and mutation. The probability of two robots to meet increases roughly linearly with the size of the population. However, due to the lower binding limit, which we impose to the time between two successive policy changes, it is evident that the rate of such changes reaches an upper limit when the number of agents is great. The convergence and the minimum time of the learning phase are confirmed by the simulations and experiments described in the following sections.

III. SIMULATION

A. The Simulator

The proposed algorithm was first evaluated via Matlab simulation. The source code is made of two main parts: the environment composed of fixed obstacles in a bounded rectangular area, and the moving agents. The agents never compute their absolute position and orientation. The knowledge of these variables only serves for the displays. On the contrary, the real sensory system is simulated and provides the agent with its local perception. The real wheel control is also simulated. An example of display is shown on Figure 2.

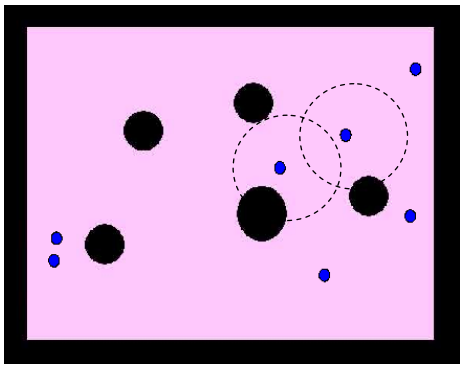

Figure 2: An example of simulation with circular obstacles (in black)

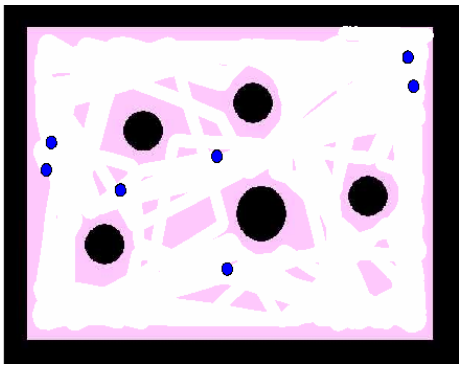

Figure 3: The explored area

During a simulated elementary sensorimotor cycle of time, each agent updates its current state, and either continues its current policy or performs mutation (if allowed) or crossover (if possible). To make crossover possible, the sensory protocol differentiates another agent from a passive obstacle, and is able to compute the robot-to-robot distance. The two dashed circles surrounding two robots in Figure 2 show the sensor ranges. If crossover is allowed, it takes several cycle times due to the need for communicating information. Else, each of the two robots considers the other as an obstacle.

B. Simulations Results

The first step is to set the simulation parameters. The communication range is measured on real robots and scaled to the simulated environment. Then it is necessary to adjust the coefficients in the probability function for mutations. The answer was found thanks to simulations. If the delay is short, the mutation occurs frequently. The space state is explored very quickly, but the performance estimation is erroneous. On the contrary, if the delay is long, the space state is explored very slowly, but the estimation is very good. The delay between two crossovers was also studied. The conclusions are close from the mutation's ones. If the delay is too short, agents will always crossover with the same agent. On the contrary, if the delay is too long, it takes a lot of time to explore the policy space state. The minimal number of cycles is chosen to fulfill the following condition: two agents having the optimal policy will not meet two times successively. Here 300 cycles are necessary.

Figure 3 shows the end of a simulation. The first observation is the explored part of the environment (in white). It results from the obtained emergent behavior due to the reactivity of the multi-agent system. We have also checked that the optimal solution, given in Table 3, is always reached whatever the initial conditions and the shape of the environment.

At the end of experiment, the chromosome encodes the optimal sequence described on table 3. In most cases, the observed evolution is the following:

1. At the beginning, the agents try random strategies. A collision quickly occurs, and the robot is jammed. The average traveled distance drops very fast, and a mutation occurs soon. Mutations stop when the behavior 4 (go backward) is associated to the state 5 (robot is jammed).
2. The agents are free in the environment, and one of them is likely to associate the behavior 1 (go forward) to the state 1 (no obstacle). It generally travels a long distance in the environment and propagates, by crossover, its chromosomal string to other agents.
3. As many agents move fluently in the environment, more crossovers occur. The optimal policy is thus given to at least one agent. Its own fitness grows quickly.
4. As soon as such a robot performs the best sequence, it travels all over the environment and meets more agents than the other ones. By doing so, it transmits its string to many agents, and all the population quickly performs the best behavior. The non-null mutation rate imposed in "good" populations ensures that the absolute maximum is obtained.

Another important parameter is the size of the population, i.e. the number of robots. Intuitively, the more agents there are, the faster the best solution is found. This hypothesis is true, if the condition of convergence is: "at least one agent finds the optimal sequence". We decided to stop the simulations when each individual would have performed the chromosomal sequence described in table 3. For each number of robots, twenty simulations were realized, and the average number of cycles was recorded. The results are shown on Figure 4.

It can be seen on figure 4 that, when less than ten robots are used, the average convergence time decreases with the number of robots. With more than ten robots, the convergence time becomes constant. The explanation is that when the number of agents is high, the best behavior is quickly found and propagated.

TABLE 3: THE OPTIMAL CHROMOSOMAL SEQUENCE

1 : No obstacle	1 : Go forward
2 : Left obstacle	2 : Turn right
3 : Right obstacle	3 : Turn left
4 : Front obstacle	2 or 3 : Turn left or right
5 : Robot is jammed	4 : Go backward

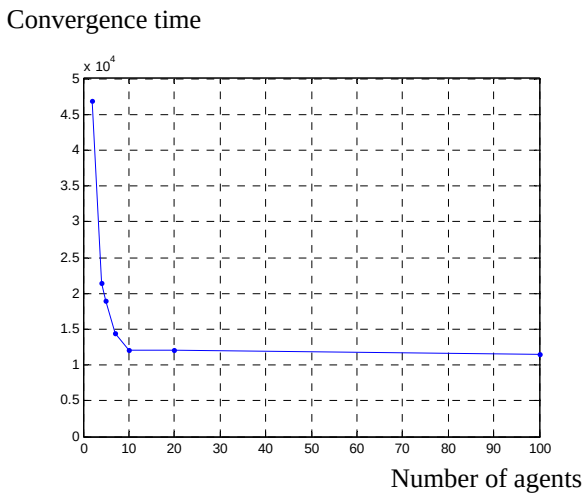

Figure 4: Convergence time vs. number of robots

IV. EXPERIMENTS WITH REAL ROBOT

To enhance our study, the proposed architecture has been implemented on a set of mobile robots.

A. Robot's Hardware

The robot performing our experiments is Type 1 [17], a mobile robot developed at LIRMM. Type 1 has many of the characteristics required by the evolutionary approach to autonomous robot learning. It has a 10 cm-height and 13 cm-diameter cylindrical shape (figure 5). Two wheels actuate it. Two small passive ball-in-socket units ensure the stability in place of usual castor-wheels. DC motors equipped with incremental encoders (352 pulses per wheel's revolution) control the wheels. The encoders are used for both speed control and measuring the performance index by odometry. The robot is surrounded with 16 infrared emitters and 8 receivers. The sensors use a carrier frequency of 40 kHz for a good noise rejection. These sensors are also used to communicate between agents. A communication protocol has been developed to differentiate obstacles from other agents: if a signal is received when the robot has stopped emitting for a short period, this means that another agent is close to it. An embedded PC (80486 DX with 66 MHz clock) manages the robot. Control to sensors and actuators is transmitted by the PC104 bus.

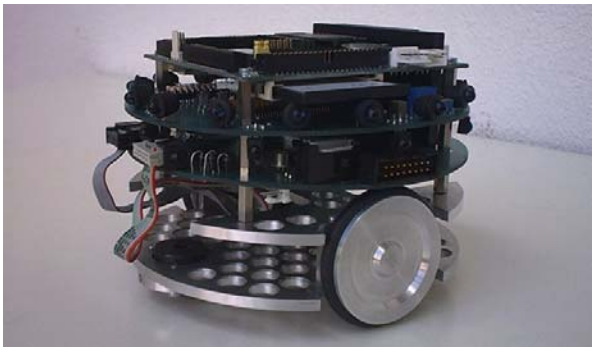

Figure 5: Type 1, the mobile robot

B. Experimental Results

The experimental site is shown on Figure 6. It is 4.80 m long and 3.60 m wide. Four autonomous robots are used in the experiments, and obstacles can be moved, added or suppressed. The maximum speed of the robots is 1 m/s but the speed has been limited to .3 m/s to prevent eventually hard collisions. The sensorimotor cycle time is about 15 ms. The range of the infrared system used for sensing and communicating can be adjusted. A typical value of .5 m has been adopted: it ensures here the required compromise between the needs for enough crossovers and for safety with respect to collisions. Following these numerical data and the experience gained during the simulations, the minimum time between two successive crossovers has been set to 3 seconds.

Figure 6: A view of one experiment

V. CONCLUSION AND FUTURE WORKS

The self-learning architecture proposed has been implemented and tested using a group of real mobile robots. The experimental results showed that the

implementation behaves as predicted by the computer simulations in the case of learning collision-free navigation. Thanks to crossover, several individuals can share the information that guides learning. Such a solution is thus different from either supervised learning or simple local rewards. The learning is quickly efficient and converges towards the best-known solution. Implementation of other basic behaviors is currently under study, for example object picking by a mobile manipulator. A further step will concern the use of similar principles for making the robots find automatically the best sequences of elementary behaviors in complex cooperative tasks.

ACKNOWLEDGMENTS

This research was supported in part by the LIRMM (Montpellier, France).

REFERENCES

[1] R. A. Brooks (1986) "A robust layered control system for a mobile robot", *IEEE Trans. on Robotics and Automation*, volume 2, pp. 14-23.

[2] D. Floreano, and F. Mondada (1994) "Automatic Creation of an Autonomous Agent: Genetic Evolution of a Neural-Network driven Robot", *SAB-3*, Brighton, pp. 421-430.

[3] H-S Lin, J. Xiao and Z. Michalewicz (1994) "Evolutionary Navigator for a Mobile Robot", *ICRA'94*, San Diego volume 3, pp. 2199-2204.

[4] J. Xiao, Z. Michalewicz, L. Zhang (1997) "Adaptive Evolutionary Planner/Navigator for Mobile Robots", *IEEE Transactions on Evolutionary Computation*, volume 1, No. 1, pp.18-28.

[5] L. E. Parker (1998) "Alliance: An Architecture for Fault Tolerant Multirobot Cooperation", *IEEE Trans. On Robotics and Automation*, volume 14, No. 2, pp. 220-240.

[6] T. Balch and R. Arkin (1994) "Communication in Reactive Multiagent Robotic Systems", *Autonomous Robots*, volume 1, No. 1, pp. 27-52.

[7] O. Simonin, A. Liégeois and P. Rongier (2000) "An Architecture for Reactive Cooperation of Mobile Distributed Robots", *DARS-4*, Knoxville, pp. 35-44.

[8] E. Yoshida, T. Arai, M. Yamamoto, and J. Ota (1998) "Local Communication of Multiple Mobile Robots: Design of Optimal Communication Area for Cooperative Tasks", *Journal of Robotic Systems*, 15(7), pp. 407-419.

[9] D. E. Goldberg (1989) "Genetic Algorithms in Search, Optimization and Machine Learning", Addison-Wesley.

[10] I. H. Harvey, P. Husbands and D. Cliff (1992) "Issues in Evolutionary Robotics", *SAB-2*, Hawaii, pp. 364-373.

[11] J. R. Koza and J. P. Rice (1991) "Genetic Generation of both the Weights and Architecture for A Neural Network", *IJCNN-91*, Seattle, 1991, volume II, pp. 397-404.

[12] A. Agah and G. A. Bekey (1996) "A Genetic Algorithm-Based Controller for Decentralized Multi-Agent Robotic Systems", *ICEC'96*, Nagoya 1996, pp. 431-436.

[13] D. Floreano and J. Urzelai (2000) "Evolutionary Robots with On-line Self-Organization", *Neural Networks*, volume 13, 2000, pp. 431-443.

[14] D. Barral, J-P Perrin, E. Dombre and A. Liégeois (1999) "An Evolutionary Simulated Annealing Algorithm for Optimizing Robotic Task Point Ordering", *ISAPT'99*, Porto, pp. 157-162.

[15] O. Pinchard and A. Liégeois (1995) "A Genetic Algorithm for Outdoor Robot Path Planning", *IAS-4*, Karlsruhe, IOS Press, pp. 413-419.

[16] O. Pinchard and A. Liégeois (1996) "Non Deterministic Methods for Robot Path Planning in the Presence of Uncertainties", *CESA'96*, Lille, Robotics and Cybernetics, pp. 593-598.

[17] P. Lucidarme, P. Rongier and A. Liégeois (2001) "Implementation and Evaluation of a Reactive Multi-Robot System", *AIM'01*, Como, pp. 165-170.