AN ADAPTATIVE EVOLUTIONARY ALGORITHM FOR ANT COLONY

L.Gacôgne * **

* LIP6 CNRS - Université Paris VI   4 place Jussieu 75252 Paris 5 France
tel : 01 44 27 88 07   mail : Louis.Gacogne@lip6.fr

** Institut d'Informatique d'Entreprise (CNAM) 18 allée J.Rostand 91025 Evry France

Résumé
Ce rapport concerne une simulation de colonie de fourmis soumise à évolution. Chaque individu est gouverné par un petit réseau de neurones à l'intérieur d'un terrain circulaire où se trouvent présents différents signaux représentés par des points colorés (nourriture, limites, stimuli laissés par les autres fourmis ...) Elles sont supposées chercher de la nourriture pour la rapporter au nid situé au centre du terrain. Sans disposer d'aucune règle pour y parvenir, nous expérimentons un algorithme évolutionnaire afin de sélectionner les mailleurs individus de générations en générations. Dans le but de formaliser un comportement par une fonction mathématique ou un système à base de règles, beucoup de voies ont été imaginées, aussi nous considérons que chaque fourmi ne peut capter les messages que sur les cinq points voisins devant elle. Celle-ci utilise alors son réseau de neurones pour choisir sa prochaine position. L'essentiel de notre algorithme d'évolution consiste à mettre à jour ces réseaux de neurones, nous étudions alors la performance de chaque fourmi et construisons la génération suivante de manière élitiste au moyen d'opérateurs liés à la représentation de ces réseaux.

Abstract

This paper deals with a simulation of an ant colony which is subject to an evolution. Each one of the ants is moving according to a small neural network, in a circular pitch where it is involved in a colored grid indicating different signals (food, borders, stimuli from other ants ...) They are supposed to look for food and carry it back to their nest located in the center of the playground. But they don't have any rule to do that and we experiment an evolutionary algorithm to select the bests individuals generation to generation.
Face to formalise a comportment as a mathematical function or a rule-based system, we can imagine many ways, so we chose to consider an ant located at any point of the playground with only the knowledge about the five neighboor points front of it. The ant is capable to use then his proper neural network to choose the next case will suit it. The key-point of our evolutionary algorithm is to set up those neural networks. So we study the fitness of each ant and build an offspring in an elitist way, with genetic operators linked with the representation of the networks.

Mots-clés Problème du fourragement - Réseaux de neurones - Algorithmes évolutionnaires - Co-évolution

Keywords Foraging problem - Neural networks - Evolutionary algorithms - Coevolution

Introduction

A lot of works deals with collective intelligence and the foraging problem, we can distinguish those where each ant is using a proper sequential algorithm, for instance in the field of genetic programming [Koza 92], the communication-cooperation [Bouron 92] (prey / predator), [Drogoul 93] (foraging problem with cooperation) and [Bonabeau, Theraulaz 94], [. It is obvious that evolution's technics must appear in this topic and we find some works about it. Co-evolution of two robots with standard genetic algorithm was made by [Ghanea-Hercock, Barnes 96] and for the design of fuzzy rules by [Mohammadian, Stonier 98], but co-evolution raises the problem of validity of the fitness, this is disscussed in [Miller, Cliff 94] and also by [Axelrod 87] and [Gacôgne 94] about the prisonner's dilemma problem where evaluation of a strategy is relative to the others, and the whole population of those strategies is moving (may be towards a homogeneous one).
Some authors focuse on the 3-layer neural network to make decision for instance for the problem of obstacle avoidance with Khepera robots [Floreano, Mondada 96] or for robot arm [Moriarty, Miikkulainen 96]. Genetic algorithms have been using for some years for the tuning of such neural networks and the foraging problem [Mataric 93].
The central feature of our work is a simulation of co-evolution in an ant-colony, each agent being directed by a specific three-layer neural network, but we make a separation between generations to keep the bests neural-networks and we intend to begin a discussion about modalities of evolutionnary algorithms to produce not a good solution but a good population for a collective solution.

I ANT COMPORTMENT

The features of each ant is at any moment characterised by a position, a direction, a trace behind it, a score and the knowledge about the food it carryes or not. Moreover an ant is moving thanks to a very simple neural network with six inputs and the neighborhood point chosed as output. The six inputs are first, the boolean fact if the ant has food or not, and the five colors on the nearest neighborhood points front of it (the three front, left and right and the two sides on the grid). The direction of an ant is a random one (at the begining of evolution), that is to say each part of the playground could be visited by them. Why an initial fixed direction ? Because we want the ants close to the nature where they are more or less able to keep their way due to a special sense. Note that this direction is randomly chosen at each birth, that is to say an ant does not inherit of his parent's direction during evolution. It could be in this case, more learning a particular playground than learning a foraging strategy.
We choose a neural network with three layers according to the results of [Cybenko 89, Hornik 89]. The hidden layer has not a fixed number of neurons because we don't have procedure to set it. Aniway experimental results show that it must be greater than the number of inputs [Vurkova 91, Comon 92].
The neural network encoding is a the list of weigths attached to the hidden neurons. So if there are six inputs and $p_1, p_2, ..., p_6$ are the weights from the inputs layer towards one of the hidden neurons, and if w is the weigth (a real number) from this one to the unique output, each hidden neuron will be coded by the sequence of real numbers $(w, p_1, p_2, ..., p_6)$.
For instance the net (right figure) will be the list of 5 neurons, each one being a 4 length list :
((3, 1, 0, 2), (1, 1, 0, 0), (-1, 0, 0, 2), (2, -1, 0, 0), (1, 0, -3, -2))
With an input vector (2, 1, 1), the states of the hidden layer will be (4, 2, 2, -2, -5) and the output will be 3 if the step function is identity.

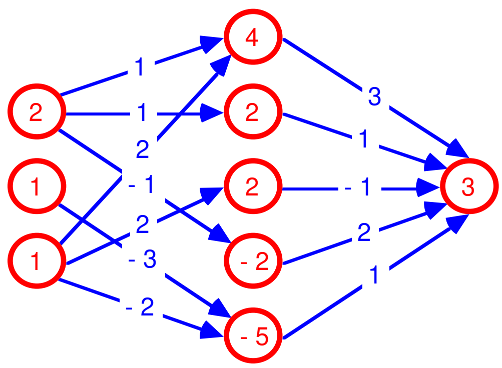

Figure 1 Example of neural network

Initialisation

The field where we make the training is a circular one with a randomly distribution of food on the left and the right of a centered nest where the ants have a gaussian initial position (figure 2).

Transitions

Each time, each ant, in order to move, uses its proper neural network and leave behind it a yellow (or orange if it carries food) pheromon' trace. That trace is in fact the ns lasts positions of its route and ns will be 20 or 30.
In the case where it arrives on a green point and not carries food, then it takes it and modifies its direction for the opposite one. It turn out that the ants are able to follow an approximative direction and face to the ordeal we experiment here, it is the only feature we set on at initialisation.
If it arrives on a border (a cyan point) there is the same change of direction.
If it arrives in the nest when carring food, then we incremente the fitness of this ant and we put a new blue point as a symbol of food in the ant'attic. Each ant has his proper fitness : food quantity brought back in his life.
We define a generation by ne steps (it will be trained from ne = 200 to 1000), and after this "life" for all the population, we apply genetic operators to them to built the offspring population.

II GENERATIONS OF THE ANT COLONY

The method to evaluate an ant is necessary to improve all of them togother, so, when the fitness for each ant separatly, is computed in a session for all the population, we create the offspring population applying genetic operators. We, then, are obliged to evaluate this one. After that we can discuss about strategies : is it better to use an elitist one as a $(\mu + \lambda)$-strategy or a (μ, λ) one [Schwefel 90, 96]

With a number nf of ants, our main algorithm is :

Create a random population of all different chromosoms P_0 and initialize (ng, nv) $\leftarrow$ (0, 0)
Simulation on the population P_0
While nv (number of evaluation of the fitness) < limit, do
    Building of P_1 as applying operators on P_0
    Generation on P_1
    $P_0 \leftarrow$ the nf firsts of sort $(P_0 \cup P_1)$

Where "generation" is the training of all a population during ne=500 steps, where ng (number of generation) is incremented and where the min, max and average of the score are stored.

An adaptative evolutionary algorithm

We introduce a new evolutionary algorithm ESAO (evolutionary strategy with adaptative operators) quite different with the classical one. Our goal is to leave all a familly of different operators to be free to be applied during evolution. That is to say they are applied in order they are able to force evolution as we detail it below. For this, we have a set of several scored operators. On the other hand, we go on exploration to avoid a too big homogeneous population. The algorithm is the following [Gacôgne 94].

1) A chromosom is here a list of genes (this list hast not a fixed length, intialisation is done with a number of neurons 1 + a random integer given by a Poisson law with average 2). The first population P_0 is nf ants with those chromosoms in a random position in the center area and a random initial direction.

2) We have a random population OP_0 of genetic operators those operators are imagined according to the problem. Here we choose :

Mutation The normal mutation of a weight randomly chosen inside the chromosom, thus for example [[a, b], [c, d, e, f]] $\rightarrow$ [[a, b], [c, **a**, e, f]].
Noise It is a gaussian perturbation on a weight, that is to say a non uniform mutation, for example [[a, b], [c, d, e, f]] $\rightarrow$ [[a, **b + 0.03**], [c, a, e, f]].
Neural-mutation A sublist of the chromosom is replaced [[a], [b, c, d]] $\rightarrow$ [[a], **[i, j, k]**].
Creation A new hidden neuron is added. [[a, b], [c, d]] $\rightarrow$ [[a, b], **[u, v, w]**, [c, d]]
Suppression One of the neuron is suppressed (except if the chromosom has the length 1).
Crossover-1 Classical cross-over between two chromosoms restricted to the first level. If [[a, b], [c, d, e], [f, g, h], [i, j, k, l]] and [[p, q, r], [s, t], [u, v], [w], [x], [y, z]] are the parents, the children can be for instance [[a, b], **[s, t], [u, v],** [i, j, k, l]] and [[p, q, r],

[c, d, e], [f, g, h], [w], [x], [y, z]]. To have an homogeneous evolutionary device, we always consider operators from P to P, so we just keep one child.

Crossover-2 Cross-over between two chromosoms, but across the levels, for the squashed list, if [[a, b], [c, d, e], [f, g, h], [i, j, k, l]] and [[p, q, r], [s, t], [u, v], [w], [x], [y, z]] are the parents, the child could be [[a, b], [c, d, **t**], [**u, v, w**], [**x, y**, k, l]] with the same structure as the first parent.

Crossover-elitist The chromosom produce a child with one of the best other chromosom (in the 10% firsts).

Migration-1 : All is replaced exept the first element (the first hidden neuron with its weight).

Migration-2 : The half (in length) of the chromosom is randomly replaced.

Migration-all : Everything is changed, this operator is used at the begining to constitute the first population and also during evolution, to explore the space.

The main idea of ESAO is each operator has an attached score initially zero, and then its algebraic capacity to better a chromosom when this operator is applied is cumulated.

3) Each generation t, the operators of OP_t are sorted and applied with their rank to the sorted population P_t. When an operator is applied to an element, ESAO eliminates the worst in the couple of child and parent in the classical optimization problems. But here, as we have a co-evolution, the offspring is first built before its evaluation during a separate generation and elimination will be executed after.

4) If there is no amelioration, any improvement for the whole population, then we make a reinitialization of OP_{t+1}. In the other case OP_t is sorted and a new operator in the same kind of the best one is pushed inside replacing the worst. Thus we have an adaptative familly of operators and we avoid this family uniform in the case where a "good" operator takes all the place in OP.

5) We stop the process after a fixed number nv (5000 or 10000) for evaluations of the fitness.

III DESIGN OF THE MODEL EXPERIMENTATION

As we said before, the main goal is to collect food distributed (with random gaussian distribution) in two sources (with 5000 points of food on each) left and right to the nest.

How to compare different sessions ? We can define a ratio of the food quantity nn brought by ant and by the number of steps during a life, and then it is possible to focuse our attention on this ratio function of the generations.

But, after some experiments, it happens that the number ne of steps during each generation must be fixed because a too short one does not give enough possibility for the ants, and too big life makes simulations very long.

So we defined a generation by a simulation of nf = 50 ants living during ne = 500 steps.

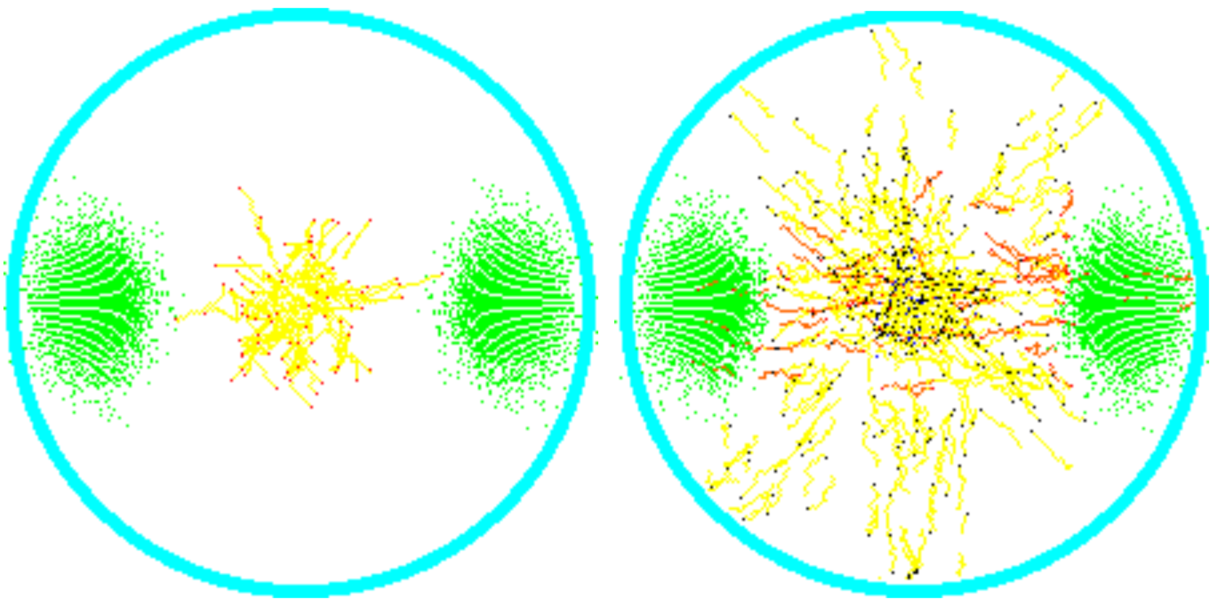

Figure 2 We show the experiment playground at the begining of a generation with 50 ants.

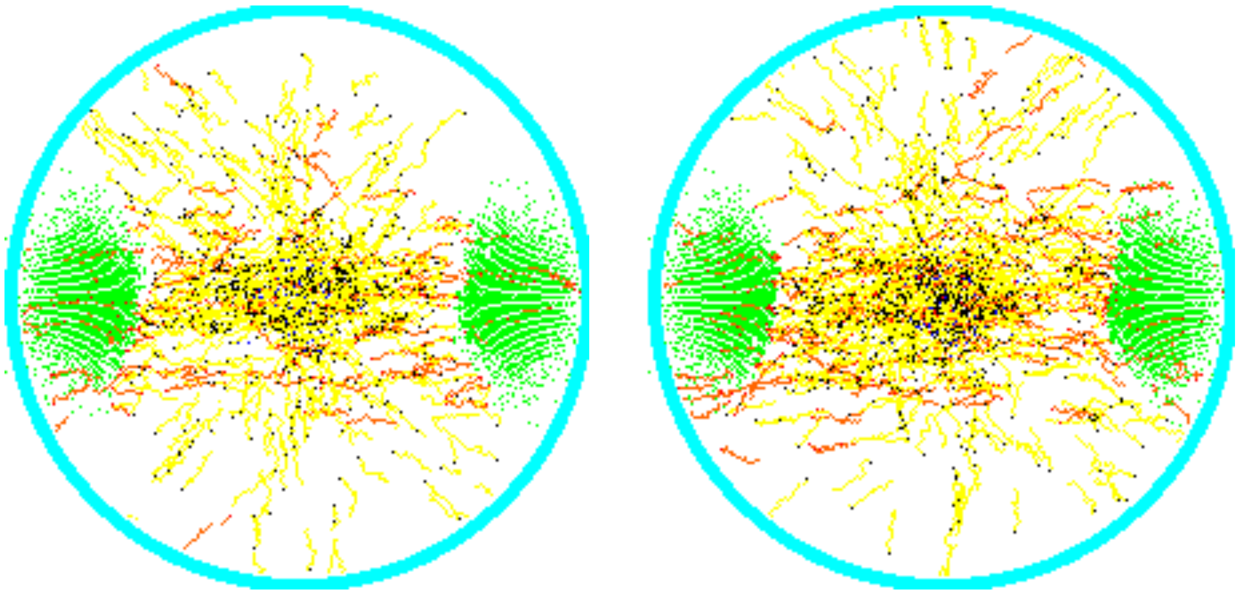

Figure 3 Constitution of two roads and exploitation with 1000 ants at the end of evolution of the colony.

IV EXPERIMENTAL RESULTS

1 Comparisons between the two evolutionary methods

We did training with nf = 50 ants living during ne = 500 steps. So each generation could have a "rentebility" defined as 100nn / (nf.ne) where nn is the number of food'grain. Here we only shows this quantity nn.

We make a comparison with the evolution strategies ($\lambda + \mu$)-ES [Schwefel 90, 95] where a population of size μ produce an offspring with size λ (to each individual is applied several times genetic operators). In that strategy, parents and children are mixed to keep the μ best of them (in the (λ, μ)-ES only the μ best of the offspring is kept). Observation shows that in (λ, μ)-ES, better results reached with a ratio λ/μ around 7 [Bäck 95, 97].

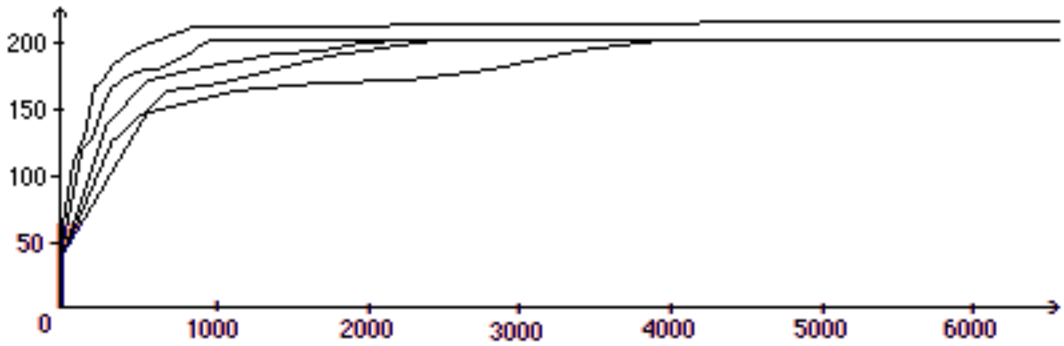

Figure 4 Food quantities brought by the colony face to the number of fitness evaluation (averaging of all 20 results).

The five different curves are from the top line to the lowest line, first with our strategy ESAO and the four lowest with some ES where λ/μ = 2, 4, 6, 8. So, our strategy is faster and it seems that we don't have the same remark as Bäck, but we must have a mention that we are in a very particular field of evolutionary applications. We did similar conclusions with mathematical tests as Rastrigin or Griewank' functions [Gacôgne 99].

2 A look over the most important operators

For our algorithm ESAO, we count how many times each operator in the familly was in the head of OP and was duplicated. We already did some observations about experiments about fuzzy logic problems [Gacôgne 97] to find that very often the operators which bring the highest perturbation like "migration-all" have a good score. Here the list is quite different :

Noise : 25, Suppression : 18, Crossover-2 : 10, Crossover-1 : 9, Mutation : 9, Neural-mutation : 8, Migration-2 : 7, Crossover-elitist : 7, Migration-1 : 5, Creation : 4, Migration-all : 1.

3 Interpretation of the best neural networks

It is quite difficult to have an interpretation of neural networks because we must improve it in several configuration. The bests networks have 4 or 5 neurons in their hidden layer, of course they choose the green neighboor if there is one, if not the orange or yellow one. For each list (neuron), first number is the weigth from it to the output, the second is the weight from the particular input (1 if carrying food, 1- else). The five lasts are the weights applied to the five color code. For that encoding we took an arbitrary hierarchy -3 for the borders (cyan), 0 for white, 1 for the yellow pheromon, 2 for blue (attic) or red ant, 3 for the pheromon indicating food and 4 for the ant which carries, 5 for the food (green).
For instance, one of the best score (9 food unit remained) neural network has five elements in the hidden layer.

[0.524895900435; 0.390520858443; -0.735615174521; -0.909020591857; -0.700367145645; 0.0883202354207; -0.438803255987];
[0.908984863045; -0.193909394657; -0.0922589229251; 0.39549633619; 0.889230329485; 0.251248920351; 0.748954707275];
[0.180999970627; 0.707216658667; -0.319929745033; -0.476022532311; 0.667201972644; -0.24509533371; 0.474405322738];
[-0.634120043102; 0.447993822707; 0.289096323962; -0.908526543514; -0.636582417397; -0.357773226012; 0.41524087226];
[0.909317261394; -0.133309407528; -0.793352459634; 0.920300259764; -0.920601178737; -0.864487752607; 0.0875257374316]]

4 Experiment in an other ground

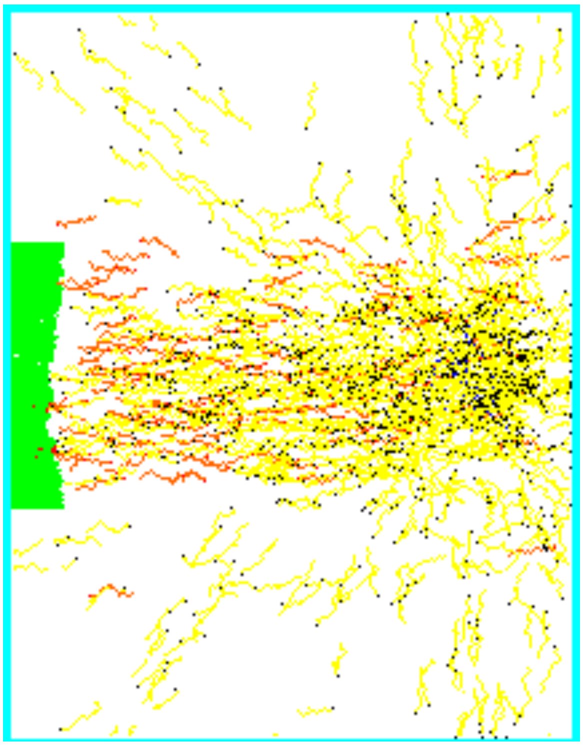

Figure 5 Comportment in an other playground, of the 1000-sized population raised after a 50-generations evolution in the first circular one with two food sources.

Conclusion

As we already note in previous works (multimodal mathematical functions and complex problems in fuzzy logic) our strategy is faster counting the number of fitness evaluations to reach a goal. We also verify it here, in the conditions of trial, but we have to say we realized a very particular application and we, now, have to do more general proofs.

The three layer neural network is a very popular learning paradigm but we intend to use in the future same experimentation with fuzzy logic, to find relevant rules to make a matching with the color distribution.
Nevertheless, we think that there are few sample rules in a such foraging problem, and our strategy is able to find them, but the most difficulty is, may be, more located in technical problems as the choice of a meaningful playground, perhaps in the color encoding or which features we really want to simulate, than in the choice of algorithms.

References

Axelrod R. *The genetic algorithm for the prisoner dilemma problem.* p.32-41 in Genetic algorithms and simulated annealing, Morgan Kaufmann Pub. 1987

Bäck T. *Evolution strategies : an alternative evolutionary algorithm,* Artificial evolution, Lectures notes in Computer Science 1063, p.3-20, 1995

Bonabeau E. Theraulaz G. *Intelligence collective,* Hermès, 1994

Bouron T. *Structures de communication et d'organisation pour la coopération dans un univers multi-agents,* Thèse Université Paris VI, 1992

Comon P. *Classification supervisée par réseaux multicouches,* in Traitement du Signal vol 8 n°6 p.387-407, 1992

Cybenko G. *Approximation by superpositions of a sigmoidal function,* Mathematics of control signals and systems vol2 n°4 p303-314, 1989

Drogoul A. *De la simulation multi-agents à la résolution collective de problèmes,* Thèse Université Paris VI, 1993

Floreano D. Mondada F. *Evolution of plastic neuro-controllers for situated agents,* From Animals to Animats 4, p.402-410, 1996

Gacôgne L. *Une extension floue du problème itéré des prisonniers,* Rapport Laforia 94/15, 1994

Gacôgne L. *About the fitness of fuzzy controllers whose rules are learned by genetic algorithms.* EUFIT p.1523-1531, Aachen, 1994

Gacôgne L. *Eléments de Logique floue,* Hermès 1997

Gacôgne L. *Les algorithmes évolutionnaires : vers des opérateurs adaptatifs*. Rapport interne LIP6 99/21, 1999

Ghanea-Hercock R. Barnes D.P. *An evolved fuzzy reactive control system for co-operating autonomous robot,* From Animals to Animats p.599-607, 1996

Heudin J.C. *La vie artificielle,* Hermès 1994

Hornik K. *Multilayer Feedforward Networks are Universal Approximators,* Neural Natworks vol2 p359-366, 1989

Koza *Genetic programming I,* (chapter 12, and Genetic programming II, 1994) MIT Press, 1992

Mataric, *Designing emergent behavior : from local interactions to collective intelligence,* From Animals to Animats 2, 1993

Miller G.F. Cliff D. *Proteau behavor in dynamic games : arguments for the co-evolution of pursuit-evasion tactics,* From Animals to Animats p.411-420, 1994

Mohammadian M. Stonier R.J. *Hierarchical fuzzy logic control,* Proceedings of IPMU, 1998

Moriarty D. Miikkulainen R. *Evolving obstacle avoidance behavior in a robot arm,* From Animals to Animats p.468-475, 1996

Schwefel H.P. *Systems analysis, systems design and evolutionary strategies,* System analysis, Modeling and Simulation vol.7 p.853-864, 1990

Schwefel H.P. *Evolution and optimum seeking.* Sixth generation computer technology series. Wiley, 1995

Vurkova V. *Kolmogorov's theorem is relevant,* Neural Networks vol3 p617-622, 1991

Zaera N. Cliff D. Bruteu J. *Not evolving collective behaviours in synthetic fishs,* From Animals to Animats p.635-644, 1996