Martin Pernollet
64125A
Erasmus Student

Use of genetic algorithms and interactions to auto-organize an adaptive community of fuzzy-logic-based robots

Project Documentation

SUMMARY

Introduction

Many recent research projects put attention on the use of genetic algorithms for solving different kind of organization problems. In industrial cases of robotics, evolutionnary methods are used to improve systems in the way they achieve various tasks, but also to provide adaptative properties to the system. Genetic algorithms can for example be applied to set a whole community of robots' parameter that offers some clever properties to the community to enhance its mission success.
After some previous research about chaos used as a ressource for artificial intelligence, it seemed interesting for me to explore genetic algorithms. Actually, these two fields lead to the same interest, the so-called *theory of complexity*. This theory states from observation of nature construction (molecules, universe, organic life) that all complex structure stability comes from disorder at a lower level. For example creation of stars comes from chaotic movement of interstellar particules.
Chaos becomes appealing for AI because of the complexity's concept : "disorder creates order". Chaos seems to be one possible key for an *emerging* artifical intelligence in a *bottom-up* approach, by opposition to pre-defined *top-down* artificial intelligence (expert systems, etc). Genetical algorithms afford an interesting point of view to understand complexity in artificial life evolution, since selection methods seam to create the paths to complexity in biology.
For these reasons I felt interested to study how a community of robots auto-organize when their decisions module is submitted to an evolutionnary algorithm.

This experience allow to observe a community of virtual robots surviving in its virtual environement. The robots have some ressource to survive (foods, etc) and some tools to consume it : detection of food, and eat. In order to give a choice in the individual robot's behaviour, the robot has the ability to move or do nothing.
Robot's behaviour is defined by a set of fuzzy logic rules that specify how to react to different situations, state or stimulation (being hungry, see food, etc). We consider a set of rules as a DNA data set : each rule is a chromosome and conditions and actions are genes. We can then apply the mecanisms of crossover, mutation, and natural selection on each rule. The genetic algorithm used here will enhance the effiency of each individual robot in its environnement made of moving foods and made of other robots having the same needs. Each robot has an energy state that is used to define its own fitness among other robots.
We add a cooperation protocole between robots : good robots (best robots of previous generation) specify to new randomly generated robots where is the best starting point for the mission. Since these parents where choosen because they find the place where the best fitness can be obtain, we use an *education* procedure to get an adaptation of the whole robot' specie to environnement's changes.

Evolutionary process is then applied on two different way. First by adapting each robot's fitness to environnement through generations. The other way consist in educating new units to find where best fitness has been recently found.
This paper will first present the fuzzy logic engine used by one robot and how it is embeded into a DNA organization. The second part will explain how the genetic algorithm is set to enhance the fuzzy logic reasoning method to fit environement situation. A third part will present interactions between robots and how they can enhance efficiency of the classical evolutionary method. In the fourth part, we will comment gene repartition and explain how we can tune the community to inhibit or encourage various startegy emergence. The fifth part will explain the concept of emergence in this project.
As a conclusion we will explain why we consider this work as an interesting method for industrial cases, and propose some possible applications in robotics.

Keywords : Genetic algorithm, fuzzy logic, emergence, chaos, collective intelligence.

I – A fuzzy logic engine to define robot's behaviour – Individual unit level

In this project, our purpose is to make some experiments about robots' interactions in collective task achieving. We first consider each robot as a single autonomous unit, able to perform some actions thanks to a decision module.
We called our units 'robots' , but we could consider it more simply as bacterias. Our 'robots' are virtual but simulate how they could interact if they were real physical entities performing a mission in an changing environement.

The fuzzy logic reasoning method

Each decision process made by the robot is done through the use of fuzzy logic. This will allow the robot to use various simultaneous stimuli from environement (or from its own state) as input, to perform one reaction, i.e. decision as output. Fuzzy logic is specialy usefull to simulate fastly and easily parallel systems from natural inspiration like neural networks, because it deals with several inputs, outputs and partial informations to take decisions.
We considere as a state (input): I_SELF_ENERGY
We considere as a stimulus (input): I_FOOD_SEE
We considere as actions (output): O_EAT, O_MOVE, O_LOOSE

Link between stimuli and response is made by a set of fuzzy logic rules like this 2-rules example set:
IF (I_FOOD_SEE(SURE)) *AND* (I_SELF_ENERGY(LOW)) *THEN* O_EAT *at C%*
IF (I_FOOD_SEE(NOTSURE)) *AND* (I_SELF_ENERGY(HIGH)) *THEN* O_MOVE *at C%*

In our simulation, we define a chromosome as a set of 2 rules to offer just one choice to the robot and simplify analysis (in last part, we'll give some idea about expected opportunities of using more rules). Each rule produce an output wich *'certainty'* parameter is expressed by the value *C* %. Selection of the appropriate behaviour is performed by analyse of highest certainty's response *C*.
To avoid logical choice between rules when several rules have the same certainty, we add a random noise to each output so that: 0% < output < 100% ; - 0.5% < noise < 0.5% - C = output+noise

This simple example set could be good to handle decision between eating its own food or moving somewhere else to find other foods. However our purpose is not to set by ourself a whole set of rules, but let it start randomly and improve through evolutionnary process. Crossover and mutation will then make the set (i.e. chromosome) evoluate from generation to generation.

Considering a rule as a chromosome, and conditions/actions as genes

To enhance each unit efficiency, we consider a rule as a chromosome. We then apply crossover mechanism by switching rule's condition cr_i of robot r_0 with rule's condition cr_{i+1} of robot r_1 considering every even i.
The crossover and mutation mechanism will be explain in the following part. We'll after before how we enhance evolution growth through education of new generation by old generation by introducing cooperation methods.

II - A genetic algorithm

We will define here how we apply crossover, mutation and selection. A robot decision module is simply produce by a set of fuzzy rules. The *crossover operation* combinate half of each rule from a robot, and the other half from an other one to offer regular variations. *Mutation* is then performed by modifying randomly some fuzzy variables' tests (i.e. linguistic tests on variables) or by modifying the resulting action in the rule.
The use of evolutionnary principles will be usefull in our case to select decision modules that best fit to situation and will help to create an adaptative thinking society.

Fitness

The fitness function is really simple : maximising robots energy. It means that the robot can decide a strategy between moving, eating or doing nothing considering environement's situation (with or without food). At the end of a generation, we just do the ranking of all robots by energy, keep the 25% of the bests (out of 20) to copy them directly to next generation, keep the 50% of following best robots to apply crossover and mutation, and generate randomly 25% of new robots.

Crossover

The crossover operation consists in using two chromosomes to cross some informations from both to create a child chromosome. We use this concept here to create evolution in the robot reasoning module. This procedure is really important because it uses most of the 'bricks' (from fuzzy logic grammar) to combinate them differently to enhance fitness or to adapt to different situations.

If we apply crossover between the rules of two *1-rule*-robots like:
R[0] *IF* (I_FOOD_SEE(SURE)) *AND* (I_SELF_ENERGY(LOW)) *THEN* O_EAT *at C%*
R[1] *IF* (I_FOOD_SEE(NOTSURE)) *AND* (I_SELF_ENERGY(HIGH)) *THEN* O_MOVE *at C%*
We obtain two new robots like:
R[0] *IF* (I_FOOD_SEE(NOTSURE)) *AND* (I_SELF_ENERGY(HIGH)) *THEN* O_EAT *at C%*
R[1] *IF* (I_FOOD_SEE(SURE)) *AND* (I_SELF_ENERGY(LOW)) *THEN* O_MOVE *at C%*

We apply the same principle for each rule of the robot' specie in *n-rule*-robots.

Mutation

There is two different possible mutations:
Mutations on variable test : I_FOOD_SEE(SURE) can mutate to I_FOOD_SEE(NOTSURE)
Mutations on resulting action : O_EAT can mutate to O_MOVE
In our test mutation rate on variable test is 1/2 and mutation on action is 1/12.

By introducing communication abilities in the community has discussed in the following part, we'll discover that we can enhance efficiency of this classical evolutionnary model.

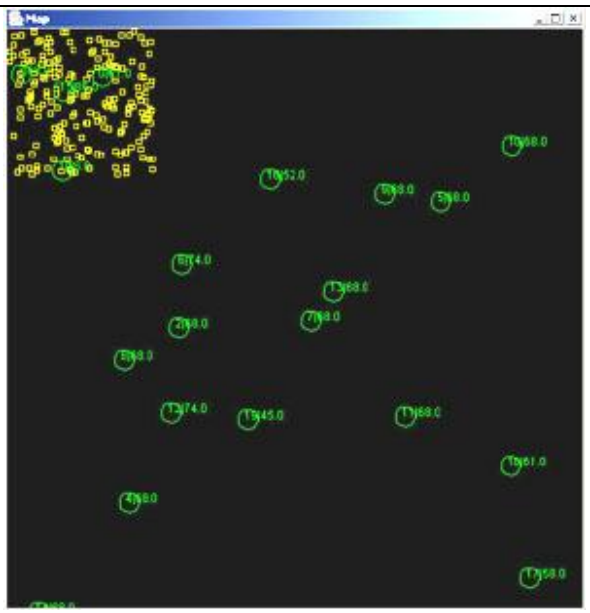
III – Driving evolution through education

The notion of community should be seen as a mean to lead each individual robot more efficiently to the given collective goal (e.g. collecting garbages). We add to evolutionnary methods a process of education in our simulation that request communication between generations to improve community's performance.
After many tests we observed that robots became adaptative to environement's changes through evolutionary algorithm *and* communication between performing units. Here we consider that our robots have to find food because this problem can be easily applied to other real cases (cleaning, exploration, examinations, etc).

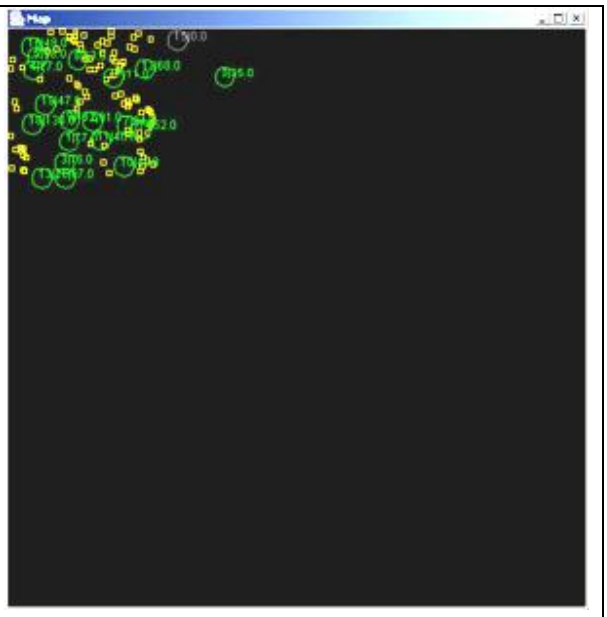
Interaction creates global adaption

In our model, elite of one generation tells to new generation where is the best place to enhance fitness. This means that elite educates new units to locate geographically the food. It simply imply to give to newly randomly generated robots (worse robots in the previous generation) the position of the best robots at the end of the previous generation. This is a way for the whole community to transmit knowledge to all units concerning the place where best fitness can be found, i.e. *where is the goal*, or even *where is the solution*.
All best robots keep their parent's previous position. All new robots (randomly generated) start where the best parents died.

To illustrate efficiency of collaboration, we simulate with and without education process until the 15th generation. When there is collaboration, the community becomes adaptative and is able to fit food's region, while non-collaborative communities don't focus on food place.



Robots in the area without interactions



Robots surrounding foods thanks to interactions

IV – Observations on auto-adaptation of community

General observations about gene regulation

In our first tests, we only use 3 possible actions that provides or supress point(s) to robot's energy (i.e. robot fitness): O_EAT = +2 ; O_MOVE = -1 ; O_LOOSE = -2. We consider that O_LOOSE is a negative action, while O_EAT is a good action that increase robot's fitness. After many tests with different parameters (population size, etc) we observe the following similarity :

We consider a community of S+N robots:

- S is the number of best robots elected to be used in the new generation (copy and crossover)
- N the number of new randomly generated robots
- R the number of rules per robot.

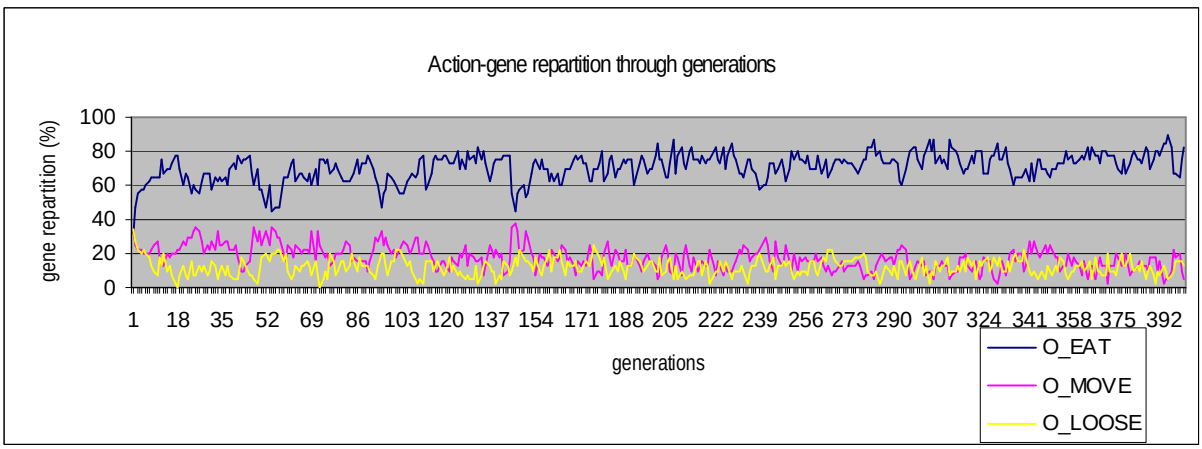
We observed that:

- There will be S*R positive action-gene in the community. (+-10% due to randomness)
- There will be N*R negative action-gene in the community.

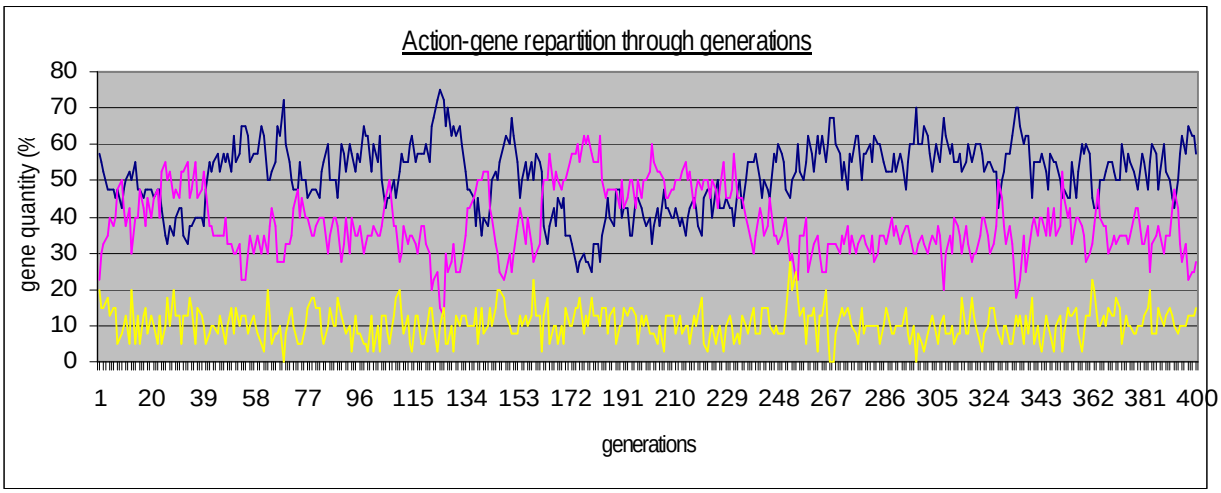
Considering that *positive action-genes* enhance fitness, *negative action-genes* decrease fitness and *neutral action-genes* don't affect fitness.

Tuning adaptation through a punition/congratulation method

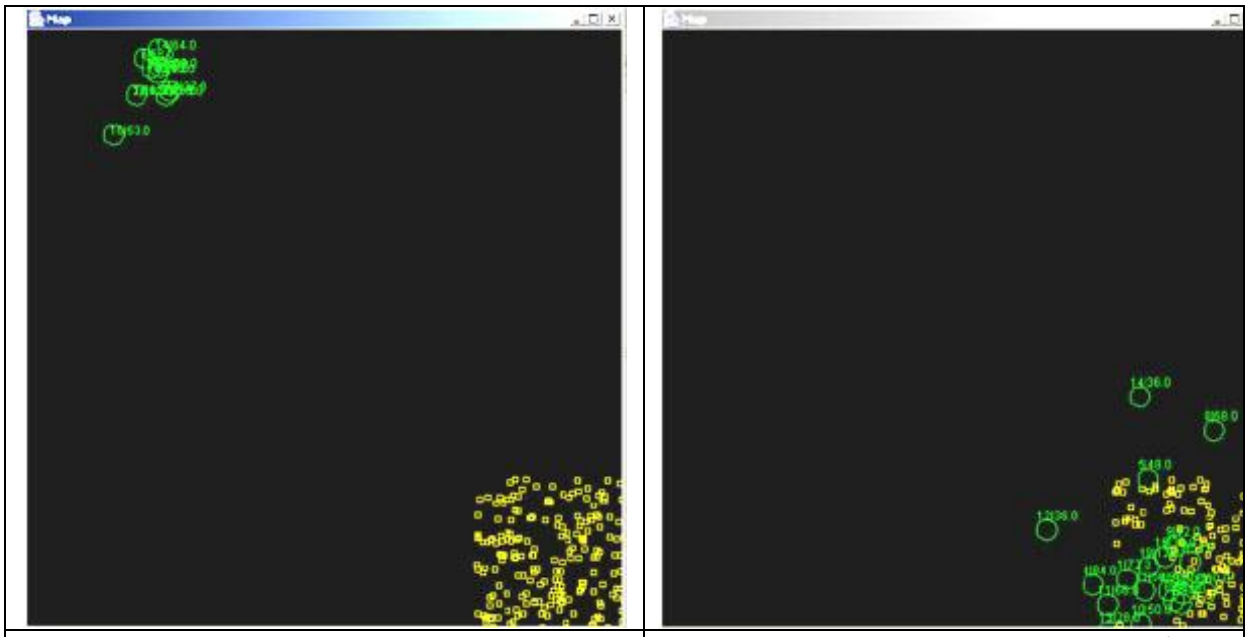
Each time the robot takes a decision or acomplish an action, its fitness evoluates thanks to a pre-defined influences : each operation (deciding, O_EAT, O_MOVE and O_LOOSE) cost or afford to fitness. By setting negative or positive values for each action performance, we encourage evolution to use or reject specific kind of behaviour.
We experienced to suddenly change food position in environement after 100 generations, to see how the community adapts to continue its mission, i.e. to find food again and live around it to consume it.
The whole simulation use 400 generations, 20 robots, keep 25% of best robots for next generation, apply crossover and mutation on 50% of best following robots, and generate randomly 25% of new robots. In adition, the 25% of best robots tells to the 25% of new robots where to start mission, i.e. where a good fitness region can be found.
In the first simulation, we set O_EAT=2, O_MOVE=-1, O_LOOSE=-2. These values mean for example that each time the robot will do the 'O_EAT' action (i.e. output), fitness will increase by the value of 2.
The resulting community behavior doesn't adapt to food position change because the action of moving is considered as an inhibitied action preset to tune the community. The following figure shows that gene repartition doesn't adapt to food position change because 'O_MOVE' strategy can't emerge.



If we set O_MOVE=0, we consider that the action of moving should not be punished (even if punished actions can actually appear in robots' strategy) and is only a *neutral action-gene*. The community then adapts to the same environement's changes than before : robots are able to change their strategy to move, instead of eat, and be *moving robots*. When they find again the food zone they become again *eating robots*.
This adaptation process can be seen on the graphic bellow. The food position change appear at the 100th generation. From this point, the specie of robot will slowly evoluate to be mainly made of *moving robots*. During the simulation, we observed that food was identified by the whole community at the 250th generation, as shown by the repartition's graphic while lines cross again to reveal a *eating robot* majority.



The following figures illustrate how community has muted at the 210th generation : specie with inhibition on 'O_MOVE' action-gene adapts to situation and learns to survive without food. On the other hand, the specie without inhibition achieve to mute to *moving robots* to find again the food and continue mission.



Inhibition on O_MOVE action leads to adaptation of robots in absence of food	Adaptation without inhibition at the 210th generation

V - Using emergence

In this part we propose to explain what is emergence. We also explain how this project get inspiration from auto-organization principles in nature, and why these notions are appealing for AI.

This experience is a really simple introduction to genetic algorithms applied to multi-agent task achieving. However, we noticed that with some really basic rules (elitism, crossover, mutation and communication) and simple methods to influence behaviours (punition/congratulation method), we obtain a system able to generate its own strategy and adapt to environement change.
We call emergence the phenomena that makes non-specified behaviours appear (note that *non-specified* doesn't mean *non-expected*). In this experience, we never tell to the robots what set of rule they should use. We only observe that these rules (i.e. strategy) *make spontaneously sense* with the situation : most robots use O_EAT when there is something to eat, and O_MOVE when there is nothing to eat, eventhough it doesn't belongs to *positive action-genes* category.
This experience also states more generaly that *interaction between solutions* (i.e. education) helps the evolutionnary algorithm to go more efficiently to mission objective.

Let's comment some basic generalities in ants' behaviour to understand emergence of organization in real life. If we attempt to describe one single ant's behaviour wandering alone in nature, we have lot of difficulties to summarize it by geometrical objetcs (lines, etc) and we would say that it goes chaotically. However, if we observe the ant in its community, it will exibit some linear behaviour set by the group's organization.
This organization itself emerges from chaos in individual ant's moves and interactions between units. Chaotic behaviours in moving improve the whole community because it helps one ant to discover new food and bring back information to the whole group. Ants' communication protocole helps to widespread information from neighbourg to neighbourg. These interactions make the group auto organize to consume or bring back the food to the anthill.
We consider that organization is emergent because it comes from a *bottom-up* process. Ants don't perform any instructions coming from any superior entity. They're just equiped with receptors (taste, temperature, messages, etc) and actuators (paws, etc). The collective *organised* behaviour comes from periodic interactions between ants : the line of ants auto organize (logisticaly, as well as geometricaly for observer) because ants are in a sufficient number to receive and transmit an information at a stable frequence. While information is widespreading, ants leave periodicaly the group of 'uninformed' ants to go to the new common goal. Auto organisation goes further in food carrying : each ant gets food from the preceding ant going back to the anthill with food. Passing food might comes from two specific behaviours that could have been selected by natural evolution as a good cooperation principle : each ant looking for food directly take the food that it can see, while each ant carrying food let other ants take it.

These observations let us think that if we add inputs and outputs to our virtual robots, strategies will automatically emerge to go further in organization complexity. A really interesting experience as a next step would be to increase the number of inputs, outputs and rules to provide more choice to each robot. Since robots could interact with more actions (fight, share food, etc) and more messages, more different teams' strategy of working robots would emerge thanks to wider configurations of settings in the punishment/congratulation method.
These asumptions lead us to consider a team strategy as a particular stable gene repartition, and to evaluate the diversity (i.e. *complexity*) of the strategy by the number of *building blocks* in the fuzzy logic grammar used for robots (inputs, outputs and rules). This means that the system will create more specified strategies for different situations with more parameters submitted to evolution.

Conclusions

The robot mission here was simply to find food because this kind of team objective is a good starting point for many real case problems of team robots: cleaning, searching, exploring, etc.
Such an interest for emergence is due to a growing difficulty to specify robotic systems that are more and more complex. Instead of programming a system in a hard and long *top-down* approach (that lead to time-consuming and expensive industrial conception), it could be interesting to use emergent properties of an intelligent system to tune it.
This totally justifies our interest for genetic algorithm and chaos in industrial robotic cases. Our bottom-up approach lying on the theory of complexity could be a powerfull way to generate and set intelligence in the way each individual robot uses inputs to adopt appropriate behaviours.
With an industrial interest in mind, we understand that the remaining question is to control robots organisation, because it emerge from individual autonomy and interactions in the group without any need of superior teaching. Tuning the congratulation/punishment method solves such a question.