Evaluation of Cooperative Strategies in Optimization Problems

Alejandro Sancho-Royo, David A. Pelta, José L. Verdegay, Juan R. Gonzalez
Working Group on Models of Decision and Optimization
Dept. of Computer Science & AI – 18071
University of Granada – Spain
Phone: +34-958240468, Fax: +34-958243317
email:dpelta@decsai.ugr.es

ABSTRACT: Cooperative problem solving is a paradigmatic case of social interaction. In the context of optimization, we propose a framework to test different cooperative strategies. The framework is problem independent and allows a simple implementation on a freely available simulation tool. The cooperative strategies are modelized using a well-known concept of control theory, namely, fuzzy rules. Preliminary results are shown with examples of cooperation among unrelated individuals obtained from the literature.

KEYWORDS: cooperative problem solving, optimization, soft computing

1.- INTRODUCTION

Biological entities (ranging from bacteria to humans) can engage in many and varied types of social interaction, ranging from altruistic cooperation through to open conflict. A paradigmatic case of social interaction is "cooperative problem solving" (CPS), where a group of autonomous entities work together to achieve a common goal. For example, we might find a group of people working together to move a heavy object, play a symphony, build a house, or write a joint paper. CPS has been studied by researchers from Distributed A. I., Soft Computing, Economics, Philosophy, Organization Science, and the Social and Natural Sciences [1,3].

In the context of optimization problem, this situation can be seen as follows: the goal is to find the "best" solution for the problem at hand, while the "entities" can be thought as optimization algorithms. The idea of having a portfolio of solving strategies and use all of them in a parallel and coordinated fashion to solve the problem is justified because of the following fact: no algorithm outperforms another one in all circumstances. Besides, experiences in the field of optimization show that problem instances can be grouped in classes and there exists an algorithm for each class that solves the problems of that class most efficiently.

In order to assess the benefits of a proposed cooperation strategy in the context of optimization, we should compare it against other ones over a set of test problems (more specifically a set of instances of each particular problem). This task is far from trivial, time consuming and requires a great amount of programming (design, implementation and test) effort. In this situation it would be of great help to have tools that allow us, easily, to get a feeling about the potential of the strategy proposed before approaching its full implementation, comparison and assessment.

For this reason, in this paper we propose to focus on the following targets: first, to present a problem-independent framework to simulate the main general features of any optimization problem; and second, to use such framework together with a simulation tool, as testbed to evaluate the behavior of cooperative strategies. In particular we will focus on cooperation models among unrelated individuals.

The modelization of the (nature inspired) cooperation strategies may be done through IF-THEN rules. Due to the inherent uncertainty of the natural processes, it seems to be more adequate to use fuzzy rules to obtain a better representation of the situation. Furthermore, the cooperation mechanism can be seen as a decentralized control problem, and fuzzy rules have been successfully applied in control contexts. That's why, we will use fuzzy rules in this proposal.

To achieve the objectives proposed, we organize the paper as follows: in Section 2 we describe the problem-independent framework to model optimization problems. Then in Section 3, we focus on the topic of cooperation among unrelated individuals and the corresponding representation using fuzzy rules. Finally, in Section 4 we show the simulation's results obtained using the simulation tool freely distributed NetLogo. Section 5 finalizes with the conclusions.

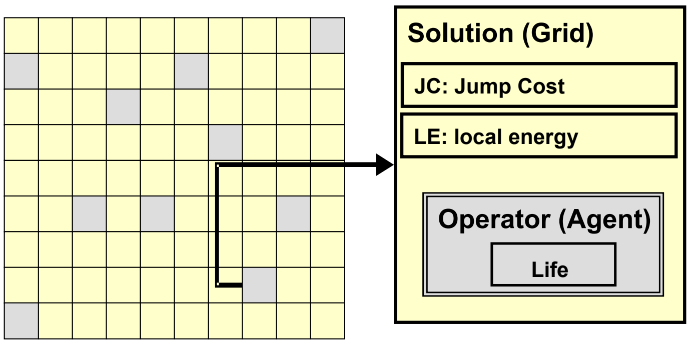


Figure 1: Scheme of the model showing the information stored in each object. Gray cells have an operator inside.

2.- THE FRAMEWORK

Every optimization problem has, at least, the following two elements: the set *S* of feasible solutions and the cost function $f(x)$ where $x \in S$. In general, the search for a solution $x^* \in S$ such that $f(x^*)$ is the maximum / minimum value is hard because of the existence of local optima, so a successful search strategy should provide mechanisms to avoid getting trapped into them.

In our model, we depart from a matrix or lattice of solutions with dimensions *N x M* where each cell *(i,j)* is occupied by a solution $x \in S$. The only information available for a solution is its current cost. It is important to note that the matrix does not capture any metric nor topological feature of the solutions; it acts as a repository of solutions. Besides solutions, our model has a set *P* of agents. Each agent can be associated with a modification or move operator $o \in P$ and, essentially, is a function $S \to S$. We start with *k* agents that are also placed in the matrix and *k << N x M.* A graphical representation of the model is shown in Figure 1.

When an agent "falls" onto the cell *(x,y)*, it modifies the solution stored in such position. Truly, this modification is done over the cost of the solution as follows: each agent has certain amount of *life* available and solutions have (besides the cost) two additional values: *JC* (jump cost), the energy needed to positively change its cost, and *LE* (local energy) the current energy stored. The essential procedure says that if an agent takes a solution where *LE* > *JC*, then it will change positively its cost, and it will gain some *life*. If *LE* < *JC*, then, the agent will change negatively the cost of the solution but it will left part of its *life* in *LE*. The amounts of energy gained/lost from the solutions/agents and the cost variations are governed by fuzzy rules that will be described later.

Once the solution is changed, the agent moves to another position (randomly or based on some information from the neighbors). The agents' *life* is also used to manage its survival and reproduction chances. So we have in this model, two sets or populations that are jointly evolved: 1) solutions, that are optimized in terms of their cost, and 2) agents, that are evolved in a "Darwinian" sense (survival of the fittest).

In order to avoid any specific relation to a concrete problem, we don't use any mention to the objective function. To do that, we have substituted the calculation of the objective function *f* with another one *h(x)* that represents the energy needed to escape from a local optimum of cost *x*. In this way, we can

- avoid thinking on a specific optimization problem
- manipulate the hardness of the model through different definitions of the function *h*

Now, the question is how *h* looks like given that the distribution of the local optima is not known. As a first approximation we assume that it is harder to escape from local optima whose values are close to the global optimum, than to escape from those that are far (in value) from this reference point. In this work, we suppose that the set of local optima are finite from 1 to 100, and we sort it in such a way that the worst is the first and the global optimum is 100. Then, for each possible value $x \in \{1,\ldots,100\}$, we calculate the amount of energy needed to escape from that point as:

$$h(x) = \left(\frac{x}{20}\right)^4 \left| \sin\left(\frac{80}{(\frac{3}{2}x)^{0.99} - 100}\right) \right| \tag{1}$$

where the domain of *h* is restricted to [0,100]. Figure 2 represents the distribution of the energies for 100 integer values of the cost function.

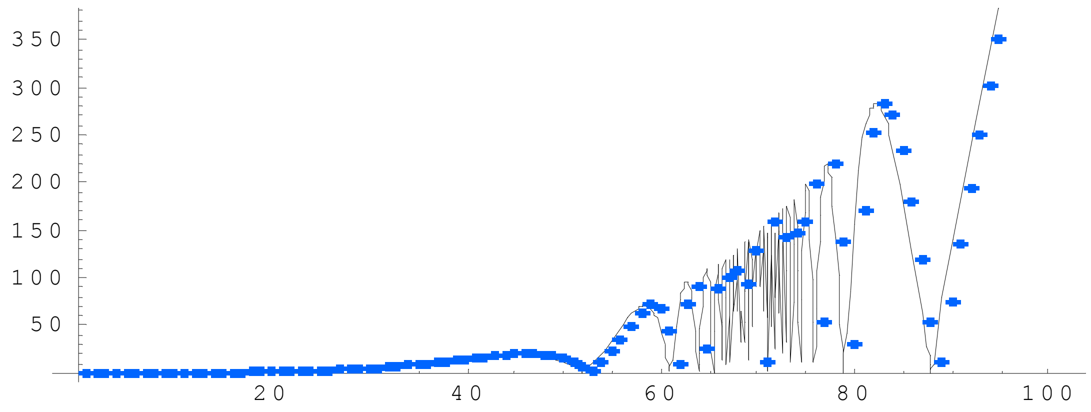


Figure 2: Function h(s) showing the energy (Y-axis) needed to escape from a solution with cost s (X-axis).

As we said before, the amounts of energy gained/lost from the solutions/agents and the cost variations are governed by fuzzy rules. These rules are defined through linguistic labels over the considered variables. In this work, we define the two following rules:

IF *JC – LE* $<^*$ *0* , ***THEN*** *"increase F"* AND *"set LE =' 0"*
IF *JC – LE* $\geq^*$ *' 0*, ***THEN*** *"increase LE"* AND *"decrease F"*

Where *F* (fitness) is the cost of the current solution, *JC* is the energy needed to positively change *F* and *LE* is the current energy stored. The arithmetic operators $<^*$ y $\geq^*$ represent fuzzy comparisons between numbers; the statement *"increase F"* stands for "*add to F a fuzzy quantity ΔF*", and *"set LE =' 0"* means "*replace LE by the fuzzy value 0*". The usual inference mechanism is applied: the membership value of the antecedent is evaluated and such value is then used in the consequents to obtain the output variables.

3.- THE COOPERATION MODELS

For illustration purposes, we will focus on two models of cooperation among unrelated individuals. These models are *reciprocity* and *byproduct mutualism*. Both strategies are described next.

We consider three types or breeds of agents: the first one, named "independents", is a breed showing no cooperation; cooperative agents following a reciprocity model form the second breed; the third type follows a byproduct mutualism model of cooperation. All the cooperation models are based on the idea of allowing the survival of the fittest individuals. For this purpose, we use a parameter *life* that is change in all iterations; when *life* reaches zero, the agent die. Two situations allow an agent to obtain *life*: when it improves the fitness *F* of the solution at hand, or when it improves the corresponding local energy *LE*. The pseudo-code for these two steps is:

```
fitness ← fitness + improving_fitness
local_energy ← local_energy + improving_local_energy
life ← life + improving_fitness + improving_local_energy – lose_of_life
```

It's important to note that *improving_fitness* can be negative, *improving_local_energy* is always non-negative and *lose_of_life* is a value that stands for the amount of *life* that each agent lost in every iteration. In this implementation, *lose_of_life* increases proportionally with the number of agents of each type. The calculus for *improving_fitness* and *improving_local_energy* are made using the fuzzy rule showed before.

Cooperation through Reciprocity

A classical example of reciprocity in nature is Wilkinson's work [4] on blood sharing among vampire bats *(Desmodus rotundus)*. In Dugatkin [5] we can read:

"*Vampire bats typically live in groups composed largely of females, with a low average coefficient of relatedness [...] Somewhat remarkably, females in a nest of vampire bats regurgitate blood meals to other that have failed to obtain food in the recent past.*"

This cooperation model among no related animals has been implemented in the simulation (in the module *Update-life*) as follows: each agent of this breed with a very high *life* looks in a radius for other agents (of the same breed) with a very low life and shares with them an amount of *life*, in a zero sum way. That is, there is not an increment of the global *life* of the agents in the sharing.

Cooperation through byproduct Mutualism
This cooperation scheme is adapted from the following example quoted by Dugatkin [5] about the mutualism in (*Passer domesticus)* that has been studied by Elgar in 1986 [6]:

"Elgar found evidence that those sparrows arriving at a patch of food first were the most likely to produce chirrup calls. Furthermore, chirrup call rates were higher when the food resource was divisible. When food items were small enough that sparrows could pick them and fly away, that is just what the sparrows did, and in the process, they produced no chirrup calls. Chirrup calls were, however, emitted when larger foods items –those that were too big to remove from the experimental area- were found at feeders."

This behavior is implemented in the simulation through the module *move-agents* (described next). When an agent of this breed is going to move, it decides if the solution where it lies is easy to improve or not. If the answer is yes, then the agent informs its current position to other agents of the same breed lying within certain radius.

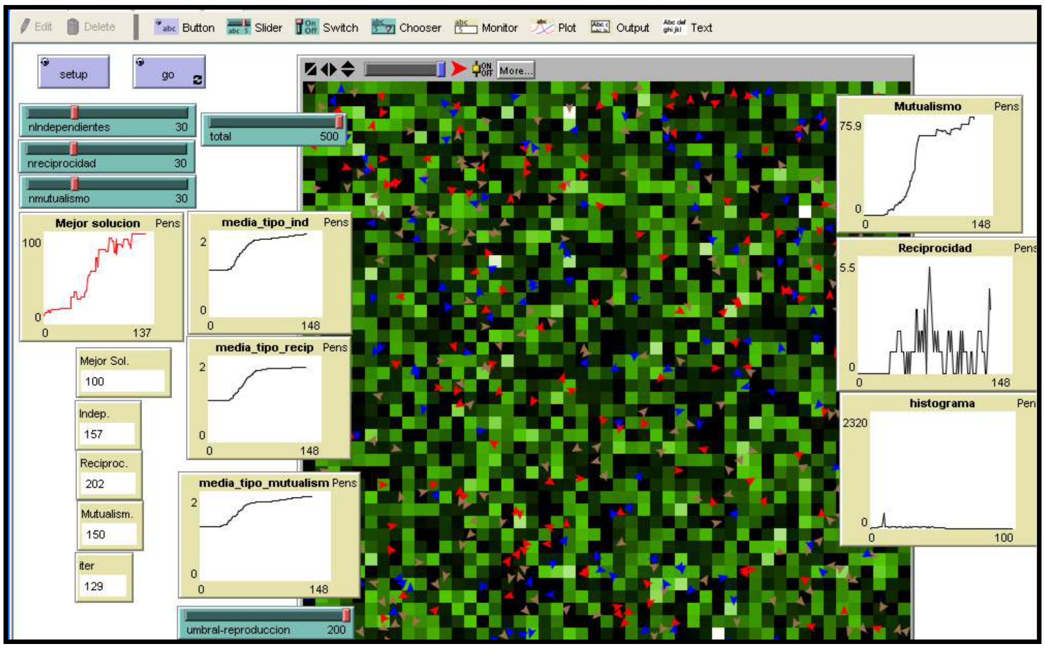
4.- THE SIMULATION

The whole algorithm that governs the simulation can be resumed in the following pseudo-code:

```
Simulation Pseudo-Code
Initialize Variables
Do while (not stop)
[
  For every solution
    [Calculate Jump's Cost
     Evaluate Fuzzy Rules]
  For every Agent
    [Move Agent
     Update life values]
  For global variables
    [Update statistics
     Do Plots]
]
```

First, the simulation initializes the variables, and then, while the stopping criterion not holds, the simulation runs. There are two different objects in the simulation: solutions and agents. For the solutions, the simulation computes JC and the values for the membership functions of the fuzzy rules. For the agents, the simulation computes the moves and the values of *life*. Finally, some statistics are collected to draw the different plots of the simulation.

All these processes are implemented using a freely distributed tool named Net Logo [2]. This tool allows us to check easily, quickly, and interactively, any working hypothesis about the simulation model. Furthermore, Net Logo interface is very helpful to understand the running simulation. An example of our simulation' interface is shown in Figure 3.



nIndependientes 30
nreciprocidad 30
nmutualismo 30
total 500
Mejor Sol. 100
Indep. 157
Reciproc. 202
Mutualism. 150
iter 129
umbral-reproduccion 200

Figure 3: Interface of our simulation with NetLogo.

We present now an experiment made with this implementation of our model. The aim of the experiment is to understand what type of cooperation models are best suited to deal with the optimization framework explained before. Moreover, we designed eleven groups or initial populations with varying number of initial agents of each breed (i.e. using different models of cooperation: independent, reciprocity, and mutualism). The following Table shows the initial values for each group:

	Number of Agents		
Group	**Independents**	**Coop. by Reciprocity**	**Coop. by Mutualism**
1	30	0	0
2	0	30	0
3	0	0	30
4	0	30	30
5	30	0	30
6	30	30	0
7	30	30	30
8	15	15	0
9	15	0	15
10	0	15	15
11	10	10	10

Table I: Groups of Initial Values

For each group, we run the simulation 50 times. Each run is stopped when the optimum was reached or when all the agents died (Although this second situation never happened). As a performance measure, we recorded the number of iterations needed to obtain the optimum. We show in Table II the average, standard deviation and minimum of this variable for each group. A simple view indicates that the values of the three first groups are higher than those of the other ones. In Fig. 4 we show the confidence interval for the average of iterations for every group at level 95%.
A Kruskal-Wallis test indicated that there were statistically significant differences among the means of iterations among the groups (signification level higher than 99.9%). This proof does not allow discriminating between any pair of groups, so we performed several non-parametric tests (U of Mann-Whitney) that are shown in Table III.

	Iterations		
Group	Average	St. Dev.	Minimum
1	358,72	178,81	113
2	338,72	166,94	118
3	340,88	247,37	50
4	122,06	36,28	58
5	133,80	52,89	61
6	126,58	39,81	80
7	109,20	20,20	53
8	122,82	41,71	72
9	117,10	47,65	47
10	135,58	89,54	48
11	110,70	32,13	47

Table II: Average, St. Dev. and Minimums of the number of iterations used to reach the optimum.

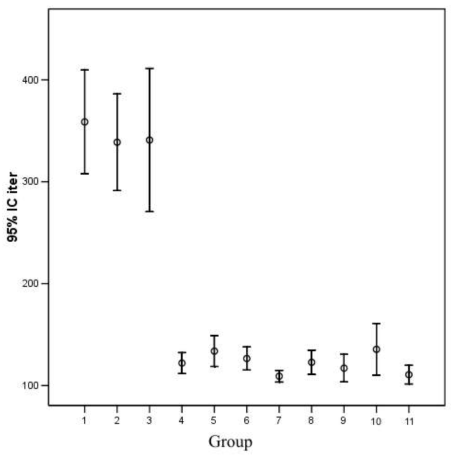


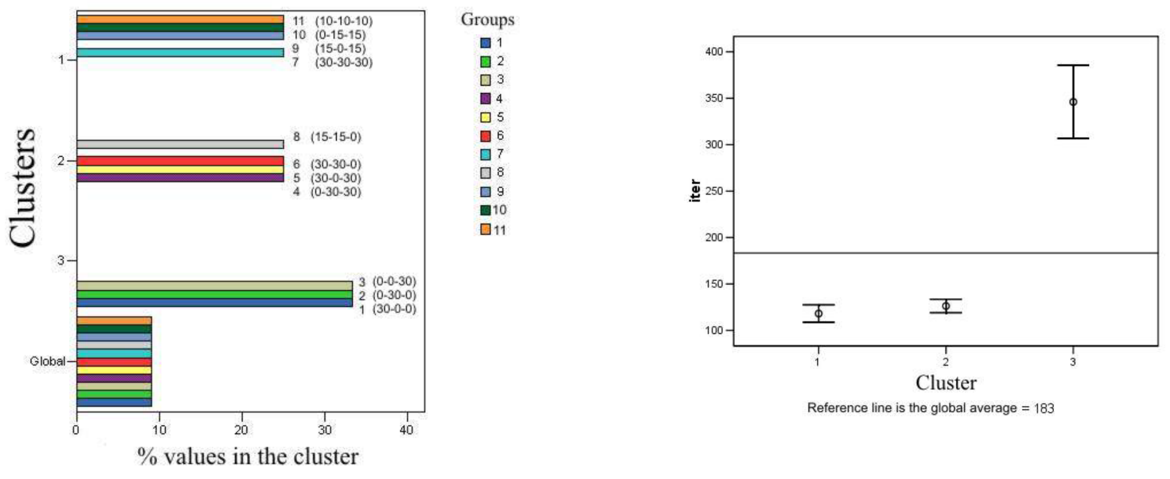
Figure 4: Confidence Intervals for the Number of Iterations

Group	1	2	3	4	5	6	7	8	9	10	11
1		N.S.	N.S.	***	***	***	***	***	***	***	***
2			N.S.	***	***	***	***	***	***	***	***
3				***	***	***	***	***	***	***	***
4					N.S.	N.S.	**	N.S.	N.S.	N.S.	*
5						N.S.	**	N.S.	**	N.S.	**
6							**	N.S.	*	N.S.	**
7								N.S.	N.S.	N.S.	N.S.
8									N.S.	N.S.	N.S.
9										N.S.	N.S.
10											N.S.
11											

*** ($\alpha < 0.001$); ** ($0.001 \leq \alpha \leq 0.125$);* ($0.125 \leq \alpha \leq 0.25$); N.S. ($\alpha > 0.25$)

Table III: Signification levels with the U Mann-Whitney non-parametrical test

It seems clear from Table III that three sets of values leading to three blocks of non-discriminated groups exist. In order to confirm or reject this hypothesis, we have done a k-means cluster analysis over the whole set of output results (50 runs x 11 groups).
After the clustering algorithm was applied, we analyzed which group produced every value and, surprisingly, no overlaps were detected. The result of this analysis appears in Figure 5 (a) where the percentage of values coming from each group in every cluster is shown. For example, the third cluster only has values corresponding to outputs from groups 1,2 and 3. Within parenthesis, the initial composition of the group is indicated. For example (15-15-0) stands for an initial value of 15 independent agents, 15 cooperative by reciprocity agents and 0 agents of the last type. Figure 5 (b) shows the confidence interval for the number of iterations within each cluster.



(a) (b)

Figure 5: a) Composition of each cluster and b) Intervals of confidence for the number of iterations within each cluster

The joint analysis of the results presented leads to a clear conclusion: it's better to use together different types of agents than just a set from a unique type. This fact confirms the claim posed at the beginning of the paper: no algorithm outperforms another one in all circumstances. So, the mix of different search strategies allows improving the number of iterations needed to reach the optimum value.

The cluster analysis was able to detect the existence of classes using just the information available about the number of iterations used. Differences among the mean values between cluster 1 and 2 are small but, as one can see in the Table III, they are statistically significant.

Two interesting subclasses can be detected: one formed by groups 7 and 11; and the second one by groups 4,5 and 6. Groups in the first one are initially composed by the three types of agents available, while two types compose the groups in the second subclass. Finally, we should point out that no significant differences have been found between groups 7 and 11, so we can conclude that the best performance is obtained when the three types of agents are used and with the same proportion.

It is interesting to remark that an indirect cooperation arise among the agents of different breeds through the changes performed on the *LE* values of the solutions. Such values act as an "environment" where, in some sense, the history of the changes done to a particular value may help or avoid to improve the cost of the corresponding solution.
The impact of this indirect cooperation needs to be assessed.

5. CONCLUSIONS

In this paper we have presented a framework to check different models of cooperation among agents in an optimization context. Our framework is based on a grid where solutions and agents co-evolve. Agents move and alter the solutions, while simultaneously, gain or loose life and in turn they reproduce or die showing a "survival of the fittest" scheme.
As a proof of concept, two nature inspired models of cooperation (that occur among non related animals) have been implemented: first, a classical example on reciprocity and second, another one on byproduct mutualism. Besides the agents showing these behaviors, we also considered "selfish" agents that do not collaborate at all.
Using a freely available tool (NetLogo) we could implement these models easily, quickly and simply.
We have checked the performance of the system when different proportions of agents of each type are initially used. The results clearly indicated that a balanced composition of the initial values of the breeds produce the better performance in terms of number of iterations used to reach the optimum value.

ACKNOWLEDGMENTS

Research supported in part by Project TIC-2002-04242-C03-02.

REFERENCES

[1] Wooldridge, Michael; Jennings, Nicholas R., 1999, "The Cooperative Problem Solving Process", Journal of Logic & Computation, 9 (4)

[2] Wilensky, U. 1999, NetLogo, http://ccl.northwestern.edu/netlogo. Center for Connected Learning and Computer-Based Modeling. Northwestern University, Evanston, IL.

[3] Forbes, Nancy, 2004, "Imitation of Life : How Biology Is Inspiring Computing", The MIT Press

[4] Wilkinson, G., 1984, "Reciprocal food sharing in vampire bats", Nature 308 pp. 181-184

[5] Dugatkin, L.A., 2002, "Animal cooperation among unrelated individuals", Naturewissenschaften 89 pp. 533-541

[6] Elgar, M., 1986, "House sparrows establish foraging flocks by giving chirrup calls if the resource is divisible", Anim Behav 34 pp. 169-174