Model and Analysis of Local Decision Based on Cooperative Self-Organization for Problem Solving

Gauthier Picard*, Pierre Glize
IRIT, Université Paul Sabatier
118, route de Narbonne
F-31062 Toulouse Cedex, FRANCE
{picard,glize}@irit.fr

Abstract

This paper presents an approach based on cooperative self-organization for artificial systems, in order to tackle openness and dynamics. In this work, cooperation is used as a local criterion enabling parts of the system –the cooperative agents– to reorganize as to autonomously modify their interactions and then the global function. The difficulty in defining cooperation and the means to reestablish a cooperative state within a non cooperative system are underlined and analyzed. Two cooperative agents' behaviors are expounded, at the boundary between altruism and selfishness. This approach is also illustrated by modeling, from a local viewpoint, a classical constraint satisfaction or optimization problems.

Keywords: Multi-Agent Systems, Self-organization, Cooperation, Emergence, Local Decision.

1 Introduction

Defining automatic decision systems may become more difficult since stakeholders are distributed –logically, geographically or temporally– like in flood forecast, timetable generation or molecule conformation [7]. Distributed and dynamic Constraint Satisfaction Problems (CSP) are classical formalisms to tackle such decisional problems. But these classical approaches often lack efficiency and adaptivity when the environment (the constraints modifier) is open and dynamic. In this paper, MAS are used to model and solve CSP by using *cooperative self-organization* notion which is inspired from social and biological phenomena.

This work aims at providing local decision criteria to distributely –in the sense that knowledge is distributed among agents– solve or optimize constraint-based problems. These criteria are twofolds. Firstly, a criterion must be defined to locally decide which agent must act to solve local problems. Secondly, another local criterion must be defined for an agent to decide which action to perform to reach the global optimum. In the approach presented in this paper, these criteria are influenced and led by the cooperation notion. The first criterion will consider the agents which is in the less cooperative situation. The second one will consider the action which is the most cooperative one by measuring the impact of actions on the neighbors.

This approach is firstly introduced by reformulating the distributed decisional problems into CSP and then into an organization oriented multi-agent paradigm in section 2. To illustrate this approach the N queens problem is modeled in section 4 and results are shown in section 5. Later, in section 6, the potential of this model is discussed by generalizing it to other problems, and in section 7, it is compared to existing approaches, before concluding in section 8.

2 Constraints and Multi-Agent Systems

Classical constraint-based decisional problems can be expressed by using the CSP formalism. A CSP is a triplet $\langle \mathcal{X}, \mathcal{D}, C \rangle$ such as $\mathcal{X} = \{x_1, \ldots, x_n\}$ is the set of variables to instantiate. $\mathcal{D} = \{D_1, \ldots, D_m\}$ is the set of domains. Each variable x_i is related to a domain of value. $C = \{c_1, \ldots, c_k\}$ is the set of constraints, which are relations between some variables from $\mathcal{X}$ that constrain the values the variables can be simultaneously instantiated to. Therefore, making a decision consists in finding a solution, i.e. a complete and consistent affectation of $\mathcal{X}$. In distributed constraint-based decisional problems (DCSP), distribution can affect either variables or constraints. Most approaches consider the first kind of distribution by defining a function ϕ (also defined by a predicate *belongs*) that bounds variables to stakeholders (agents for example): $\phi(c_i) = j$ (or $belongs(c_i, j)$) means that the constraint c_i belongs to stakeholder j [21].

As in a dynamic and complex environment every constraint cannot be completely satisfied, distributed CSP are often tackled as an optimization problem. This formulation is called a distributed constraint optimization problem (DCOP). Finding an optimized solution is equivalent to finding a solution in which the sum of all the non satisfied weighted constraints is minimal. The two major and complete algorithms to solve such optimization problems are Adopt and OptAPO [16, 13]. But all problems cannot be viewed as in an utilitary point of view, and optimization functions can be more complex than a sum. For example, in frequency assignment problems, cost function is a weighted sum considering, hard constraints and multiple levels of soft constraints, which can be viewed as a leximin criterion [6]. Other compositions, for multi-criterion optimization can also be defined. Moreover, complete algorithms often remains not efficient to solve large problems. Thus, local research based and/or uncomplete algorithms can be considered as in ERA [11].

2.1 Multi-Agent-Based Viewpoint

Agentifying CSP implies that constraints are owned by agents and the environment is the variable possibility space, i.e. a n-dimension grid –the cartesian product of all domains of $\mathcal{D}$– composed of cells at the intersections of all the domains. Agents, as constraints owners, have to explore this grid to find a location, i.e. an affectation of variables verifying every agents' constraints. Therefore, the main goal of an agent is to book some cells of the grid by moving from cell to cell. The limited perceptions (some cells, and not all the grid) of an agent means its satisfaction is only *locally* determined. Moreover, the situatedness of an agent implies it can occupy only one cell at a given time; but a cell can contain several agents. We distinguish a cell that is occupied, from the cells that are reserved. Finally, the solution is obtained by running the multi-agent system, i.e. running concurrently the agents in the grid in order to find a correct location. These characteristics allow changing the agents (updating, removing, adding) or the constraints. The environment can also change by adding dimensions or cells at runtime.

2.2 Organizational Viewpoint

Now, after the agentification of CSP, the distributed decision making problem can be reformulated by using an important agent-based notion: the *organization*. An organization is a description of all the inter-agent relations. In the presented approach, these relations are not the relations defined by the constraints that are shared between several agents, but the spatial environmental relations. In fact, since agents only perceive a limited number of cells, their interactions are only defined by its position at a given time.

Then, the distributed decision problem becomes finding an adequate organization, i.e. a positioning of all agents that satisfies every agent's constraints. The main problem of the distributed approach of MAS is that since agents only have limited perceptions and knowledge, and there is no global controller, agents cannot locally detect the minimum level of constraint cost (the sum of all non satisfied constraints, for example) and cannot use it during their search process, contrary to local search methods like simulated annealing [9] or tabu search [8], which consider the global cost level to choose the next state to explore.

Since a MAS is here considered as a dynamic system, it can investigate several organizations before reaching a solution. Considering the autonomy of agents, there is no controller to force the reorganization. Agents move in the grid as a consequence of internal and local decisions. Therefore, designers must provide micro-level capabilities to change local interactions and then to change the global organization of the whole system. The main problem of self-organization is to define the trigger of reorganization. In Kohonen's maps, the organization is represented by weights affected to each neuron and its neighbors. Neurons change their weights in terms of a given DOG function (*Difference of Gaussians*) [10]. In artificial ant nests, ants stochastically react to attraction of pheromonæ, ressources and their nest position [5]. Similarly, self-organization can be defined by only providing local capabilities, as the DOG function or the stochastic rules of ant algorithms. This criterion must be as generic as possible.

3 Cooperative Self-organization

In order to provide similar behaviors to agents having to solve a CSP, this paper proposes to use the social cooperation notion, as in [2]. Here, cooperation is not only the tasks or resources sharing but is mainly a behavioral guideline to design agents. Cooperation can also becomes a local criterion to self-organize once it is viewed in a proscriptive manner: agents have to reorganize when they are no more cooperative.

3.1 Cooperation

From social definitions, cooperation is the happy medium (or the equilibrium) between altruism and selfishness (see figure 1). Altruism characterizes entities that prefer helping others to reach their goals than achieving their own goals, contrary to selfish ones that

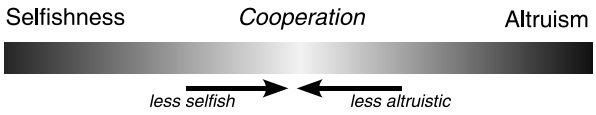

Figure 1: Cooperation: the happy medium between altruism and selfishness.

prefer reaching their own goals. Therefore, cooperative agents must try to satisfy their goals and the other ones' goals as equally as possible. As it may become difficult to precisely define cooperation, it is also possible to find the limit of cooperative behaviors, less altruistic as possible and less selfish as possible. Another definition of cooperation is found in natural systems, in which it is often synonymous of symbiotic relations between two or more entities [15].

Considering agents, cooperation is defined by three metarules –which must be instantiated to particular problems– in terms of each phase of a "*perceive-decide-act*" lifecycle:

Definition 1 *An agent is cooperative if it verifies the following conditions:*

- c_{per}*: perceived signals are understood without ambiguity;*
- c_{dec}*: received information is useful for the agent's reasoning;*
- c_{act}*: reasoning leads to useful actions toward other agents and the environment.*

The first metarule (c_{per}) concerns agents able to interpret the richness of signals coming from the environment and requiring shape recognition, for example. To be as cooperative as possible, an agent must know and learn interaction languages, if necessary. The second metarule (c_{dec}) concerns all the kinds of agents, as soon as they have decisional capabilities. This implies that cooperative agents must be able to produce reasonings from their skills (knowledge about their tasks), their representations (knowledge about their environment) and their aptitudes (rules from which an agent deduce new facts) [1]. The last metarule (c_{act}) is the most known and used in MAS. It requires measuring the impact of an action on the social or physical environment, by defining a cooperation measure, for example, as in section 4.

Non cooperation is simply the exact opposite: $(\neg c_{per} \vee \neg c_{dec} \vee \neg c_{act})$. Such situations are called *non cooperative situations* (or NCS) and are the trigger to reorganize. To sum up, cooperation is a local criterion that enables agents to reorganize when the system is not adequate –i.e. is not in cooperative interaction with the environment. From a global point of view, cooperation can be considered as a meta-heuristic to explore the possibility space by cutting branches leading to non cooperative situations. But the main problem lies in the high level definition of cooperation. Even if Camps et al identify several kinds of NCS, they do not propose a low level generic model [2]. Therefore, that way to instantiate these metarules is illustrated with classical problems in the next section.

3.2 Implementing Cooperation

Cooperative behavior can be specified as exceptions to repair NCS. Such an idea has been more developed by Bernon et al [1]. The main purpose of cooperative agent design is to equip agents with a nominal behavior and goals, and then to add non cooperative situations processing capabilities as exceptions in object-oriented programming –but at a higher level. Another algorithm has also been specified by Capera et al to express the different kinds of NCS an agent may detect [3]. In this paper –since examples are quite simple– the main idea is to consider an agent as an autonomous object following a classical "*perceive-decide-act*" cycle. This cycle can be interrupted when non cooperative situations are locally detected by the agent. Therefore, defining the agents' behavior is equivalent to:

1. specifying a nominal behavior by attributing goals, skills, capabilities, as said in the ADELFE methodology which is devoted to AMAS design [18].
2. specifying *condition-action* pairs describing cooperation exceptions. Actions must be as cooperative as possible –at the happy medium between altruism and selfishness (see section 4.2).

4 Case Study: the N Queens Problem

This paper aims at showing an multi-agent modeling of CSP by using a cooperative self-organization approach. Therefore, to illustrate the model, a classical example is developed: the N queens problem, presented in section 1. Here, the problem is a satisfaction problem with no soft constraints.

4.1 Queen Nominal Behavior

As said in section 3.2, the nominal behavior derives from local goals, skills and capabilities. A queen-agent ($q_i \in \mathcal{A}$) can perceive all attackable cells ($pCells(q_i)$)

and the cell it occupies ($cell(q_i)$). Concerning actions, a queen can move on a perceived cell[1] ($moveTo(c_j)$), reserve (or mark) the cells to inform other queens and identify conflicts ($reserve(c_j)$ or $reserve(\{c_j, \ldots, c_k\})$), and simply rest on the current cell ($rest$). The markers affect the common environment –the grid– and not the MAS which is composed of distributed autonomous queens. These marks are deleted once the agent moves. To decide which next cell to occupy, queens must be able to perceive other queens ($pAgents(q_i)$[2]). Moreover, when reserving a cell, a queen puts extra-data about the less conflictual cell it perceives (see section 4.3), and therefore a queen can know the number of queens which perceive a given cell ($cost(c_j)$ or $cost(q_i)$[3]) and their constrainedness degree.

During its lifetime, a nominal queen follows the behavior presented in the algorithm 1. This algorithm represents a nominal behavior which only leads the queen to a local minimum without taking care of other queens. In order to avoid local minimum resting, this behavior must be enriched by cooperation rules.

4.2 Cooperative Behavior

As previously said, cooperative self-organization rules to avoid NCS are specified as exceptions. The condition guards are instantiations of the metarules presented in section 3.1. The actions to repair NCS must be chosen within the actions agents can perform as cooperatively as possible. This may imply defining a cooperation measure to sort multiple actions. In the N queens problem, considering the given perceptions and actions, two NCS can be identified:

- concurrency ($\neg c_{act}$): two queens are located on the same cell;
- conflict ($\neg c_{act}$): two queens can respectively attack themselves.

Since queens cannot directly communicate by using complex semantics in the present case, there is no c_{per} case. In the same manner, queens can always find at least one action to perform, since they can rest[4]; therefore there is no c_{dec} case. The two identified NCS are different instantiations of the c_{act} rule. Actions to solve these NCS are the following:

[1] Even if another queen occupies the cell.
[2] $q_j \in pAgents(q_i) \equiv (\exists k((c_k \in pCells(q_i)) \wedge (c_k = cell(q_j))))$
[3] $cost(q_i) \equiv (cost(c_j) | (c_j = cell(q_i)))$
[4] which is, finally, their absolute goal

Name: **Concurrency (for queen q_i)**

Condition:
$\exists j((j \neq i) \wedge (cell(q_i) = cell(q_j)))$

Actions: `moveTo(`$mostCooperativeCell(q_i)$`)`

This previous NCS specification is quite simple: if two queens are located on the same cell, the first to detect this situation moves to the less conflictual (see section 4.3), i.e. $mostCooperativeCell$, cell it perceives.

Name: **Conflict (for queen q_i)**

Condition: $\exists j((j \neq i) \wedge (q_j \in pAgents(q_i)))$

Actions:

```
//less-altruistic-as-possible
let qj = lessConstrainedAgent(pAgents(qi));
if ((qi = qj) ∨ (cost(ci) > cost(qj))
   then rest
   else moveTo(mostCooperativeCell(qi))

//less-selfish-as-possible
let qj = lessConstrainedAgent(pAgents(qi));
if ((qi = qj) ∨ (cost(ci) < cost(qj))
   then rest
   else moveTo(mostCooperativeCell(qi))
```

To solve this NCS, two different actions can be identified. The first one is called *less-altruistic-as-possible* behavior and depends on the other agents. Here, agents only act if they are less constrained than their neighbors (current perceived agents). In other words, if an agent is more constrained than another one, it will wait until the other moves. As the other agent respects the same rules, it will detect this situation and then will move.

The second possible action is performed if an agent detects is more constrained than its neighborhood. Here, agents are cooperative because they are *less-selfish-as-possible*.

In the two cases, agents must all respect the same cooperation rules, and if moving is necessary, they will move to the less conflictual cell ($mostCooperativeCell$).

4.3 Cooperation Measure

In order to evaluate the most cooperative cell to explore, agents must be able to locally measure the cooperativeness degree of move actions. This measure must take into account the constrainedness degree of the other agents. The idea is to limit the impact of a movement by analyzing the worst constrained agents that see a given cell, so as to choose the cell that does not increase the cost for a queen.

Algorithm 1 – Nominal behavior for a *nominal* agent (q_i)

```
while alive do
  updatePerceptions();                          //removing previous marks and adding new ones
  if (cost(q_i) == 0) then                      //not occupying a conflictual cell
    rest()
  else
    moveTo(minConflictCell(q_i));
    reserve(pCells(q_i))
  endif
done
```

Definition 2 *Let $\mathcal{P}^{q_i}$ the set of cells perceived by agent q_i:*

$$\mathcal{P}^{q_i} \equiv pCells(q_i) \cup \{cell(q_i)\}$$

Definition 3 *Let the cooperation measure* $coop : \{c_1, \ldots, c_n\} \rightarrow \mathcal{N}$*:*

$$coop(c_i) \equiv \max_{q \in \{q' | c_i \in \mathcal{P}^{q'}\}} cost(q)$$

Definition 4 *Let $C^{q_i}_{min}$ the set of cells with a minimum cost from the point of view of an agent q_i:*

$$C^{q_i}_{min} \equiv \{c \in \mathcal{P}^{q_i} | \nexists c' \neq c, cost(c') < cost(c)\}$$

Definition 5 *Let $C^{q_i}_{coop}$ the set of most cooperative cells from the point of view of an agent q_i:*

$$C^{q_i}_{coop} \equiv \{c \in C^{q_i}_{min} | \nexists c' \in C^{q_i}_{min}, c' \neq c, coop(c') \leq coop(c)\}$$

Therefore, the set of the most cooperative cells to choose is the set of cells with a minimum cost and a minimum impact for the other agents. A first remark concerns the contents of $C^{q_i}_{coop}$, which cannot be empty: it contains at least the current cell (see definition 2). A second remark can be done on the data an agent needs for evaluating a cell: the number of markers on the cell, and the cost of each agent having marked the cell (see definitions 3 and 4). Therefore, a marker for a cell only contains the current cost of its owner.

Cooperative agents are then able to determine the set of cooperative cells to move to. But, how can the next cell be chosen without leading to a local minimum? Here is the main decisional challenge for a cooperative agent. By now, Capera et al do not give any guidance [3]. Therefore, any kind of method can be considered: random, tabu, etc. In the next experimentations, good results are obtained with a very simple selection criterion (see section 5.1).

5 Experimentations

In this section, several results are obtained by simulating two different cooperative agents' behaviors: *less-altruistic-as-possible* behavior and *less-selfish-as-possible* behavior, which have been defined in section 4.2

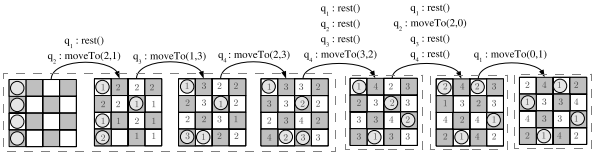

Figure 2: Solving trace for 4 queens, with a *less-altruistic-as-possible* cooperative behavior, in 4 steps and 6 moves.

5.1 Experimental Setup

The two main choices before encoding agents' behaviors and launching the solving process are:

- the initial positioning: all the agents are initially positioned at the left border of the grid (see fig.2);
- the selection function ($mostCooperativeCell(q_i)$) for choosing the most cooperative cell: since implementation implies to choose an order in perceived cells (from closest to farest, from east to north east, clockwise), this order is used to choose the next cell. Moreover, a limited memory of one cell is added to avoid the previous visited cell (very simple tabu implementation).

5.2 Simple Trace with 4 Queens

At the beginning of the solving process, all the agents are positioned on the left side and the environment is *not marked*. A number in a cell represents the number of markers, i.e. the number of agents seeing the cell.

5.2.1 *Less-altruistic-as-possible* behavior.

The figure 2 shows a trace for the 4-queens problem solving. The system finds a collective solution in 4 steps (during which every agent acts), which are delimited by dotted rectangles. Only 6 moves are performed to reach a solution. Some agents do not act during some steps, since they are in cooperative situations, and are not situated on non conflictual cells. Contrary to classical CSP solving methods with global knowledge, the agents move to different lines, as shown in the figure

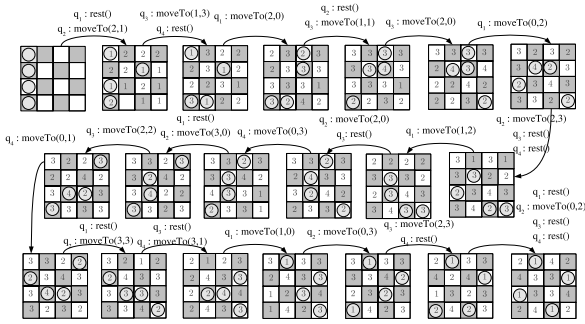

Figure 3: Solving trace for 4 queens, with a *less-selfish-as-possible* cooperative behavior, in 8 steps and 19 moves.

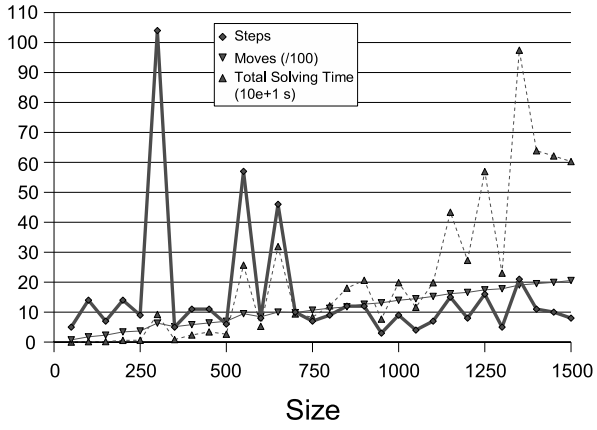

Figure 4: Solving results for different sizes (from 50 to 1500) with *less-altruistic-as-possible* cooperative agents.

when q_3 moves to the position $(1, 3)$. Even if it is not a solution, it represents an intermediary stage to a more adequate state.

5.2.2 *Less-selfish-as-possible* behavior.

Figure 3 shows the solving trace for a *less-selfish-as-possible* cooperative behavior. The system reach a collective solution in 8 steps (twice than *less-selfish-as-possible* cooperative agents) and 19 moves. This is due to the fact that agents prefer moving when they detect NCS rather than resting until another one moves.

5.3 Results for Different Problem Sizes

5.3.1 *Less-altruistic-as-possible* behavior.

Figure 4 shows solving results for different sizes N (from 50 to 1500) with *less-altruistic-as-possible* cooperative agents:

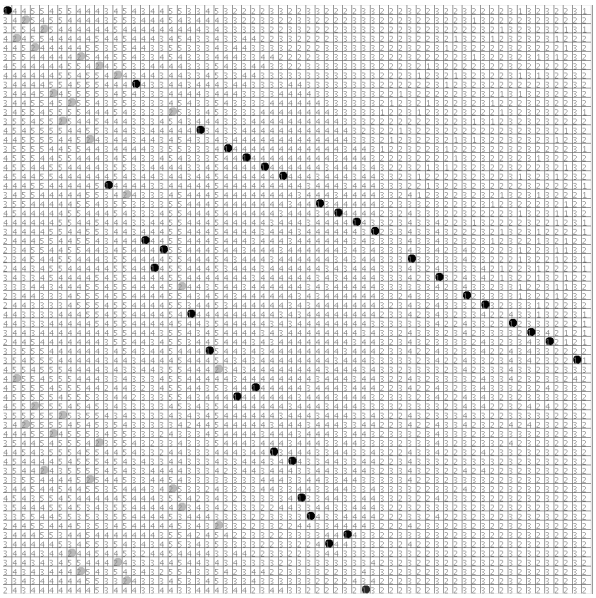

Figure 5: Image capture at step 1 of the solving process for the 64 queens problem. Circles represent queens (black for agents in cooperative state, grey for agent in non cooperative state).

- the total number of moves (for all the agents) during the solving linearly increases in terms of the number of agents ($O(4n/3)$),
- the total solving process execution time (in seconds) significantly increases in terms of the number of agents. This result shows the complexity of the global algorithm, $O(n^2 s)$: n agents perceiving and analyzing at worst $4n$ cells during s steps,
- the number of steps (s), during which every agent acts, seems not to depend on the size of the problem. Informally, it might depend on the cooperation definition. Since this value is the link between micro-level (the agents) and macro-level (the system), it cannot be, by now, formally defined. But experimentally, it is overestimated at $O(n)$.

Therefore, the global complexity of the solving process can be experimentally overestimated at $O(n^3)$, which is a common complexity to solve this problem with classical informed heuristics like hill-climbing [20], whereas the proposed algorithm is not informed. Moreover, at the beginning of the solving process, agents react as in a classical forward checking method. This is shown by the positioning of the agents in figure 5 with the two diagonal lines which are obtained by using classical forward checking.

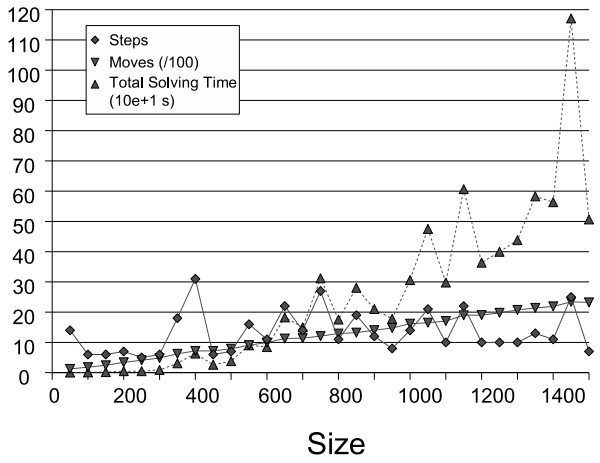

Figure 6: Solving results for different sizes (from 50 to 1500) with *less-selfish-as-possible* cooperative agents.

5.3.2 *Less-selfish-as-possible* behavior.

Figure 6 shows results for different sizes n of problems, with *less-selfish-as-possible* cooperative agents. These results are equivalent –steps, solving time and moves– to those previously expounded, for *less-altruistic-as-possible* cooperative agents –contrary to the results obtained for a 4 queens problem, for which moves and steps were higher.

It is then interesting to remark that those two proposed behaviors tend to a similar cooperative behavior, when the size increases, at the boundary of altruism and selfishness.

5.4 Adaptation and Robustness

Section 1 mainly focuses on the need to provide solutions to dynamic problems which require adaptivity and robustness. To show how relevant the presented approach is concerning dynamics, a set of experiments has been performed with the same experimental setup than before, but with 512 agents and random disturbances: each 10 steps, 10% of randomly chosen agents are moved to random positions in the grid.

The figure 7 shows the global cost ($\sum_{q \in \mathcal{A}} cost(q)$) in terms of steps. The collective, after every disturbance, quickly repairs the organization and find a solution: the system is adaptive and robust to environmental disturbances. Moreover, these results show that a solution is found with a random initial agent positioning. Finally, the first steps of the solving process (see fig.2) are also a particular case of perturbation since the constraint degrees, associated to cells, are erroneous. Experimentations, with *less-selfish-as-possible* cooperative agents and the same setup, shows similar results.

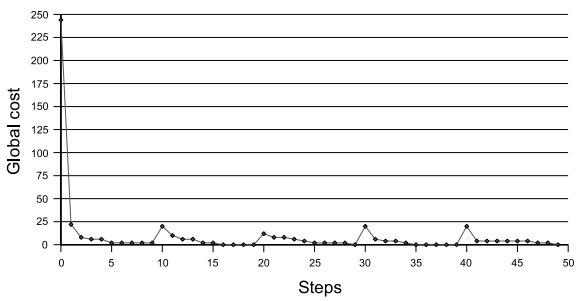

Figure 7: Global cost in terms of solving steps for 512 *less-altruistic-as-possible* cooperative agents with random disturbances.

By using AMAS terminology, the result of the solving process is a spatial organization of the system. Since this organization is found without any global knowledge (positions, constrainedness degrees, etc.), it is a self-organizing process, within which cooperation guarantees the evolution toward a collective solution.

6 Generalizing to Other CSP

Previous sections presented the cooperative self-organization based approach for a precise example, the N queens problem. This section discusses about generalizing this approach to other CSP. As the presented algorithm are quite simple, and do not provide any guidance on selection criteria, it remains generic. Nevertheless, the cooperation criterion, and the perceived cells it requires, may become more difficult to define and/or to implement to tackle more complex problems, with more than 2 domains, for example.

6.1 The $N^2/2$ Knights Problem

In a first stage, it is possible to solve problems very similar to the N queens problem; the $N^2/2$ knights problem, for example. This toy-problem consists in positioning knights rather than queens. Contrary to the N queens problem, a solution is known for all size of problem: positioning all the knights on all the white cells (or all the black ones). Nevertheless, interesting results are obtained, with only minimum changes in algorithms. The only modification concerns the perceived and attackable cells.

Table 1 shows results for different size of $N^2/2$ knights problems with *less-altruistic-as-possible* cooperative agents. The system reaches a collective solution, whereas the algorithm has not been modified. Moreover, the complexity increases since the number of agents is proportional to N^2. These results are posi-

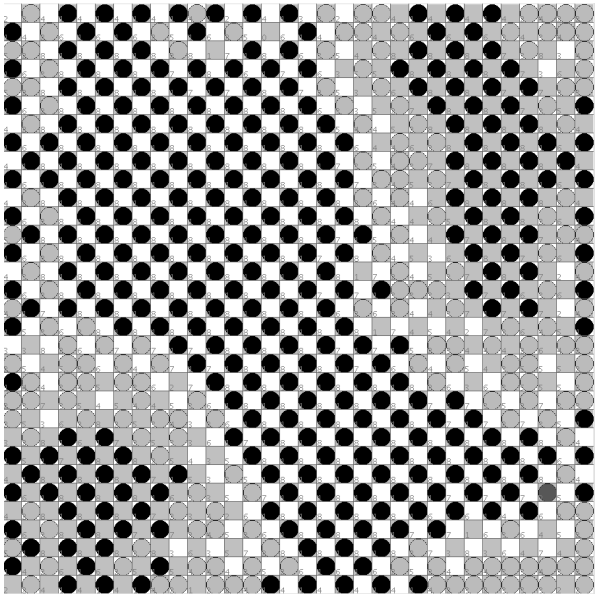

Figure 8: Image capture at step 4 of a solving process for the $32^2/2$ knights problem.

Size	Knights	Steps	Moves	Total time (ms)
4	8	1	6	32
8	32	7	74	62
16	128	11	451	235
32	512	129	16535	9483
64	2048	402	101168	172901
128	8192	277	124341	772483

Table 1: Solving results for different size of $N^2/2$ knights problems (N from 4 to 128) with a *less-altruistic-as-possible* cooperative behavior.

tive in the sense that generalizing to other CSP with two domains is quite immediate. Figure 8 shows the state of the system at step 4 of the solving process for the 32 knights problem. It underlines the range of local interactions between agents that forms three kinds of areas:

- areas with all knights on black cells with no conflicts,
- areas with all knights on white cells with no conflicts,
- and areas with conflicts to solve.

6.2 University Timetabling Problem

We also have been interested in a problem with more than two domains: university timetabling. Here, agents (teachers and student groups) must cooperate to find partners, rooms, timeslots and days. For this problem, a prototype has been developed to illustrate the ADELFE methodology [18]. This application requires to equip agents with limited memory of the other agents and the previous occupied cells [17]. Here agents represent courses of teachers or student groups for specific teachings. Agents must find *partners* and reservation (timeslot and room) that fit personal constraints. Table 2 shows results for different size of problems. The benchmark[5] it is based on has been proposed by the group ASA (Approach by Societies of Agents) of the AFIA (French Association for Artificial Intelligence). Since the choice of the next cell to explore is here partially random, the results are averaged for 10 solvings.

Agents	Steps	Time (s)
2	54.25	82.36
4	58.8	88.9
8	80.2	123.38
16	69.65	113.1
32	71.25	124.15
64	74.65	145.02
128	48.95	159.75

Table 2: Solving results for different size of time tabling problems.

Table 2 shows the evolution of solving time as a consequence of the growing number of BAs in the system. For these experiments, the same exploration space size is kept by increasing the number of cells in the grid proportionally to the number of agents. Only availability constraints are owned by teachers: one time slot per day is forbidden. Once the maximum reached (average 8 BAs), the number of cycles (during which every agent acts one time) decreases as the number of BAs increases. The time that varies the less is the real time. Therefore, it is the most relevant indicator of the solving time evolution. Beyond 32 BAs, it has a logarithmic evolution. More BAs the system has more efficient the solving is – if a solution exists.

We also shown positive results on adaptation and robustness to environmental disturbance such as constraint modifications or agent removals as for the N queens problem with random disturbances. The system is able to quickly repair non satisfying states and to reach by local decision global optimum when no solution exists [17].

[5] http://www-poleia.lip6.fr/~guessoum/asa/BenchEmploi.pdf

7 Discussion

7.1 Distribution

A first discussion point concerns the physical distribution. The presented application does not really show the potential of the approach. In fact, for all the presented solvings for *N* queens problem, the agents always act in the same order. But, as previously shown, the solving process also works with random disturbances. Some other solving –time tabling problems– have been executed with a random scheduler and showed similar results. This is the first step to real distribution, even if the cell reservation remains in a critical section.

7.2 Other Approaches

Concerning distributed CSP/COP solvers, ABT and AWCS where the first relevant algorithms [21]. They are complete and a total order on the set of agents is an important element that avoid loops, but it adds some disequilibrium within the agents' roles and the capability for the system to be openned. For an optimization problem, the nogoods –which consist in memorizing bad past solutions– must be kept in memory, which can be costly. Another algorithm is APO, which lies on mediation to solve local constraint problems, but it requires more messages during the solving process [14]. Finally, the algorithm Adopt, for DCOP solvings, uses on thresholds to reduce the number of memorized explored partial solutions, but still requires an order within agents.

Concerning dynamic distributed CSP/COP solvers, DynDBA is one of the more relevant and efficient [12]. This algorithm is not complete and functions in a synchronous way. But, here, the approach is more local in the sense that agents only reason on neighbors-by-constraints. Moreover, the interesting notion of quasi-minimum allows reorganization by using weighted constraints.

The other family of CSP solving methods, which are influenced by optimization and metaheuristics, considers the problem in a local[6] viewpoint. For example, the tabu search considers states (a given affectation of values to variables) and their neighborhood, i.e. the states that are directly accessible by applying an action (a variable affectation). The idea is to explore the state which optimizes the system function. But this search is restricted by avoiding past states (the tabus) [8]. Another example concerns the simulated annealing which consists in favoring decreasing, in a random exploration of the space, but without completely avoiding increasing by using stochastic structures [9]. Another interesting multi-agent oriented approach has been used in ERA, which assign stochastic behavior to agents to avoid local minima [11]. But agent are still led by conflict evaluation to solve CSPs. Nevertheless, these methods remain global, in the sense that they consider the system as a whole and calculate the global system energy (or function) to explore the neighborhood.

Finally, one promising method to tackle the CSP in a really local viewpoint is the ant approach which consider the solving process only by providing behaviors to the parts of the system –the ants [5]. By using mechanisms such as stigmergy and by simulating the collective, the global system is able to reach an optimized state. Our approach is quite close to ant algorithms or to the approach of Rogers et al [19], but does not need stochastic behaviors: the reaction to markers is deterministic. As for Rogers et al, using only agents' local properties (like power level for physical sensors), the behaviors provided enable adding or removing agents with minimal global disturbances.

7.3 Evaluation Criteria of an Agent Difficulty

The cooperative solving process needs the knowledge of constrainedness degree of an agent and its neighbors. This knowledge is easy to obtain for the queens or knights problems, because it depends directly on the number of neighbors able to attack a given agent. Unfortunately, this is not so easy to know for the timetabling because it is related with the number of constraints an agent must satisfy (size of a room, timeslot, type of course, etc.). Thus, there is no mean to evaluate with a simple formula this difficulty which depends greatly on environmental characteristics not a priori knowledgeable. The only way to evaluate the constrainedness degree in that cases is to learn in real time (during the process solving itself) how many time an agent spent to satisfy its own set of constraints. This is the way used by the timetabling agents. Consequently the priority between agents could be dynamically modified during time and some values previously assigned to variables could be further reconsidered.

7.4 Local optima

The cooperative process solving seems sometimes similar to a gradient descent approach. In this case the gradient descent is the measure of the non cooperative situations an agent encounter. Then it must act in order to reduce this number. For example, in the *N* queens problems, the global sum of non cooperative situations generally decreases regularly during time. Nevertheless

[6] But this *local* notion is related to the search space and *not* to the parts composing the system.

it is not sufficient to obtain a global solution, because some agents remains in a local optimum. But cooperative approach allows to increase locally the number of non cooperative situations for two reasons:

- firstly an agent cannot be inactive when there exist non cooperative situations in its neighborhood: a queen must move even it was in a local minima,
- secondly an agent cannot act to obtain a local similar situation from a cooperative analysis viewpoint. A queen is in a similar situation when it replaces a local minima with a queen by the same local minima with another queen.

Thus the local minima must increase allowing to search a collective solution in another problem subspace without any local knowledge on this. This is similar to a random behavior in other algorithms, with the difference that the perturbation obtained in the cooperative process solving is guided by the cooperative algorithm itself.

7.5 Emergence

Queens and knights problems are satisfaction ones, because the global goal is to find a collective solution satisfying all the constraints of each variables. We can easily understand that when all agents have their own constraints satisfied the global goal is achieved. Nevertheless, there is no way to assert that a local action reduces the distance between the current global state and the global goal. This is the main reason to search theories allowing this converging process where the local behavior is not directly dependent of the global state. Theories of emergence do not need objective function (like in evolutionnary algorithm) which are dependent on the global problem. According to the computational definition of emergence, cooperative self-organization can be considered as a good candidate for an emergence theory [4].

Timetabling is an optimization problem where a solution can be found only if some constaints are relaxed for an overconstrained problem. As underlined in this article, cooperative self-organization allows also to find good solution where agents relax locally some of their constraints in order to obtain satisfiable solution to their neighbors. Taking into account its local evaluation criteria a cooperative agent gives an evaluation of its current solution quality, which could be compared to the quality measure of its neighbors. Thus, without global evaluation function, the system find solutions which are an equilibrium between the constraints of each agents. This is also an emergent result of the cooperative self-organization.

8 Conclusion

This paper has presented a cooperative self-organization based approach to tackle distributed and dynamic decision problems, and more precisely, CSP. These problems have been reformulated by using adaptive multi-agent terminology. Here cooperation is viewed as the boundary between altruism and selfishness. When cooperation is too difficult to be precisely defined, the presented approach proposes to analyze the behaviors which tends to cooperation; from the altruist and the selfish viewpoints.

To illustrate this work, different example of CSP have been implemented and commented. These implementations show promising results on adaptivity and robustness when the system is subject to environmental disturbances. They also raise some discussion points, such as the generalization of the proposed algorithm or the need of memory, when the agents require more complex decision criteria to find an adequate organization. These problematics cross the domains of local search and metaheuristics to implement optimized search processes. Cooperation seems a relevant meta-level criterion, but requires more studies about memory charge, complexity and halting. This last point –even if experimentally obtained– may become a good panel to cooperative self-organizing systems and represents the main perspective.

References

[1] C. Bernon, V. Camps, M.-P. Gleizes, and G. Picard, Designing Agents' Behaviors and Interactions within the Framework of ADELFE Methodology, in: Fourth International Workshop on Engineering Societies in the Agents World (ESAW'03), A. Omicini, P. Petta, and J. Pitt, eds., volume 3071 of Lecture Notes in Computer Science (LNCS), Springer-Verlag, 2003, pp. 311–327.

[2] V. Camps, M.-P. Gleizes, and P. Glize, A Theory of Emergent Computation Based on Cooperative Self-Organization for Adaptive Artificial Systems, in: Fourth European Congress of Systems Science, Valencia, 1999.

[3] D. Capera, JP. Georgé, M-P. Gleizes, and P. Glize, The AMAS Theory for Complex Problem Solving Based on Self-Organizing cooperative agents, in: First International TAPOCS Workshop at IEEE Twelfth WETICE, IEEE Computer Society, 2003, pp. 383–388.

[4] G. Di Marzo Serugendo, M.-P. Gleizes, and A. Karageorgos, Self-Organization and Emergence in Multi-Agent Systems, The Knowledge Engineering Review, 20(2):165–189, 2005.

[5] M. Dorigo, V. Maniezzo, and A. Colorni, Ant System: Optimization by a Colony of Cooperating Agents, IEEE

Transactions on Systems, Man and Cybernetics- Part B: Cybernetic, 26(1):29–41, 1996.

[6] P. Galinier, M. Gendreau, P. Soriano, and S. Bisaillon, Solving the Frequency Assignment Problem with Polarization by Local Search and Tabu, 4OR: A Quarterly Journal of Operations Research, 3(1):59–78, 2005.

[7] J.-P. Georgé, M.-P. Gleizes, P. Glize, and C. Régis, Real-time Simulation for Flood Forecast: an Adaptive Multi-Agent System STAFF, in: Proceedings of the AISB'03 symposium on Adaptive Agents and Multi-Agent Systems(AAMAS'03), D. Kazakov, D. Kudenko, and E. Alonso, eds., AISB, 2003, pp. 109-114.

[8] F. Glover and M. Laguna, Tabu Search, Kluwer, 1997.

[9] S. Kirkpatrick, C. Gellat, and M. Vecchi, Optimization by Simulated Annealing, Science, 220(4598):671–680, 1983.

[10] T. Kohonen, Self-Organising Maps, Springer-Verlag, 2001.

[11] J. Liu, H. Jing, and Y. Y. Tang, Multi-Agent Oriented Constraint Satisfaction, Artificial Intelligence, 136(1):101–144, 2002.

[12] R. Mailler, Comparing two approaches to dynamic, distributed constraint satisfaction, in: Proceedings of the Fourth International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS'05), ACM Press, 2005, pp. 1049–1056.

[13] Roger Mailler and Victor Lesser, Solving Distributed Constraint Optimization Problems Using Cooperative Mediation, in: Proceedings of Third International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS'04), IEEE Computer Society, 2004, pp. 438–445.

[14] Roger Mailler and Victor Lesser, Using Cooperative Mediation to Solve Distributed Constraint Satisfaction Problems, in: Proceedings of Third International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS'04), IEEE Computer Society, 2004, pp. 446–453.

[15] H.R Maturana and F.J. Varela, The Tree of Knowledge, Addison-Wesley, 1994.

[16] P.J. Modi, W. Shen, M. Tambe, and M. Yokoo, ADOPT: Asynchronous Distributed Constraint Optimization with Quality Guarantees, Artificial Intelligence, 161:149–180, 2005.

[17] G. Picard, C. Bernon, and M.-P. Gleizes, Emergent Timetabling Organization, in: Multi-Agent Systems and Applications IV - Fourth International Central and Eastern European Conference on Multi-Agent Systems (CEEMAS'05), volume 3690 of Lecture Notes in Artificial Intelligence (LNAI), Springer-Verlag, 2005, pp. 440–449.

[18] G. Picard and M.-P. Gleizes, The ADELFE Methodology – Designing Adaptive Cooperative Multi-Agent Systems, in: Methodologies and Software Engineering for Agent Systems, F. Bergenti, M-P. Gleizes, and F. Zambonelli, eds., Kluwer Publishing, 2004, pp. 157–176.

[19] A. Rogers, E. David, and R. Jennings, Self-Organized Routing for Wireless Micro-Sensor Networks, IEEE Transactions on Systems, Man and Cybernetics, Part A, 35(3):349–359, 2005.

[20] S. Russel and P. Norvig, Artificial Intelligence: a Modern Approach, Prentice-Hall, 1995.

[21] M. Yokoo, E.H. Durfee, Y. Ishida, and K. Kubawara, The Distributed Constraint Satisfaction Problem : Formalization and Algorithms, IEEE Transactions on Knowledge and Data Engineering, 10:673–685, 1998.