Published in Prasad, B., editor, *Second International Indian Conference on Artificial Intelligence (IICAI'05)*, 20-22 December 2005, Pune, India, pages 3009-3024, 2005.

Model and Experiments of Local Decision Based on Cooperative Self-Organization

Gauthier Picard and Pierre Glize

IRIT, Université Paul Sabatier
118, route de Narbonne
F-31062 Toulouse Cedex, FRANCE
http://www.irit.fr/SMAC
{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. We propose to use cooperation as a local criterion enabling parts of the system –the cooperative agents– to reorganize as to modify their interactions and then the global function. We underline the difficulty of defining cooperation and the means to reestablish a cooperative state within a non cooperative system. 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 problem: the N queens problem.

Keywords. Multi-Agent Systems, Self-organization, Cooperation, 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 [1]. 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.

To illustrate this model, a classical constraint based example is developed: the N queens problem. This problem consists in positioning N queens, from the chess game, on a $N \times N$ chessboard, so that no queen can attack another one. Classically, this problem is modeled with CSP formalism such as the variables to set are the coordinates of the queens; the domains are two N-sized natural sets; and the constraints imply the queens cannot be in the same line, column or diagonal. Here, agent identification is quite easy: the queens.

This crossroad approach is firstly introduced by reformulating the distributed decisional problems into CSP and then into an organization oriented multi-agent

Published in Prasad, B., editor, *Second International Indian Conference on Artificial Intelligence (IICAI'05)*, 20-22 December 2005, Pune, India, pages 3009-3024, 2005.

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, we discuss the potential of this model by generalizing it to other problems, and in section 7, we compare it 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 $\langle \mathcal{X}, \mathcal{D}, \mathcal{C} \rangle$ 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 [2].

Finally, as in a dynamic and complex environment every constraint cannot be completely satisfied, distributed CSP are often tackled as an optimization problem. Finding an optimized solution is equivalent to finding a solution in which the sum of all the non satisfied weighted constraints is minimal. This formulation is called a distributed constraint optimization problem (DCOP) [3].

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 constraints. The environment can also change by adding dimensions or cells at runtime.

Published in Prasad, B., editor, *Second International Indian Conference on Artificial Intelligence (IICAI'05)*, 20-22 December 2005, Pune, India, pages 3009-3024, 2005.

2.2 Organizational Viewpoint

Now, after the agentification of CSP, we can reformulate the distributed decision making problem by using an important agent-based notion: the *organization*. An organization is a description of all the inter-agent relations. In our 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) and cannot use it during their search process, contrary to local search methods like simulated annealing [4] or tabu search [5].

Since we consider a MAS 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*) [6]. In artificial ant nests, ants stochastically react to attraction of pheromonæ, ressources and their nest position. Similarly, we can define self-organization only by 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, we propose to use the social cooperation notion, as in [7]. 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. But how can cooperation and non cooperation be defined?

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 then achieving their own goals, contrary to selfish ones that prefer reaching their own goals. Therefore, cooperative agents must try to satisfy their goals and the other ones' goals as equally

Published in Prasad, B., editor, *Second International Indian Conference on Artificial Intelligence (IICAI'05)*, 20-22 December 2005, Pune, India, pages 3009-3024, 2005.

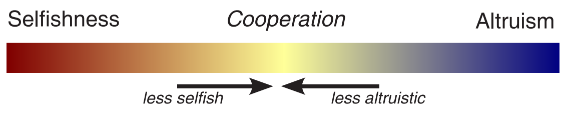

Fig. 1. Cooperation: the happy medium between altruism and selfishness.

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 [8].

Considering agents, cooperation is defined by three meta-rules, 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 meta-rule (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 meta-rule (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) [9]. The last meta-rule (c_{act}) is the most used. 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 [7]. Therefore, we illustrate how to instantiate these meta-rules with classical problems in the next section.

Published in Prasad, B., editor, *Second International Indian Conference on Artificial Intelligence (IICAI'05)*, 20-22 December 2005, Pune, India, pages 3009-3024, 2005.

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 [9]. 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 [10]. In this paper –since examples are quite simple– the main idea is to consider agents as autonomous objects 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 [11].
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 Example: 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.

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.

[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] $cost(q_i) \equiv (cost(c_j)|(c_j = cell(q_i)))$

Published in Prasad, B., editor, *Second International Indian Conference on Artificial Intelligence (IICAI'05)*, 20-22 December 2005, Pune, India, pages 3009-3024, 2005.

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
```

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 meta-rules 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, we can identify two NCS:

– 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, we consider queens always can 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:

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.

[4] which is, finally, their absolute goal

Published in Prasad, B., editor, *Second International Indian Conference on Artificial Intelligence (IICAI'05)*, 20-22 December 2005, Pune, India, pages 3009-3024, 2005.

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 $q_j$ = lessConstrainedAgent($pAgents(q_i)$);
if (($q_i = q_j$) $\vee$ (cost($c_i$) > cost($q_j$))
   then rest
   else moveTo($mostCooperativeCell(q_i)$)
```

```
//less-selfish-as-possible
let $q_j$ = lessConstrainedAgent($pAgents(q_i)$);
if (($q_i = q_j$) $\vee$ (cost($c_i$) < cost($q_j$))
   then rest
   else moveTo($mostCooperativeCell(q_i)$)
```

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.

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* : $\{c_1, \ldots, c_n\} \rightarrow \mathcal{N}$*:*

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

Published in Prasad, B., editor, *Second International Indian Conference on Artificial Intelligence (IICAI'05)*, 20-22 December 2005, Pune, India, pages 3009-3024, 2005.

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

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

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

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

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 $\mathcal{C}^{q_i}_{coop}$, 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 [10]. Therefore, we can choose any kind of method: 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

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, we add a limited memory of one cell 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*. The number in the cells represents the number of markers, i.e. the number of agents seeing the cell.

Published in Prasad, B., editor, *Second International Indian Conference on Artificial Intelligence (IICAI'05)*, 20-22 December 2005, Pune, India, pages 3009-3024, 2005.

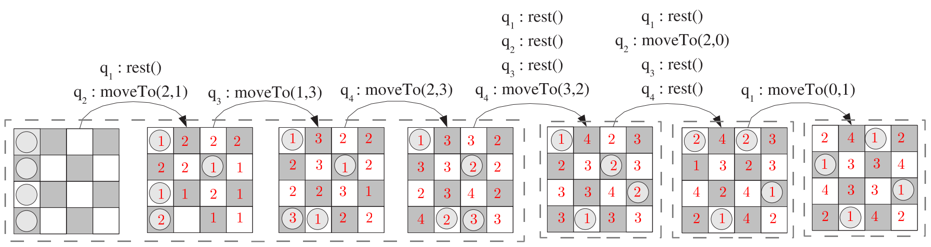

Fig. 2. Solving trace for 4 queens, with a *less-altruistic-as-possible* cooperative behavior, in 4 steps and 6 moves.

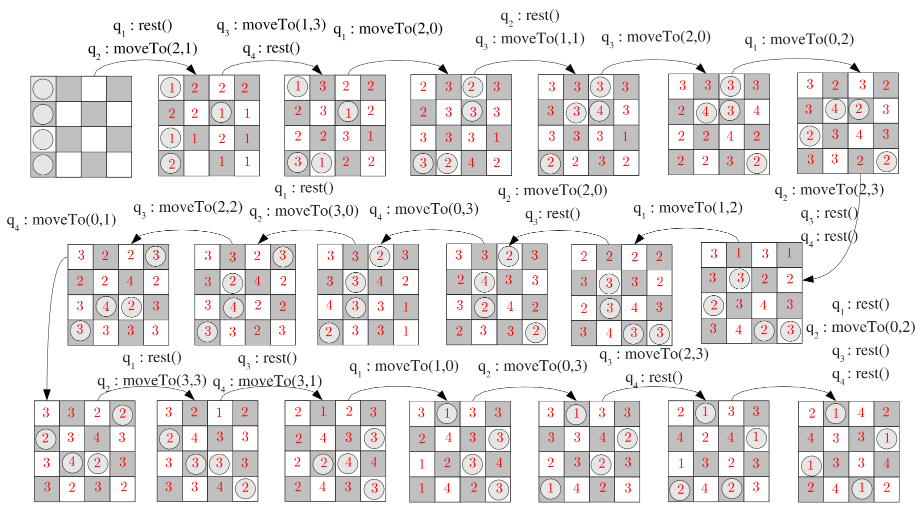

Fig. 3. Solving trace for 4 queens, with a *less-selfish-as-possible* cooperative behavior, in 8 steps and 19 moves.

***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 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.

***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.

Published in Prasad, B., editor, *Second International Indian Conference on Artificial Intelligence (IICAI'05)*, 20-22 December 2005, Pune, India, pages 3009-3024, 2005.

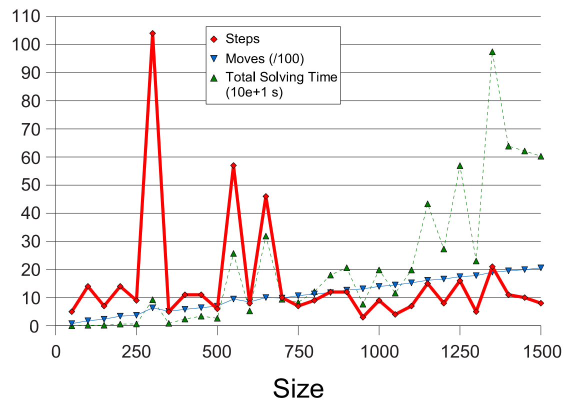

Fig. 4. Solving results for different sizes (from 50 to 1500) with *less-altruistic-as-possible* cooperative agents.

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

***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:

– the total number of moves (for all the agents) during the solving linearly increases in terms of the number of agents ($\mathcal{O}(4n/3)$),
– 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, $\mathcal{O}(n^2 s)$: 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), we cannot, by now, formally define it. But experimentally, we can overestimate it at $\mathcal{O}(n)$.

Therefore, the global complexity of the solving process can be experimentally overestimated at $\mathcal{O}(n^3)$, which is a common complexity to solve this problem with classical informed heuristics like hill-climbing [12], whereas the proposed algorithm is not informed.

***Less-selfish-as-possible* behavior.** Figure 4 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

Published in Prasad, B., editor, *Second International Indian Conference on Artificial Intelligence (IICAI'05)*, 20-22 December 2005, Pune, India, pages 3009-3024, 2005.

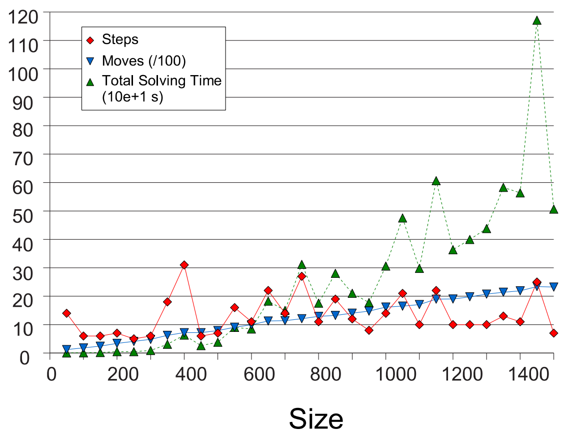

Fig. 5. Solving results for different sizes (from 50 to 1500) with *less-selfish-as-possible* cooperative agents.

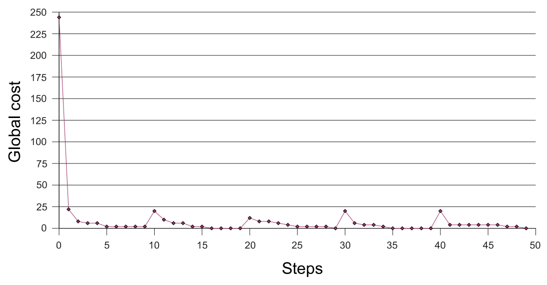

Fig. 6. Global cost in terms of solving steps for 512 *less-altruistic-as-possible* cooperative agents.

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

In section 1, we mainly focus 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.

Published in Prasad, B., editor, *Second International Indian Conference on Artificial Intelligence (IICAI'05)*, 20-22 December 2005, Pune, India, pages 3009-3024, 2005.

The figure 6 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.

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. In this section, we discuss 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 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, we can easily see, that we can obtain interesting results, with only minimum changes in algorithms. The only modification concerns the perceived and attackable cells.

Figure 7 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 positive in the sense that generalizing to other CSP with two domains is quite immediate.

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

Published in Prasad, B., editor, *Second International Indian Conference on Artificial Intelligence (IICAI'05)*, 20-22 December 2005, Pune, India, pages 3009-3024, 2005.

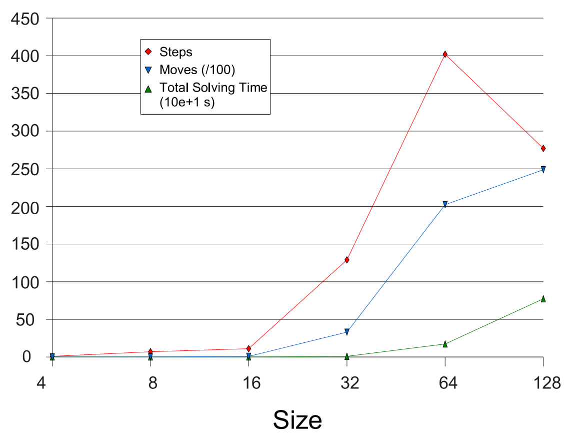

Fig. 7. Solving results for different size of $N^2/2$ knights problems (N from 4 to 128) with a *less-altruistic-as-possible* cooperative behavior.

developed to illustrate the ADELFE methodology [11]. This application requires to equip agents with limited memory of the other agents and the previous occupied cells [13]. We also shown positive results on adaptation and robustness to environmental disturbance such as constraint modifications or agent removals.

7 Discussion

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, the agents always act in the same order. But, as we have shown, the solving process also works with random disturbances. Some other simulations, which are not presented in this paper, 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.

Another important point of discussion is the halting property of the proposed algorithm. By now, it has not been formally proved that the collective can find the adequate organization in a finite solving time, contrary to classical distributed and dynamic approaches [2, 14]. This is mainly due to the huge gap between micro-level entities and the macro-level whole system. But this halting property remains secondary, and we mainly inspect the termination experimentally in most of the cases –like in optimization domains or in the ant approach [15].

Finally, our approach can be briefly compared to chosen existing ones. The most popular algorithms to solve DCSP are ABT (*Asynchronous Backtracking*) and AWCS (*Asynchronous Weak Commitment Search*) [2]; but they have some weaknesses as highlighted in [14]. Finally, distributed and dynamic CSP have been defined to add constraint updating possibilities. In this modeling, decision making is performed by exploring the constraint graph. Several search algorithms have been defined, such as DisDB, which is sound, complete and terminates [14]. Nevertheless, these approaches and the related solving algorithms only consider the distribution of variables and their main focus is completeness and correctness properties. The other family of CSP solving methods, which are influenced by optimization and meta-heuristics, considers the problem in a local[5] 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) [5]. 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 [4]. 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 [16]. 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 [17], 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 we provide enable adding or removing agents with minimal global disturbances.

8 Conclusion

In this paper, we 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, we propose to analyze the behaviors which tends to cooperation; from the altruist and the selfish viewpoints.

To illustrate this work, the example of the N queens problem has been implemented and commented. This implementation shows promising results on adaptivity and robustness when the system is subject to environmental disturbances. This implementation also raises 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 meta-heuristics to implement optimized search processes. Cooperation seems a relevant meta-level criterion, but requires more studies about memory charge, complexity and halting. This

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

Published in Prasad, B., editor, *Second International Indian Conference on Artificial Intelligence (IICAI'05)*, 20-22 December 2005, Pune, India, pages 3009-3024, 2005.

last point –even if experimentally obtained– may become a good panel to cooperative self-organizing systems and represents our main perspective.

References

1. Georgé, J.P., Gleizes, M.P., Glize, P., Régis, C.: Real-time Simulation for Flood Forecast: an Adaptive Multi-Agent System STAFF. In Kazakov, D., Kudenko, D., Alonso, E., eds.: Proceedings of the AISB'03 symposium on Adaptive Agents and Multi-Agent Systems(AAMAS'03), University of Wales, Aberystwyth (2003)
2. Yokoo, M., Durfee, E., Ishida, Y., Kubawara, K.: The Distributed Constraint Satisfaction Problem : Formalization and Algorithms. IEEE Transactions on Knowledge and Data Engineering **10** (1998) 673–685
3. Modi, P.J., Shen, W., Tambe, M., Yokoo, M.: An Asynchronous Complete Method for Distributed Constraint Optimization. In: Proceedings of the Second International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS'03). (2003) 161–168
4. Kirkpatrick, S., Gellat, C., Vecchi, M.: Optimization by Simulated Annealing. Science **220** (1983) 671–680
5. Glover, F., Laguna, M.: Tabu Search. Kluwer (1997)
6. Kohonen, T.: Self-Organising Maps. Springer-Verlag (2001)
7. Camps, V., Gleizes, M.P., Glize, P.: A Theory of Emergent Computation Based on Cooperative Self-Organization for Adaptive Artificial Systems. In: 4^{th} European Congress of Systems Science, Valencia. (1999)
8. Maturana, H., Varela, F.: The Tree of Knowledge. Addison-Wesley (1994)
9. Bernon, C., Camps, V., Gleizes, M.P., Picard, G.: Designing Agents' Behaviors and Interactions within the Framework of ADELFE Methodology. In Omicini, A., Petta, P., Pitt, J., eds.: Fourth International Workshop on Engineering Societies in the Agents World (ESAW'03). Volume 3071 of Lecture Notes in Computer Science (LNCS)., Springer-Verlag (2003) 311–327
10. Capera, D., Georgé, J., Gleizes, M.P., Glize, P.: The AMAS theory for complex problem solving based on self-organizing cooperative agents. In: 1^{st} International TAPOCS Workshop at IEEE 12th WETICE, IEEE (2003) 383–388
11. Picard, G., Gleizes, M.P.: The ADELFE Methodology – Designing Adaptive Cooperative Multi-Agent Systems. In Bergenti, F., Gleizes, M.P., Zambonelli, F., eds.: Methodologies and Software Engineering for Agent Systems (Chapter 8), Kluwer Publishing (2004) 157–176
12. Russel, S., Norvig, P.: Artificial Intelligence: a Modern Approach. Prentice-Hall (1995)
13. Picard, G., Bernon, C., Gleizes, M.P.: ETTO : Emergent Timetabling Organization. In: 4^th International Central and Eastern European Conference on Multi-Agent Systems (CEEMAS'05), 15-17 September, Budapest, Hungary. (2005)
14. Bessière, C., Maestre, A., Meseguer, P.: Distributed Dynamic Backtracking. In: Proceedings of the Workshop on Distributed Constraints, in IJCAI-01, Seattle, WA, United States. (2001)
15. Socha, K., Knowles, J., Sampels, M.: A MAX-MIN Ant System for the University Timetabling Problem. In: Proceedings of 3^{rd} International Workshop on Ant Algorithms, ANTS'02. Volume 2463 of LNCS., Springer-Verlag (2002) 1–13
16. Dorigo, M., Maniezzo, V., Colorni, A.: Ant System: Optimization by a Colony of Cooperating Agents. IEEE Transactions on Systems, Man and Cybernetics- Part B: Cybernetic **26** (1996) 29–41

17. Rogers, A., David, E., Jennings, R.: Self-Organized Routing for Wireless Micro-Sensor Networks. Special issue of IEEE Transactions on Systems, Man and Cybernetics, Part A - Self-organization in Distributed Systems Engineering (2005)