Distributed Frequency Assignment Using Cooperative Self-Organization

Gauthier Picard, Marie-Pierre Gleizes, Pierre Glize

Distributed Frequency Assignment Using Cooperative Self-Organization

Gauthier Picard, Marie-Pierre Gleizes and Pierre Glize
IRIT - Université de Toulouse III
118, route de Narbonne, F-31062 Toulouse cedex 9 FRANCE
{picard, gleizes, glize}@irit.fr

Abstract

This paper presents an approach using self-organizing multi-agent systems to solve distributed constraint satisfaction problems (DCSP), which concerns distribution among agents which task is to assign personal variables to values with respect with known constraints. Agents only know their variables and the constraints affecting them, and have to negotiate to find a collective solution. The proposed approach defines cooperative self-organization as the process leading the collective to the solution: agents can change the organization by their own decision to improve the state of the system. This work is illustrated on distributed frequency assignment, a classical constraint-based problem.

1. Introduction

Multi-agent systems are a design paradigm to tackle distributed and difficult problems such as autonomic computing, ad-hoc networks, manufacturing control or supply chaining [10, 4]. Once plunged in dynamic environments, such systems have to respond to changes and pressure. Since centralized methods can no longer provide answers for really distributed and huge problems, the adaptive multi-agent system (AMAS) approach considers that decision to change the function of the system is owned by the parts of the system (the agents). These agents are then able to change the organization of the system to modify the behavior of the whole system [1].

For agents, the trigger to decide when to change the organization locally is *cooperation*. Cooperation is considered as the boundary between two kinds of behaviors: "less altruistic as possible" and "less selfish as possible" [12]. Once an agent is no more cooperative (conflict, uselessness, concurrency, etc.) it modifies the organization of the system by changing its position within the environment and the other agents it knows. This approach, called *cooperative self-organization*, has already been applied to difficult problems such as distributed university time tabling [11, 12].

This paper presents a new solution for distributed constraint-based problems using cooperative self-organization. This work is close to well-known ABT (Asynchronous Back-Tracking), ADOPT (Algorithm for Distributed Constraint Optimization) or AWC (Asynchronous Weak Commitment) algorithms [14, 9], but does not require a total order among agents, and is comparable to local search-based algorithms such as tabu search. The AMAS approach is here applied to a distributed *frequency assignment with polarization problem* (FAPP) which is a distributed version of the classical frequency assignment problem [2].

This document is structured as follows. First, the existing multi-agent-based approaches to tackle distributed constraint satisfaction and optimization problems are presented. In section 3, the FAPP is formally described. The proposed approach and algorithms are then presented in sections 4 and 5, and some results are shown in section 6 and discussed in section 7. Finally, section 8 concludes the paper and draws some perspectives to this work.

2. Multi-Agent Constraint Solving

Many decision problems can be expressed as *constraint satisfaction problems* (CSP). A problem is there expressed as a set of variables (the objectives) which can be assigned to values in given domains with respect to a set of constraints. A CSP is a triplet $\langle X, D, C \rangle$ such that: $X = \{x_1, \ldots, x_n\}$ is a finite set of variables to assign values to, $D = \{D_1, \ldots, D_n\}$ is a finite set of domains for each variable, $C = \{c_1, \ldots, c_m\}$ is a finite set of constraints on the values of variables. Domains D_i can be finite or infinite, and discrete or continuous. Moreover constraints from C can be expressed as tuples of impossible values or as predicates on several variables. This implies several way to model a problem into a CSP, and therefore several kinds of solvers to support them.

A *distributed constraint satisfaction problem* (DCSP) is a quintuplet $\langle A, X, D, C, f \rangle$ such that: $\langle X, D, C \rangle$ is a CSP, A is a finite set of agents, $f : A \times X \mapsto \{true, false\}$ is a predicate which assigns variables to agents. Solving a DCSP requires providing agents with behavior as to negotiate to find a solution. In fact, agents do not know the whole problem and must interact to get some information useful for solving their constraints. DCSP framework was firstly introduced by Yokoo et *al.* who propose asynchronous algorithms inspired from non-distributed search algorithms [14]. ABT (Asynchronous Back-Tracking) is an asynchronous version of a backtrack algorithm with a *predefined order* on agents to ensure consistency and exchange of nogoods. AWC (Asynchronous Weak Commitment) is an enhancement of ABT with two main extensions: *min-conflict* heuristic and dynamic agent ordering depending on agents' nogoods and their neighborhood. AWC is faster than ABT to find a solution, but ABT is faster to prove a solution is not satisfiable. These two algorithms are complete. Other works inspired from on DCSP are based on more local view for agents, but are not complete such as DynDBA (Dynamic Distributed Breakout Algorithm) [8].

There exist problems for which no solution is satisfiable. But, by defining a cost function for each constraint, the problem becomes an optimization problem in which the objective function to optimize (maximize or minimize) is a combination of the cost functions, such as the sum or a weighted sum, for example. These problems, once distributed among agents, are called *distributed constraint optimization problems* (*DCOP*). The most popular multi-agent algorithms to solve such optimization problems are ADOPT (Algorithm for Distributed Constraint Optimization) [9] and its successors, and the ERA (Environment, Reactive rules, and Agents) approach [6], in which agents evolve within the domains of their variables and follow different stochastic behaviors to converge towards a solution.

In this paper, only the satisfaction part of the frequency assignment problem with polarization is expounded, within a potential dynamic environment. Still, ERA is kept in mind for their approach by positioning.

3. Frequency Assignment Problem

3.1. Presentation

The frequency assignment with polarization problem (FAPP) consists in finding frequency allocation within a hertzian telecommunication network, which is composed of several transmitter and/or receiver sites [2]. A hertzian connection can be constituted of one or more radioelectrical unidirectional paths between antennas of distinct geographical sites. Each path is characterized by its frequency and its polarization, which can get values within predefined domains.

Let T be the set of paths. Let $F_0, F_1, ..., F_n$ be the n domains of frequencies. Let $P_{-1} = \{-1\}, P_1 = \{1\}, P_0 = \{-1, 1\}$, the three possible domains of polarization. At each $i \in T$, we associate (f_i, p_i) such that $f_i \in F_{\varphi(i)}$ and $p_i \in P_{\pi(i)}$, where $\varphi(i)$ and $\pi(i)$ indicate respectively the frequency and polarization domain numbers of i.

There exist two types of constraints on these networks. Physical constraints related to the proximity or, at the contrary, to the geographical distance. These interference constraints are called electromagnetic constraints (CEM), and are soft constraints. The difference of polarization is a technical means to diminish interference problems between frequencies caused by these constraints. The second type of constraints are imperative technological hard constraints (CI), which impose a distance between the frequencies of two paths. In matrix form, the constraints are:

$$\begin{cases} (CEM) & |f_i - f_j| \geq \frac{|p_i+p_j|}{2}\gamma_{ij}^{(k)} + \frac{|p_i-p_j|}{2}\delta_{ij}^{(k)} \\ & \text{with } k = 0 \text{ to } 10 \\ (C2) & |f_i - f_j| = \epsilon_{ij} \\ (C3) & |f_i - f_j| \neq \epsilon_{ij} \\ (C4) & f_i = f_j \\ (C5) & f_i \neq f_j \\ (C6) & p_i = p_j \\ (C7) & p_i \neq p_j \end{cases}$$

where γ_{ij} and δ_{ij} indicate required frequency distance as a function of polarization.

A feasible solution corresponds to the assignment of a frequency and a polarization to each path by satisfying the set of given constraints. The problem being often over constrained, CEM can be relaxed over 11 levels (from 0 for satisfaction to 10). The objective is therefore twofold:

1. to satisfy the constraints C2, C3, C4, C5, C6 and C7,
2. to minimize a cost function, which is not described here, since the paper only addresses the satisfaction part of the FAPP. More details on this function are available in [2].

Halfway between real problems and case studies (such as graph coloring or n queens problems), the study of FAPP represents an interesting step towards the resolution of real constraint-based problems. The heterogeneity of the constraints and the large sizes (up to 3,000 paths) make the FAPP be an interesting challenge. Nevertheless, two characteristics make its study easier: finite and discrete domains and binary constraints. The originality of the FAPP also lies in the melting of hard and soft constraints. The question is both a constraint satisfaction problem and an optimization one. Therefore, this implies modeling and implementing

bivalent algorithms to *necessarily* process a set of hard constraints and next to *possibly* improve the global state by focusing on soft constraints. This paper only focuses on the first objective.

3.2. Towards the Distribution of FAPP

There exist several efficient solutions to tackle the FAPP: local search and tabu search [3], lower bounding and tabu search procedures [5], or arc-consistency and tabu search [13]. All these approaches are based on tabu search and are not distributed, nor adaptive. These algorithms are centralized and manage all the network. If a change occurs during the solving process, all the process must be relaunched. For example, if a site disappears (transmitter/receiver dysfunction), these systems are not able to respond by providing a new assignment from the current one.

Several constraints naturally appear when using the agent paradigm, from the viewpoint of dynamic and distributed problems solving. First, the absence of global control makes difficult the separation into phases, as it appears in the three previously quoted algorithms. Second, the dynamical aspects, i.e. the capability to add or remove constraints or paths in a FAPP during the solving process, prevents using arc-consistency pre-processing.

However, several concepts can be reused at the agent level:

- the concept of *penalty* on values used by [13] to force exploring the domain of a path,
- the concept of *criticity* of a variable or a value, used by [3] to favor, at the contrary, some values or to make some agents have priority,
- the concept of *pairs of paths* linked by C2 or C4 constraints (cf. 3.1) used by [5], which can be implemented as a dialog between agents,
- the concept of *tabu*, in general, and forbidding some paths to modify their values, in particular, can be transposed at the agent level.

Besides the fact that tabu approaches are outstanding to solve the FAPP, this approach can be reused at the agent level, at the contrary of methods using variable neighborhood on the search space. The multi-agent approach presented in the next sections, based on AMAS, brings a novel point of view, different from the previous approaches, since agents self-organize to find a solution.

4. Self-Organizing Multi-Agent Modeling

In this section, a multi-agent system to solve the FAPP, by using cooperative self-organization, is described.

4.1. Problem Specification

To describe the MAS, here are some characteristic data of the FAPP:

- T, the set of paths,
- $\forall t \in T,\ F_t$ the frequency domain associated to t,
- $\forall t \in T,\ P_t$ the polarization domain associated to t,
- C, the set of constraints,
- CI, the set of hard constraints,
- CEM, the set of soft constraints (CEM),
- $\forall t \in T,\ C_t = CI_t \cup CEM_t$, the set of constraints of path t,
- $\forall c \in C,\ t_1(c)$ and $t_2(c)$, the two paths linked by the constraint c,
- $\forall t, u \in T,\ v(t,u) \equiv \exists c \in (C_t \cap C_u)$, the neighborhood relation between two paths,
- $\forall t \in T,\ V_t = \{u \in T \mid v(t,u) = true\}$, the set of neighbors of the path t, called *neighborhood*.

The system to model includes a set of autonomous agents, which are homogeneous concerning their representations and behaviors. A distinct agent is associated with each path (or variable). Let A be the set of agents of the system and let T be the set of paths. For all agent $x \in A$, we note $t_x \in T$ the path associated to the agent. So, $f(x, t_x) = true$, by using the notation of DCSP in section 2. An agent x is the neighbor of an agent y if and only if their paths are neighboring (they share a constraint). The neighborhood of an agent, i.e. the social environment, is defined as the set of its neighbors:

$$\forall x, y \in A,\ v(x,y) \equiv v(t_x, t_y)$$
$$\forall x, y \in A,\ y \in V_x \equiv t_y \in V_{t_x}$$

An agent has the same frequency domain than the variable its represents:

$$\forall x \in A,\ F_x = F_{t_x}$$
$$\forall x \in A, P_x = P_{t_x}$$

Hard constraints and CEM are known by each agent the variable depends on. Let C_x be the set of constraints known by an agent:

$$\forall x \in A,\ C_x = C_{t_x}$$

A satisfied constraint $c \in CI_x$ is noted $c = true$.

As this paper only deals with hard constraints, CEM are not formally presented here.

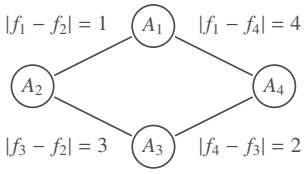
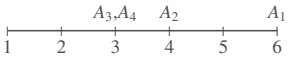


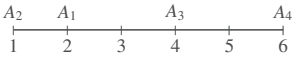
Figure 1. Simple example with 4 agents and 4 hard constraints

In the example of fig. 1, $A = \{A_1, A_2, A_3, A_4\}$. The agents have the same domains $D_f = \{1, 2, 3, 4, 5, 6\}$ and $D_p = \{1\}$. The problem is simplified by putting aside the polarization. There are only two types of constraints: C2 and C4, i.e. distance constraints. For example, a constraint exists between A_1 and A_2: $|f_1 - f_2| = 1$. It is easy to find solutions for this FAPP, which are, for $(f_{A_1}, f_{A_2}, f_{A_3}, f_{A_4})$: $(2, 1, 4, 6)$ and $(5, 6, 3, 1)$.

Therefore, the problem of frequency assignment can be viewed as a positioning problem in which agents represent frequencies, as viewed in ERA [6]. Each agent must find a right position (value) within their domain, by following a self-organizing process. At the beginning of the process, agents are randomly positioned, as follows, for example, with a shared domain:



One possible solution is therefore:



To pass form the initial state to the solution, we propose to use self-organization and negotiation between agents. Each agent aims at finding the right place within the organization by using only local interactions with its neighbors (agents linked by constraints) and deciding by itself to move from one position to another. Local criteria, inspired by *cooperation*, are embedded in agents to implement their decision making. This notion is here more than sharing resources or skills to perform a common task, but it also encapsulates the capability to anticipate conflicts and to act to repair or to remove non cooperative situations, such conflicts.

4.2. Cooperative Agents' Characteristics

Agents are the only active entities of the system, which interact to cooperatively solve the FAPP. Two main data characterize an agent, except its environment: its *value* and its *difficulty*. The value of an agent x is the current pair $(f, p)_x$ with $f \in F_x$ and $p \in P_x$. The difficulty d_x of an agent x is a quantitative measure which take into account the current environment and value of an agent. The higher it is, the more the agent believes it is far from a solution, as a heuristic does. This measure is the basis for cooperative mechanisms. For an agent, the difficulty is its own and is more detailed in section 5.

In addition to these two values, the agent has a view on its environment. This view is partial. An agent only knows the value and the difficulty of each of its neighbor, but does not know constraints, views, nor domains. The communication between agents is implemented with messages.

Each agent evolves within an own *physical* environment which consists in the domains of frequency and polarization. These domains, even if possibly identical, are not shared. In the same manner, each constraint is duplicated for each agent knowing it.

The coherence between an agent's view and its constraints is always ensured. Every changes of the view implies changes of the state of the constraints (satisfied or not satisfied). However, the global coherence between the views of different agents is not guaranteed since messages still flow. In fact, the view of a neighbor on an agent can differ from the current state of the agent. Therefore, constraints between two agents can be in different state. To ensure the global coherence once the system is stopped, some hypothesis, raised by Yokoo et *al.* for DCSP solving, must be assumed [14]:

1. messages are received in a finite time,
2. messages are not lost or changed during transmission,
3. messages are always processed by the agent receiving it.

To avoid ambiguity concerning the views of agents:

- let $c^x \in C_x$ be the constraint c from the point of view of x,
- let d_x^y be the difficulty of x from the point of view of y,
- let $(f, p)_x^y$ be the current value of x from the point of view of y.

4.3. Cooperative Agents' Behavior

The behavior of the cooperative agent is composed of 4 steps, as shown in fig. 2. This algorithm is common to all the agents which run it concurrently.

Initialization consists in choosing (randomly, for example) a value within the domains (line 1) and informing the neighborhood that the agent is running (line 2).

As communication is implemented with messages, each agent owns a mailbox it *checks* eventually. Messages are

```
   // Step 1: Initialization
 1 selectValue;
 2 sendValuesToNeighbourhood;
 3 while isRunningSystem do
   |  // Step 2: Check
 4 |  checkMessages;
   |  // Step 3: Decision
 5 |  if receivedAllValues then
 6 |  |  evalDifficulty;
 7 |  |  if isDifficultyChange then
 8 |  |  |  sendValuesToNeighbourhood;
 9 |  |  end
10 |  |  if receivedAllDifficulties then
11 |  |  |  if mostDifficult then
   |  |  |  |  // Step 4: Assignment
12 |  |  |  |  assignment;
13 |  |  |  end
14 |  |  end
15 |  end
16 end
```

Figure 2. The 4-step behavior of an agent

processed consequently to the check from the oldest to the newest. This phase ends when the mailbox is empty (line 4).

Decision leads to choose the agent that will act to change the global solution by assigning a new value (line 11), in a given neighborhood. An agent decides only if it knows the difficulties of its neighborhood (line 10). Moreover, an agent can only calculate its difficulty (line 6) if it knows the values of its neighbors (line 5). These are not blocking conditions since agents communicate data as soon as they get them (cf. section 4.4). From its point of view, if an agent has the highest difficulty, for a given neighborhood[1], it is *elected* as the agent that have to act for the neighborhood. With equal maximum difficulties, the agent is randomly elected.

For $x \in A$, we define $e(x)$ the fact that x is eligible:

$$\forall x \in A,\ e(x) \equiv \forall y \in V_x,\ d_x^x \geq d_y^x$$

Therefore, this is the agent that has the highest difficulty which will try to improve the current situation. This approach is preferred to an approach considering this is a neighbor of the agent with the maximum difficulty that is chosen. In fact, a neighbor has only a limited number of constraint with an agent, and therefore is not well placed to help it. This phase underlines a first aspect of the cooperation notion: *agents let neighbors with more difficulty than themselves act*. It is a self-organized process.

Assignment is only accessible to elected agents. The elected selects, for its current value in its domain, a value it believes being the best one for itself and its neighborhood. There is no way for an agent to force other agents to change their values. If more than one value seems correct, the final value is randomly chosen. At the end of the assignment phase, the agent disables itself (only for decision): it informs its neighborhood that it will not participate to the next elections until another agent is elected. This is an *egalitarian* policy that allows other agents to act. A disabled agent can however be invited to an assignment session (cf 4.4). The state (enable/disable) of an agent is known by its neighbors and can be interpreted as a *local tabu*.

Once the system is running, all the agents follow the *check-decision-assignment* cycle without being able to stop the process by themselves.

[1] Therefore, there may be several elected agents within the system.

4.4. Communication Protocol

An agent only communicates with its neighbors, by using direct messages containing the view (i.e. the displayed values) of the sender, the view of the receiver, and the assignment session if necessary (cf. next paragraphs). Moreover, an agent systematically informs its neighbors if changes of value or difficulty occur, not to distort the views and the decisions of its neighbors.

Not to overload the communication network, here are the triggers to send messages:

- after *initialization*,
- during *check*, at each message reception:
  - either the value of the neighbor has changed, therefore the difficulty must be calculated,
  - or the neighbor is disabled, therefore the view of the state must be changed,
  - or the difficulty of the neighbor has changed or it is enabled.
- at the end of the *check* step, the agent sends a message to its neighborhood only if a change of difficulty occurred consequently to the change of a value of a neighbor or if it changed its state (lines 7-8).
- at the end of the *assignment* step, the agent disables itself and informs its neighbor of the change of values, state or difficulty.

To prevent *deadlocks*, it is necessary to introduce a synchronization point during the assignment. This is implemented by using the notion of *assignment session*. For a given neighborhood, an elected agent first creates an assignment session, which is a synchronized object (monitor or equivalent). Then, it sends invitations to its neighbors, with its last value, state and difficulty. A neighbor can then accept or reject the invitation if either it is already within an assignment session for which the elected is more in difficulty or it has itself a higher difficulty than the elected. Once the elected has received all the answers, it begins the assignment. Once finished, it informs its invited neighbors

that the session is correctly ended by sending its new value. Until the session is not ended, the *invited agents* do not process the messages coming from the elected agent.

5. Cooperative Criteria for Hard Constraints

5.1. Difficulty Measurement

Difficulty is the decision criterion for electing an agent. It is a sorted tuple of sub-criteria. The difficulty for an agent $x \in A$ viewed by an agent $y \in A$ is noted:

$$d_x^y = [Im_x^y, Po_x^y, NS_x^y, Ag_x^y]$$

The *improvement* (or Im), of an agent is the best possible improvement for this agent considering the current situation by counting the number of unsatisfied hard constraints. For all $x \in A$, for all $(f, p) \in F_x \times P_x$, $NS_x(f, p)$ is the number of unsatisfied hard constraints with the value (f, p). With $NS_x \equiv NS_x(f, p)_x$, we note:

$$\forall x \in A,$$
$$Im_x = NS_x - min\{NS_x(f, p),\ (f, p) \in F(x) \times P(x) \setminus (f, p)_x\}$$
and
$$\forall x \in A, \forall y \in V_x,$$
$$Im_x^x > Im_y^x \Rightarrow d_x^x > d_y^x$$

The cost to calculate this criterion is $|F_x \times P_x - 1|$ since the current value is not included. The idea of this criterion is to consider that the agents, that know they can strictly improve their local situation, know they are not well positioned. This requirement of projecting forward, even if costly, significantly improves the search for solutions.

Contrary to Im, the next criteria are not based on domains but on constraints. For all $x \in A$, for all $c^x \in CI_x$, let $FPS(c^x)$ be the set of pairs $(f, p) \in F_x \times P_x$ which satisfy c^x from the current view of x.

The second criterion, *possibilities* (or Po), expresses the fact that a constraint is difficult if there only exist few possible assignments that satisfy the constraint:

$$\forall x \in A,$$
$$Po_x = min\{|FPS(c^x)|,\ c^x \in CI_x \text{ and } c^x = false\}$$

and in case of equality of the previous criterion (Im):

$$\forall x \in A,\ \forall y \in V_x,$$
$$\text{s.t. } Im_x^x = Im_y^x,$$
$$Po_x^x < Po_y^x \Rightarrow d_x^x > d_y^x$$

The third criterion is the *number of unsatisfied hard constraints* (or NS_x). This criterion assumes that an agent having more unsatisfied constraints has more difficulties, in case of equality of the previous criteria (Im then Po):

$$\forall x \in A,\ \forall y \in V_x,$$
$$\text{s.t. } Im_x^x = Im_y^x \text{ and } Po_x^x = Po_y^x,$$
$$NS_x^x > NS_y^x \Rightarrow d_x^x > d_y^x$$

Finally, the last criterion to measure the difficulty is the *oldness* (or Ol). For all $x \in A$ and for all $c^x \in CI_x$, let $TI_x(c^x)$ be the number of assignments performed by x or its neighborhood since the last satisfaction it knows. This represents the oldness of constraints and we set:

$$\forall x \in A,\ Ol_x = max\{\ TI_x(c^x,\ c^x \in CI_x|)\}$$

The agent having the oldest constraint is in the more difficulty, in case of equality of the previous criteria:

$$\forall x \in A, \forall y \in V_x,$$
$$\text{s.t. } Im_x^x = Im_y^x \text{ and } Po_x^x = Po_y^x \text{ and } NS_x^x = NS_y^x,$$
$$Ol_x^x > Ol_y^x \Rightarrow d^x > d^y$$

5.2. Other Decision Criteria

Besides the difficulty notion, there exist two other criteria to elect agents. First, an agent will try to be elected if and only if it owns unsatisfied constraints, not to elect agents that will not change the current state of the system:

$$\forall x \in A, solved(x) \equiv \forall y \in V_x \cup \{x\},\ NS_y^x = 0$$

If two agents have the same difficulty at this time of the comparison, an agent is randomly chosen (Eq).

The last criterion is the fact that an agent which is the only active agent for a given neighborhood is automatically elected (De).

5.3. Choosing Values

The key idea for choosing values in a cooperative manner is that an agent will select the value to assign as a function of the criterion that trigger its election. This criterion, called *discriminant criterion*, that discriminates agent x from agent y, is noted cd_y^x. The criterion for x to select the action to perform is called *selection criterion* and noted cs^x:

$$\forall x \in A,\ cs^x = \ min\{cd_y^x,\ y \in V_x\}$$

A different manner of choosing possible values is associated to each decision criterion:

- $cs^x = Im$: select the values that maximize the improvement,
- $cs^x = Po$: select the set of constraints with the minimum of possibilities,
- $cs^x = NS$: select the set of constraints with the maximum of possibilities,
- $cs^x = Ol$: select the oldest constraints,
- $cs^x = Eq$: select the constraints shared with the agents having a difficulty equal to the agent's difficulty,

- $cs^x = De$: select the set of all the constraints for the agent.

Let VEl_x be the set of values chosen by x.

Nevertheless, there is a drawback to make this choice using the sets of constraints: all the constraints can be unsatisfiable in the situation and therefore, all the values are possible except the current one. It implies implementing some dialog capabilities between an elected and the invited agents.

5.4. Dialog with Invited Agents During a Session

An assignment is useful for a neighbor in two situations:

- this assignment improves *immediately* the situation of the neighbor,
- this assignment improves *possibly* the situation of the neighbor, by leading it, for example, to assign a value that decreases its number of unsatisfied constraints, if the neighbor is elected in the future.

Let $S_x(f, p)$ be the number of satisfied hard constraints of the agent x with the value (f, p). The exchanged data to represent the utility of a value (f_x, p_x) of an elected agent x for a neighbor y is:

$$u_x^y(f_x, p_x) = max\{|S_y(f_y, p_y)|,\ (f_y, p_y) \in F_y \times P_y \text{ and } (f, p)_x^y = (f_x, p_x)\}$$

The cost of this calculus is initially high: for each value of the elected, a matrix of the number of constraints satisfied must be calculated. But once calculated a first time, it only requires updates.

With this utility for each value and each neighbor, the elected agent x selects the values that maximize the sum of the utilities of its neighbors:

$$VNe_x = max\{\sum_{y \in V_x} u_x^y(f, p) \mid (f, p) \in VEl_x\}$$

If there still exist several equivalent values, x does the same process by only selecting values in VNe_x. This last selection is possible because it appends after the choice of values relatively to the selection criterion and before the participation of its neighbors: the agent has achieved its tasks, has requested its neighbors' proposals and thus it can satisfy its own needs. This represents the *equilibrium between altruism and selfishness* that defines the notion of cooperation, as presented in [12].

Finally, if there still exist equivalent values after this selection, one of them is chosen *randomly*.

Table 1. Solving trace for the FAPP of fig. 1

	Step 1		Step 2		Step 3		End
A_1	***3*** [1,0,2,1]	1,2,4,***5***,6	5 [1,0,2,2]	0	5 [0,0,1,3]	1 (5)	5
A_1-A_2	[2,1] (2,4)		[2,2] (2,4)		[1,0] (5)		
A_1-A_4	[0,1]		[0,2]		[0,3]		
A_2	3 [1,1,2,1]	0,0,0,1(6),0	***3*** [***2***,1,2,2]	6	6 [0,1,0,0]	n/a	6
A_2-A_1	[2,1] (2,4)		[2,2] (4,6)		[2,0] (4,6)		
A_2-A_3	[1,1] (6)		[1,2] (6)		[1,0] (6)		
A_3	3 [1,1,2,1]	n/a	3 [1,1,2,1]	0	3 [0,1,1,2]	1 (3)	3
A_3-A_2	[1,1] (6)		[1,1] (6)		[1,0] (3)		
A_3-A_4	[2,1] (1,5)		[2,1] (1,5)		[2,2] (1,5)		
A_4	3 [1,0,2,1]	1(5),0,0,1(1),0	3 [2,1,2,2]	n/a	***3*** [***2***,1,2,2]	1	1
A_4-A_1	[0,1]		[1,2] (1)		[1,2] (1)		
A_4-A_3	[2,1] (1,5)		[2,2] (1,5)		[2,2] (1,5)		

5.5. Example

Table 1 shows an example of solving trace for the FAPP presented in fig. 1. $(f_{A_1}, f_{A_2}, f_{A_3}, f_{A_4}) = (3, 3, 3, 3)$ is the initial state. Three steps are required to achieve a solution. Since all the agents are in a same neighborhood, there is no concurrent solving, contrary to larger problems.

For each step, data are presented as follows. A *decision part* (sub-column at left of each step), for each agent, is composed of the frequency, followed by the difficulty vector, and for each constraint (noted as $A_x - A_y$) a vector $[possibility, age]$ followed by the frequencies that can satisfy the constraint. For example, at the beginning, A_1's frequency is 3, its difficulty is $[1, 0, 2, 1]$, and the constraint $A_1 - A_2$ from A_1's viewpoint has two possibilities which are $(2, 4)$ and its age is 1.

An *invitation part* (sub-column at right of each step) for the elected agent shows the frequencies it believes as being possible. Its neighbors send, for each of these frequencies, the sum of possible improvement and the immediate improvement, with the frequency that implies this improvement (if any). For example, at step 1, A_1 is elected (frequency in italics) and presents the set of possible frequencies $VEl_{A_1} = \{1, 2, 4, 5, 6\}$ to its neighbors. A_2 sends $\{0, 0, 0, 1(6), 0\}$ to A_1, in order to inform that if A_1 chooses the frequency 5 then A_2 has a possible improvement of 1 by choosing the frequency 6. At step 2, A_1 is disabled (gray background) because it just assigns a new value.

At step 1, two agents are equivalent: A_1 and A_4. A_1 is randomly chosen and his discriminant criterion is therefore $cs^{A_1} = Eq$. A_1 looks at the constraint it shares with A_4: $A_1 - A_4$. As there is no possibility to satisfy this constraint, A_1 is not able to choose a good value, and it selects the whole domain except its current value: $\{1, 2, 4, 5, 6\}$. The answer of its neighbors is that the best choice is 5. That lets only one possible assignment for A_2 to 6 and for A_4 to 1.

Table 2. Experimental setup

FAPP Instance	01	11	22	33	38
Agents	200	1000	1750	650	2500
CI	168	978	1799	578	3112
Runs	400	400	67	400	48
Time limit (s)	60	160	1000	150	1400

Table 3. Percentage of unsatisfied CI

FAPP Instance	01	11	22	33	38
CI	168	978	1799	578	3112
% Unsatisfied	0	0,015	3,49	0,004	6,82

Table 4. Assignments

FAPP Instance	01	11	22	33	38
Assign.	107.24	547.02	964.81	338.35	1146.18
Assign./CI	0.64	0.56	0.54	0.59	0.37
Assign./Time(s)	17.87	3.42	0.96	2.26	0.82
σ	5.33%	6.89%	5.36%	2.28%	3.88%

At step 2, A_1 is disabled and A_2 and A_4 are elected with the criterion *Im*. Since agents can only be invited at one session at a same time, A_1, for example, rejects the invitation of A_4, which will wait until the assignment session of A_2 ends, to send another invitation. A_2 performs the assignment, for the criterion *Im*, and it looks at the values that imply the highest improvement, i.e. 6 which satisfies the two constraints of A_2. The dialog with its neighbors is useless since there is only one possible value: 6.

At step 3, A_4 is elected despite the difficulty changes for A_1 and A_3, and it reasons as A_2 did by choosing the value 1 that satisfies its two constraints. Once the assignment done, all the agents are solved: a solution has been found.

6. Results and Analysis

The different instances referred in this paper come from the ROADEF challenge 2001[2], have been ran on one computer several times, with specific time limits shown in table 2.

6.1. Convergence

The convergence towards solutions is shown in fig. 3. The important initial decreasing is due to the fact that some constraints are satisfied as soon as the random initialization is done. These constraints are often C3 or C5 constraints, since they are easier to satisfy. The decreasing appearing once the time limit is reached corresponds to the termination of agents that process their remaining messages before ending, without beginning new assignment sessions: they

[2]http://uma.ensta.fr/conf/roadef-2001-challenge/index_main.html

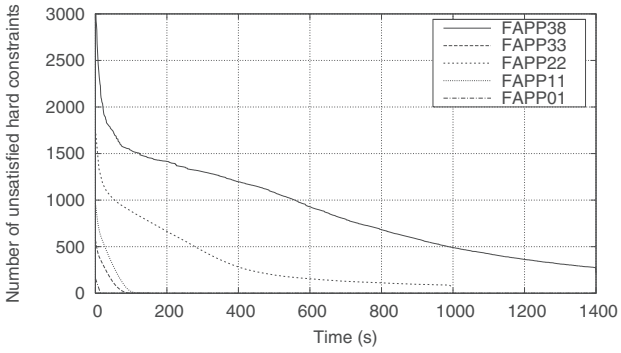


Figure 3. Convergence to solutions

reached a stable situation. But results for FAPP22 shows a stagnation: a local minimum is reached, and it remains, on average, 3.49% of unsatisfied constraints, as shown in table 3. Another problem is the time to solve a non complete problem (there is no CEM), which depends on the number of agents. Therefore, even if performing on small instances (200 agents), the approach does not manage to find solutions systematically.

6.2. Assignments and Messages Traffic

Table 4 shows there is only few assignments. Considering the size of the search space, the exploration is limited. In fact, the main idea behind the notion of cooperation is to make the good choices upstream to ensure a quick convergence, instead of exhaustively explore the search space: cooperation is here a *meta-heuristic*, in the sense criteria are not problem-specific. The small ratio between the number of assignments and the number of hard constraints confirms the process is strongly guided. The standard deviation σ is small too, in the order of 5%, which emphasizes the adaptation capability of the algorithm, independantly of the initial situation, and therefore adaptation to changes during the solving process.

However, the speed of assignment is low, and this ratio decreases with a great number of agents. But, at each time, there is more than one assignment session, and since the behavior is completely concurrent, the time, on distributed environment is far better, even by taking into account the message traffic, shown in table 5: there are few messages sent by agents. Moreover, there are few situations of conflict (assignment session canceling), as shown in table 6.

6.3. Decision Criteria Relevance

Among the criteria presented in section 5, fig. 4 shows which ones are used by the agents during the solving process. The value for the criterion *De* (when neighbors are already elected) is very small. But, the criterion *NS* (number

Table 5. Message traffic

FAPP Instance	01	11	22	33	38
Agents	200	1,000	1,750	650	2,500
Total Messages	2,335	15,841	33,247	8,348	48,727
Messages/Agent	11.7	15.8	19	12.8	19.5

Table 6. Percentage of canceling messages

FAPP Instance	01	11	22	33	38
% canceling	4.21	6.02	13.56	5.22	10.28

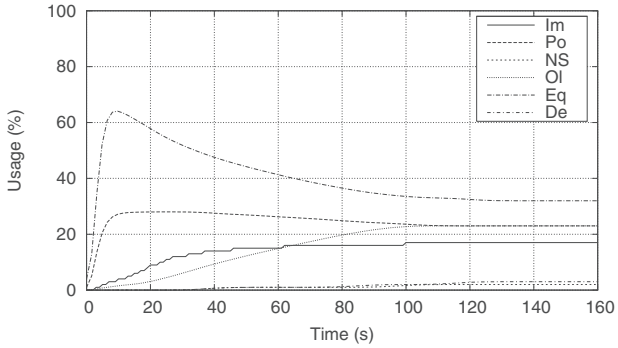


Figure 4. Distribution of the discriminant criteria for the instance FAPP11

of unsatisfied constraints) is not frequently triggered, contrary to *Ol*, *Im* and *Po* which are eventually used. Finally, average 30% of agents evaluate one of their neighbors as equal (*Eq*).

7. Discussion

7.1. Communication

Although there are few sent messages by agent, information updates can be limited even more outside of the assignment sessions. It is possible to only keep necessary updates (e.g. during initialization) and let the agents share and commit their views during assignment sessions during which canceling messages are more numerous: the system will become more synchronous. On the other side, to obtain more flexibility, invitations to assignment sessions which are blocking points may be limited too. But, without this synchronization point, the system is unable to stabilize since views constantly change. However, modifications can be considered: either reducing the time an invitation requires, because once the dialog is done it is no more needed to wait; or, limiting the number of invited neighbors, by using some preference measure and memory. Another approach can also be to allow agents to participate to more than one session at a time, while keeping them still blocked. In fact, there are only few sessions at a time.

7.2. Difficulty Evaluation

A first observation is that the criteria are *generic*: they are not relative to FAPP, even if *Po* is close to the FAPP notions. Moreover, the concept of pair of paths, introduced by [3] to manage particular cases of paths sharing C2 or C4 constraints, does not appear in the decision, even if it partially impacts on *Po*.

Second, sorting the criteria is one remaining open question, which validity is more complex to observe. Even if it seems relevant to position *Im* at the head (by analyzing the results of section 6.3), the other criteria can be discussed and experimented more precisely.

7.3. Soft Constraints and Optimization

Tackling CEM soft constraints is one perspective of this work. These constraints are far more numerous and increase the neighborhoods, and will put the communication protocol on the spot by imposing a great number of invited during sessions. To solve this problem, a possible track is to consider two kinds of neighborhoods: CI and CEM.

Afterwards, another problem is to include soft constraints in the difficulty measure. If it seems relevant to position them after the criteria for hard constraints to discriminates the numerous equality cases, as emphasized in section 6.3, it must not corrupt the solutions for CI. For that, new criteria must be modeled and new relative actions must be implemented.

7.4. Dynamic Problems

Introducing dynamics within the system, by adding or removing agents and constraints during the solving process is another interesting point to discuss and may be the easier one. This is possible thanks to the self-organization mechanisms, which are implemented as local and asynchronous, among agents. This algorithm can easily adapt to changing environments. The only point to warn is the assignment session: deleting agents which invited in sessions must be carefully studied. Results on the standard deviation (cf. section 6.3) comfort this idea.

Nevertheless, a drawback is the fact that the solving process relies on communication of information between agents, relatively to the assumptions of section 4.2. What if the communication network breaks down? By now, the system is robust because the problem can change in runtime, but it is not robust if the communication network is not reliable. One solution is to manage uncertainty on neighbor's

information to tackle the approximative rightness of information, for example. This subject will be prospected in future works.

7.5. Comparison to Other Approaches

Contrary to APO [7] or ABT [14] algorithms, there is no addition of new virtual neighborhood links and therefore new virtual neighbors during the process to solve a static problem, which forbids increasing the link density and then harms the notion of locality.

The termination criteria is only a time limit. This is due to the fact that agents are unable to terminate by themselves because of their limited view, and because of the absence of hierarchy, contrary to ADOPT algorithm [9], but similarly to ERA approach [6].

Because this work only addresses the hard constraints of the FAPP, it is not possible to directly compare our algorithm to existing ones, in terms of performance. But, to validate our approach, several comparisons have been done by translating graph-coloring problems into FAPP, as ADOPT presents some results for these problems [9]. Results are not really good (up to 2 times less efficient), even if these two kinds of problems are CSP, because they are slightly different. First, instances for graph-coloring involve domains with only 3 values, when domains of FAPP contains to 100 times more values. Moreover, in graph-coloring, there are few variables/agents (some tens), when FAPP requires at least 200 agents. Finally, and this is the most important difference, graph-coloring is more constrained (even over constrained) than FAPP. There are numerous constraints per agent. Therefore, the order on criteria we used may be not adapted to this kind of problems.

8. Conclusion

In this paper, a multi-agent system, using self-organization mechanisms, has been developed to tackle a frequency assignment problem (FAPP). The local algorithm we present is based on cooperation between agents which elect and negotiate during assignment sessions to improve the global satisfaction, by reasoning on their difficulties. The main outcome is to produce adaptive systems able to respond to environmental changes. The systems find solutions for large FAPP, even if the performance are not high, compared to other algorithms. Two major perspectives are emphasized. First, improving the global behavior by studying other decision criteria and sorting them within the difficulty measure. Second, enhancing the system to take into account soft constraints such CEM for the FAPP.

Acknowledgments

We would like to thank Florian Cornet for his contribution to the development of the solver, and Gérard Verfaillie for his enlightened advices.

References

[1] D. Capera, J. Georgé, M.-P. Gleizes, and P. Glize. The AMAS theory for complex problem solving based on self-organizing cooperative agents. In *1st Int. TAPOCS Workshop at 12th IEEE WETICE*, pages 383–388. IEEE, 2003.

[2] T. Defaix. FAPP: Frequency Assignment with Polarization Problem - ROADEF Challenge 2001. Technical Report Revision 2, CELAR/TCOM, 2000.

[3] P. Galinier, M. Gendreau, and P. Soriano. Solving the Frequency Assignment Problem with Polarization by Local Search and Tabu. *4OR*, 3(1):59–78, 2005.

[4] K. H., P. Valckenaers, B. Saint-Germain, P. Verstraete, C. B. Zamfirescu, and H. Van Brussek. Emergent Forecasting Using Stigmergy Approach in Manufacturing Coordination and Control. In *Engineering Self-Organizing Applications (ESOA'04)*, pages 210–226. Springer, 2004.

[5] A. Hertz, D. Schindl, and N. Zufferey. Lower bounding and tabu search procedures for the frequency assignment problem with polarization constraints. *4OR*, 3(2):139–161, 2005.

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

[7] R. Mailler. Using Cooperative Mediation to Solve Distributed Constraint Satisfaction Problems. In *AAMAS'04*, pages 446–453. ACM, 2004.

[8] R. Mailler. Comparing two approaches to dynamic, distributed constraint satisfaction. *AAMAS'05*, 2005.

[9] P. J. Modi, W. Shen, M. Tambe, and M. Yokoo. ADOPT: Asynchronous Distributed Constraint Optimization with Quality Guarantees. *Artificial Intelligence*, 161(2):149–180, 2005.

[10] P. Peeters, P. Valckenaers, J. Syns, and S. Brueckner. Manufacturing Control Algorithm and Architecture. In *2nd International Workshop on Intelligent Manufacturing Systems*, pages 877–888, 1999.

[11] G. Picard, C. Bernon, and M.-P. Gleizes. ETTO : Emergent Timetabling by Cooperative Self-Organization. In *Engineering Self-Organizing Applications (ESOA'05)*, pages 31–45. Springer, 2005.

[12] G. Picard and P. Glize. Model and Analysis of Local Decision Based on Cooperative Self-Organization for Problem Solving. *Multiagent and Grid Systems (MAGS)*, 2(3):253–265, 2006.

[13] M. Vasquez. Arc-consistency and tabu search for the frequency assignment problem with polarization. In *CP-AI-OR'02*, pages 359–372, 2002.

[14] M. Yokoo, E. H. Durfee, T. Ishida, and K. Kuwabara. The Distributed Constraint Satisfaction Problem : Formalization and Algorithms. *IEEE Transactions on Knowledge and Data Engineering*, 10(5):673–685, 1998.