Affectation distribuée de fréquences par auto-organisation coopérative

Gauthier Picard Marie-Pierre Gleizes Pierre Glize
picard@irit.fr gleizes@irit.fr glize@irit.fr

IRIT - Université de Toulouse III
118, route de Narbonne
31062 Toulouse cedex 9 – FRANCE

Résumé

Cet article présente une approche pour résoudre des problèmes de satisfaction de contraintes distribués (DCSP) par des systèmes multi-agents auto-organisateurs. Les DCSP considèrent la distribution parmi des agents coopératifs dont la tâche est d'affecter une variable propre en respectant des contraintes connues et en négociant pour trouver une solution collective. L'approche proposée définit l'auto-organisation coopérative comme le processus guidant le collectif vers la solution : les agents, suivant un modèle comportemental coopératif, changent leur organisation pour améliorer l'état courant du système. Ce travail est illustré par un problème d'affectation distribuée de fréquences, un problème classique de modélisation sous contraintes.

Mots-clés : *Satisfaction de contraintes, problème d'affectation, auto-organisation, coopération*

Abstract

This paper presents an approach to solve DCSP by using self-organizing multi-agent systems. DCSP concerns distribution among agents which task is to assign values to variables with respect with known constraints. Agents only know their variables and the constraints on 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.

Keywords: *constraint satisfaction, assignment problem, self-organization, cooperation*

1 Introduction

Les systèmes multi-agents constituent un paradigme de conception pertinent pour des problèmes distribués et difficiles comme l'*autonomic computing*. Plongés dans des environnements dynamiques, de tels systèmes sont capables de répondre aux changements. Plus particulièrement, l'approche par systèmes multi-agents adaptatifs (AMAS) considère que la décision du changement de la fonction du système est prise par les parties du système, les agents, qui sont ainsi capables de changer leur organisation pour modifier le comportement du système dans son ensemble [4]. Les agents suivent un modèle comportemental guidant la réorganisation par *coopération*. Lorsqu'un agent n'est plus dans un état coopératif (conflit, inutilité, concurrence, etc.), il modifie l'organisation du système en changeant sa position dans l'environnement social ou physique. Cette approche par *auto-organisation coopérative* a déjà été appliquée à des problèmes difficiles comme l'emploi du temps distribué [9] ou le contrôle de production [1].

Cet article présente une solution pour les CSP distribués utilisant l'auto-organisation coopérative. Ce travail est proche des célèbres algorithmes ABT, AWCS ou ADOPT [12, 8], mais ne nécessite pas d'ordre sur les agents, et est comparable à des algorithmes de recherche locale comme la recherche tabou. L'approche AMAS est ici appliquée à un problème d'affectation de fréquence avec polarisation (FAPP), qui est une version distribuée du problème classique d'affectation de fréquences [2]. Dans un premier temps, les approches multi-agents existantes pour résoudre des problèmes de satisfaction de contraintes distribués sont présentées. Dans la section 3, le FAPP est formellement décrit. L'approche et l'algorithme proposés sont présentés dans les sections 4 et 5 et des résultats sont fournis dans la section 6 puis discutés dans la section 7. Enfin, la section 8 conclut l'article et dresse quelques perspectives pour ce travail.

2 Résolution multi-agent de CSP

De nombreux problèmes de décision peuvent être formalisés comme des problèmes de satisfaction de contraintes (CSP), représentés comme un ensemble de variables (les objectifs) qui peuvent prendre leur valeur dans des domaines donnés en respectant certaines contraintes. Un CSP est un triplet $\langle X, D, C\rangle$ tel que : $X = \{x_1, \ldots, x_n\}$ est un ensemble fini de variables à affecter, $D = \{D_1, \ldots, D_n\}$ est un ensemble fini de domaines pour chaque variable, $C = \{c_1, \ldots, c_m\}$ est un ensemble fini de contraintes sur les valeurs des variables. Les domaines D_i peuvent être finis ou infinis, discrets ou continus, bien que la grande majorités des approches multi-agents considèrent des domaines finis et discrets (problèmes combinatoires). De plus, les contraintes de C peuvent être représentées comme des n-uplets de valeurs interdites ou comme des prédicats sur plusieurs variables.

Un CSP distribué (DCSP) est un quintuplet $\langle A, X, D, C, f\rangle$, tel que : $\langle X, D, C\rangle$ est un CSP, A est un ensemble fini d'agents, $f : A \times X \mapsto \{true, false\}$ est un prédicat assignant les variables aux agents. Résoudre un DCSP nécessite de pourvoir les agents de modèles comportementaux incluant des capacités de négociation pour trouver une solution commune. En effet, les agents ne connaissent pas le CSP dans son entier et doivent donc interagir pour récupérer l'information utile à la résolution des conflits induits par les contraintes. Le cadre des DCSP a été introduit par Yokoo *et al.* qui proposaient des algorithmes asynchrones inspirés des algorithmes de résolution classiques [12]. ABT (*Asynchronous Back-Tracking*) est une version asynchrone d'un *backtrack* avec un *ordre prédéfini* sur les agents pour assurer la consistance et l'échange des *nogoods*. AWC (*Asynchronous Weak Commitment*) améliore ABT grâce à l'heuristique *min-conflict* et l'ordonnancement dynamique des agents, en fonction des *nogoods* des agents et de leur voisinage. AWC est plus efficace qu'ABT pour trouver une solution, mais ABT est plus rapide pour montrer qu'une solution n'est pas satisfiable. Ces deux algorithmes sont complets, alors que d'autres travaux, basés sur une vue plus locale des agents comme DynDBA (*Dynamic Distributed Breakout Algorithm*), ne le sont pas [7]. Certains problèmes n'ont pas de solution, mais, un problème devient un problème d'optimisation, en fournissant une fonction de coût sur les contraintes. Des approches inspirées des algorithmes ci-dessus ont été développées pour cela, comme ADOPT [8] ou ERA [6], dans lequel les agents évoluent dans un environnement physique, leur domaine de valeur, et suivent des comportements stochastiques différents pour converger vers la solution. Dans cet article, seule la partie satisfaction du FAPP est présentée, mais nous garderons à l'esprit l'idée d'ERA.

3 Le FAPP

3.1 Présentation

Il s'agit d'allouer des fréquences dans les réseaux de télécommunications par voie hertzienne. Un réseau est composé d'un ensemble de sites comprenant des émetteurs ou/et des récepteurs [2]. Une liaison hertzienne établit un lien entre deux sites géographiques. Elle peut être constituée de un ou plusieurs trajets. Un *trajet* est un bond radioélectrique *unidirectionnel*, établi entre des antennes sur des sites géographiques distincts. Une *ressource fréquentielle* est définie comme un couple *(fréquence,polarisation)*.

Soit T, l'ensemble des trajets du problème. Soient $F_0, F_1, ..., F_n$, n domaines fréquentiels associés au problème. Soient $P_{-1} = \{-1\}, P_1 = \{1\}, P_0 = \{-1, 1\}$, les trois domaines de polarisation possibles. À chaque $i \in T$, on associe $f_i \in F_{\varphi(i)}$ et $p_i \in P_{\pi(i)}$ où $\varphi(i)$ et $\pi(i)$ indiquent respectivement le numéro de domaine fréquentiel et de polarisation associés au trajet i.

Soient les contraintes suivantes :

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

où γ_{ij} et δ_{ij} sont des données du problème indiquant l'écart entre les fréquences nécessaires selon la polarisation.

Les contraintes existantes pour le réseau sont de deux types. Les contraintes électromagnétiques, ou *CEM*, liées à la proximité (transmetteur et récepteur sur un même site) ou au contraire à l'éloignement géographique. La différence de polarisation est un moyen technique de diminuer les problèmes d'interférences entre

les fréquences induits par ces contraintes. Les contraintes technologiques impératives, ou *CI*, forment le deuxième groupe de contraintes qui imposent une distance entre les fréquences de deux trajets formant, par exemple, une liaison duplex entre deux sites.

Une *solution réalisable* correspond à l'allocation d'une fréquence et d'une polarisation à chaque trajet en satisfaisant l'ensemble des contraintes posées. Le problème étant souvent sur-contraint, les contraintes CEM peuvent être relâchées sur 11 niveaux de repli. L'objectif est alors double : *(i)* satisfaire les contraintes $C2$ à $C7$ et *(ii)* minimiser une fonction de coût sur les niveaux de repli, que nous ne détaillerons pas dans cet article.

À mi-chemin entre les problèmes réels et les cas d'école (comme la coloration de graphe ou les *n* reines), l'étude du FAPP permet de faire un pas supplémentaire vers la résolution de problèmes de contraintes réels. Par l'hétérogénéité de ses contraintes et la taille importante des problèmes (jusqu'à 3000 trajets à allouer), le FAPP est un challenge intéressant. Il conserve cependant deux caractéristiques facilitant son étude : des domaines finis et discrets de fréquences et de polarisation à allouer et des contraintes binaires. L'originalité du FAPP réside aussi dans ce mélange de contraintes impératives et de contraintes souples. Ceci implique de concevoir un algorithme bivalent, devant traiter *obligatoirement* un ensemble de contraintes dures et pouvant ensuite optimiser la situation en se concentrant sur les contraintes souples.

3.2 Distribution du problème

Il existe plusieurs solutions efficaces pour résoudre le FAPP : recherche locale et tabou [3], arc-consistance et recherche tabou [11] ou borne inférieure pour k et recherche tabou [5]. Toutes ces approches sont basées sur la recherche tabou, mais ne sont ni distribuées ni adaptatives. Ces algorithmes sont centralisés et gèrent tout les réseau. En cas de changement pendant la résolution (apparition/disparition de site), tout le processus doit être relancé, et ne produira pas forcément de solution proche de la précédente, impliquant un minimum de changement pour chacun des sites.

Plusieurs contraintes s'imposent naturellement quand on utilise le paradigme agent en gardant à l'esprit la résolution de problèmes *dynamiques* et *distribués*. Tout d'abord l'absence de contrôle global rendant impossible la *séparation en phases* comme c'est le cas dans les trois algorithmes présentés. Ensuite, la volonté de garder un système pouvant être dynamique empêche d'appliquer l'arc-cohérence en pré-calcul. Plusieurs concepts peuvent par contre, être repris au niveau agent :

– l'idée de *pénalité* sur les valeurs pour inciter à explorer le domaine d'un trajet ;
– la notion de *criticité* d'une variable ou d'une valeur pour favoriser, au contraire, certaines valeurs ou rendre des agents plus prioritaires ;
– le maintien des paires de trajets liés par une contrainte d'écart $C2$ ou $C4$ (cf. 3.1) qui pourrait être réalisé par un *dialogue* entre agents ;
– Le tabou en général et l'interdiction à certains trajets de se modifier en particulier pourraient être transposés au niveau d'un agent.

Outre le fait que ces approches tabous soient les plus performantes pour traiter ce problème, cette approche peut être réutilisée au niveau de l'agent. L'approche multi-agent présentée apporte un point de vue radicalement nouveau et différent des autres approches tout en constituant un bon point de départ pour une étude plus approfondie du FAPP et des DCSP en général, par le biais des SMA.

4 Modélisation multi-agent

4.1 Spécification du problème

Pour décrire le système multi-agent, voici quelques caractéristiques du FAPP :

– T l'ensemble des trajets ;
– $\forall t \in T,\ F_t$ le domaine fréquentiel associé au trajet t ;
– $\forall t \in T,\ P_t$ le domaine de polarisation associé au trajet t ;
– C l'ensemble des contraintes ;
– CI l'ensemble des contraintes impératives ;
– CEM l'ensemble des contraintes de compatibilité radio-électromagnétiques ;
– $\forall t \in T,\ C_t = CI_t \cup CEM_t$ l'ensemble des contraintes d'un trajet t ;
– $\forall c \in C,\ t_1(c)$ et $t_2(c)$ les deux trajets reliés par la contrainte ;
– $\forall t, u \in T,\ v(t,u) \equiv \exists c \in (C_t \cap C_u)$ la relation de voisinage entre deux trajets ;
– $\forall t \in T,\ V_t = \{u \in T \mid v(t,u) = true\}$ l'ensemble des voisins d'un trajet appelé voisinage.

Le système à modéliser va comporter un ensemble d'*agents*, autonomes, homogènes tant dans leur comportement que dans leur représentation. À chaque trajet (ou variable), on associe un agent distinct des autres. Soit A l'ensemble

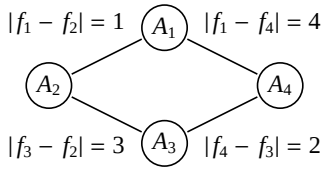


FIG. 1 – Un exemple simple avec 4 agents et 4 contraintes

des agents du système multi-agent. Pour tout agent $x \in A$, on note $t_x \in T$ le trajet associé à l'agent x. On a $f(x, t_x) = true$, en utilisant la notation de la section 2. Un agent x est *voisin* d'un agent y si et seulement si leurs trajets sont voisins, i.e. ils partagent une contrainte. Le voisinage d'un agent est défini comme l'ensemble des voisins de cet agent.

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

L'agent possède le même domaine fréquentiel et le même domaine de polarisation que la variable qu'il représente :

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

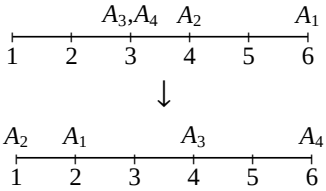
Les CI et CEM sont connues de chaque agent dont la variable dépend. Soit l'ensemble C_x représentant l'ensemble des contraintes connues de l'agent :

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

Une contrainte $c \in CI_x$ satisfaite s'écrit $c = true$. Comme cet article ne présente que la partie satisfaction du problème, les CEM ne sont pas présentées ici.

Dans la fig. 1, $A = \{A_1, A_2, A_3, A_4\}$. Les agents partagent les mêmes domaines $D_f = \{1, 2, 3, 4, 5, 6\}$ et $D_p = \{1\}$. Le problème est simplifié en mettant de côté la polarisation. Seuls deux types de CI sont représentés : $C2$ et $C4$, des contraintes d'écart. Par exemple, une contrainte lie A_1 et A_2 : $|f_1 - f_2| = 1$. Il est aisé de trouver des solutions pour ce FAPP, qui sont, par exemple, pour $(f_{A_1}, f_{A_2}, f_{A_3}, f_{A_4})$: $(2, 1, 4, 6)$ et $(5, 6, 3, 1)$.

Par conséquent, le problème d'affectation peut être vu comme un problème de positionnement dans lequel les agents représentent les fréquences, comme dans ERA [6]. Au début du processus, les agents sont positionnés aléatoirement, comme dans l'illustration suivante, par exemple :



Pour passer de l'état initial à la solution ci-dessus, nous proposons d'utiliser l'auto-organisation et la négociation entre agents. Chaque agent doit trouver une position correcte (une valeur consistante) dans son domaine, en suivant un processus auto-organisateur, guidé par le comportement coopératif des agents. Chaque agent vise à trouver la bonne place dans l'organisation en utilisant des interactions locales avec ses voisins (les agents liés par des contraintes directes) et en décidant par lui-même de se déplacer. Pour mettre ceci en œuvre et pour permettre à un agent de décider, des critères locaux de *coopération* sont utilisés.

4.2 Caractéristiques des agents coopératifs

Deux données principales caractérisent les agents : leur *valeur* et leur *difficulté*. La valeur d'un agent x est la paire $(f, p)_x$ avec $f \in F_x$ et $p \in P_x$. La difficulté d_x d'un agent x est une mesure quantitative prenant en compte l'environnement courant et la valeur courante de l'agent. Plus cette difficulté est élevée, plus l'agent se juge éloigné d'une solution à ses problèmes. Cette mesure est à la base des mécanismes de coopération. La difficulté d'un agent lui est propre (cf. section 5).

De plus, l'agent possède une vue partielle de l'environnement. Il ne connaît de ses voisins que leur *valeur* et leur *difficulté*, mais n'a aucune idée ni des contraintes de ses voisins, ni de leur vue, ni de leurs domaines. La *communication* entre les agents est réalisée par envoi de messages. Chaque agent évolue dans un environnement physique qui lui est propre. Le domaine fréquentiel et le domaine de polarisation, même s'ils peuvent être identiques entre plusieurs agents, ne sont pas pour autant partagés. De la même manière, chaque agent possède une copie non partagée des contraintes qui le concernent. La cohérence entre la vue de l'agent et ses contraintes locales est permanente. Tout changement de vue entraîne une mise à jour immédiate de l'état des contraintes. La cohérence au niveau global n'est par contre pas garantie tant qu'il existe des messages en circulation. En effet, la vue d'un voisin sur un agent pouvant différer de l'état actuel de l'agent, les contraintes entre les deux agents peuvent ne pas

être dans le même état. Pour assurer la cohérence globale quand le système est arrêté, il faut poser les hypothèses suivantes [12] : les messages sont transmis en un temps fini au destinataire, les messages ne sont pas perdus ou altérés lors de la transmission et un agent traite toujours les messages qu'il reçoit. Pour éviter toute ambiguïté en rapport avec les vues des agents, on introduit les notations suivantes :
– $c^x \in C_x$ la contrainte c vue par l'agent x ;
– d_x^y la difficulté de l'agent x pour l'agent y ;
– $(f, p)_x^y$ la valeur actuelle de x pour l'agent y.

4.3 Comportement des agents coopératifs

Le comportement de l'agent se déroule en 4 étapes (cf. fig. 2). Cet algorithme est commun à tous les agents qui l'exécutent en parallèle. Une fois le système démarré, les agents vont réaliser ce cycle sans qu'ils puissent se terminer d'eux-mêmes.

L'*initialisation* consiste à choisir (aléatoirement, par exemple) une valeur dans le domaine (ligne 1) et informer le voisinage que l'agent est en cours d'exécution (ligne 2). La communication étant réalisée par messages, chaque agent possède une boîte aux lettres qu'il va consulter régulièrement. La *consultation* se fait par ordre d'arrivée des messages et est toujours suivie d'une étape de traitement du message. Cette phase dure tant que la boîte aux lettres n'est pas vide (ligne 4). La *décision* permet de choisir le ou les agents qui vont agir (ligne 11). Elle n'a lieu que si l'agent connaît la difficulté des agents de son voisinage (ligne 10). Un agent ne va pouvoir calculer sa difficulté (ligne 6) que s'il connaît la valeur des agents de son voisinage (ligne 5). Ces conditions ne sont pas bloquantes dans la mesure où les agents communiquent leurs informations dès qu'ils les ont (cf. 4.4). De son point de vue, si un agent est le plus en difficulté, pour un voisinage donné, il est *élu* comme l'agent ayant à agir pour le voisinage. Il peut donc y avoir plusieurs agents élus dans tout le système. À difficulté égale entre plusieurs agents du voisinage, l'un des agents est élu de manière aléatoire. Pour $x \in A$, on note $e(x)$ le fait que x soit éligible :

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

C'est donc l'agent le plus en difficulté qui va tenter d'améliorer la situation en étant élu. En effet, un voisin de l'agent le plus en difficulté, l'aide au mieux en lui laissant le choix de l'affectation. De plus, un voisin n'a qu'un nombre

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

FIG. 2 – Comportement d'un agent en 4 étapes

limité de contraintes en commun avec l'agent et n'est donc pas mieux placé que lui pour l'aider, ce dernier ayant connaissance de toutes ses contraintes. Cette phase marque un premier aspect de la l'auto-organisation coopérative : *les agents laissent agir les voisins plus en difficulté qu'eux-même*.

L'*affectation* est réservée aux agents élus lors de la phase de décision (ligne 12). L'élu va sélectionner une valeur qu'il considère comme la meilleure pour lui et ses voisins parmi son domaine de valeurs privé de sa valeur courante. En aucun cas un élu ne peut inciter ses voisins ou obliger ses voisins à modifier leur valeur. Seule la valeur de l'élu est modifiée au cours de cette phase. S'il existe plusieurs valeurs que l'agent juge équivalentes, il en choisit une de manière aléatoire. À la fin de l'affectation, l'agent se désactive : il signale à son voisinage qu'il ne participera pas aux prochaines élections tant qu'un de ses voisins n'aura pas été élu. Cette politique *égalitariste* pour l'élection permet aux voisins d'avoir la possibilité d'être élu. Bien que désactivé, si un de ses voisins est élu, l'agent est tout de même *invité* à la session d'affectation (cf. 4.4). L'état (activé/déactivé) d'un agent est connu de ses voisins et peut être interprété comme un *tabou local*.

4.4 Le protocole de communication

Un agent ne communique qu'avec ses voisins en utilisant des messages directs contenant la *vue* (i.e. les valeurs affichées) de l'émetteur, la vue

du destinataire et l'identifiant de la session d'affectation si nécessaire (voir plus bas). De plus, un agent informe systématiquement ses voisins s'il change de valeur ou de difficulté, afin de ne pas perturber les vues et les décisions des voisins. Il convient donc d'étudier les situations où il est utile de communiquer sans pour autant surcharger le système :

- après l'*initialisation* (ligne 2) ;
- pendant la *consultation*, à chaque réception de message :
  - soit la valeur de ce voisin a été modifiée, il faut alors réévaluer la difficulté de l'agent ;
  - soit ce voisin s'est désactivé, il faut alors changer l'état de l'agent ;
  - soit la difficulté de ce voisin a été modifiée ou il est de nouveau actif ;
- à la fin de la *consultation*, l'agent envoie un message à son voisinage uniquement s'il a connu un changement de difficulté lié à un changement de valeur d'un voisin ou s'il a changé d'état (ligne 7-8) ;
- à la fin de l'*affectation*, l'agent se désactive et signale à ses voisins son changement de valeur, d'état et éventuellement de difficulté.

Pour éviter les *deadlocks*, il est nécessaire d'introduire un point de synchronisation lors de l'affectation, mis en œuvre au travers de *sessions d'affectation*. Pour un voisinage donné, l'agent élu crée tout d'abord une session d'affectation, qui est un objet synchronisé (moniteur, par exemple). Ensuite, il envoie des invitations (comprenant ses derniers état, valeur et difficulté) à ses voisins. Ces derniers peuvent accepter ou rejeter l'invitation s'ils sont déjà dans une session d'affectation pour laquelle l'élu est plus en difficulté ou s'ils ont eux-même une difficulté plus importante. Une fois toutes les réponses reçues, l'élu procède à l'affectation. Puis, il informe ses invités que la session s'est correctement déroulée en envoyant sa nouvelle valeur. Tant que la session n'est pas terminée, les *agents invités* ne traitent pas les messages provenant de l'élu.

5 Critères coopératifs pour les contraintes impératives

5.1 Mesure de difficulté

La *difficulté* est le critère de décision pour élire un agent. Elle est représentée sous la forme d'un vecteur *ordonné* de sous-critères. La difficulté d'un agent $x \in A$ vue par un agent $y \in A$ s'écrit :

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

L'*amélioration* (ou *Im*) d'un agent est la meilleure amélioration possible pour cet agent par rapport à sa situation courante, au regard du nombre de contraintes impératives insatisfaites. Pour tout $x \in A$, on a, pour tout $(f, p) \in F_x \times P_x$, $NS_x(f, p)$ le nombre de contraintes impératives insatisfaites avec la valeur (f, p). En posant $NS_x \equiv NS_x(f, p)_x$, on note :

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

Le coût de calcul de ce critère est $|F_x \times P_x - 1|$ car la valeur courante n'est pas incluse. L'idée derrière ce critère est de dire que des agents qui savent qu'ils peuvent améliorer strictement leur situation locale ont conscience qu'ils sont effectivement les moins bien placés actuellement. Cette nécessité de se projeter en avant, bien que coûteuse, améliore significativement la recherche.

Contrairement à *Im*, les critères suivants ne sont pas basés sur les domaines, mais sur les contraintes. Pour tout $x \in A$, pour toute $c^x \in CI_x$, on note $FPS(c^x)$ l'ensemble des couples $(f, p) \in F_x \times P_x$ vérifiant c^x par rapport à la vue courante de x.

Le deuxième critère, les *possibilités* (ou *Po*), traduit le fait qu'une contrainte est difficile s'il existe peu d'affectations possibles pour l'agent, capables de vérifier cette contrainte :

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

et en cas d'égalité du critère précédent (*Im*) :

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

Le troisième critère est le *nombre de contraintes impératives insatisfaites* dans la situation courante (ou NS_x). Ce critère considère qu'un agent ayant plus de contraintes insatisfaites est plus en difficulté. En cas d'égalité des critères précédents (*Im* et *Po*) :

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

Enfin, le dernier critère pour mesurer la difficulté est l'*âge* (ou *Ol*). Pour tout $x \in A$ et pour toute $c^x \in CI_x$, on note $TI_x(c^x)$ le nombre d'affectations réalisées par x ou son voisinage depuis la dernière *satisfaction* de c^x. Soit :

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

L'agent ayant la plus vieille contrainte est le plus en difficulté, en cas d'égalité des critères précédents :

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

5.2 Autres critères de décision

Outre la notion de difficulté, il existe deux autres critères pour élire les agents. Tout d'abord, un agent essaiera d'être élu si et seulement s'il possède des contraintes non satisfaites, pour ne pas élire des agents qui n'apporteront aucun changement :

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

Si deux agents ont la même difficulté après évaluation des critères, un agent est choisi aléatoirement (Eq). Enfin, le dernier critère considère qu'un agent étant le seul actif d'un voisinage donné est automatiquement élu (De).

5.3 Choix des valeurs

La mesure de difficulté d'un agent a servi d'argument à son élection. En, conséquence, il semble judicieux de choisir la valeur d'affectation parmi celles qui ont servi à cette élection. L'idée est donc de sélectionner la valeur à affecter en fonction du critère qui a été décisif pour son élection. Ce critère, appelé *critère discriminant* l'agent x de l'agent y est noté cd_y^x. Le *critère de sélection* de l'action est noté cs^x :

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

Une manière différente de choisir une valeur ou une contrainte à résoudre est associée à chaque critère de sélection :
- $cs^x = Im$: sélectionner la valeur qui maximise l'amélioration ;
- $cs^x = Po$: sélectionner l'ensemble des contraintes ayant le minimum de possibilités ;
- $cs^x = NS$: sélectionner l'ensemble des contraintes avec le maximum de possibilités ;
- $cs^x = Ol$: sélectionner les contraintes les plus âgées ;
- $cs^x = Eq$: sélectionner les contraintes partagées avec des agents ayant la même difficulté que l'agent ;
- $cs^x = De$: sélectionner l'ensemble de toutes les contraintes de l'agent.

Soit VEl_x l'ensemble des valeurs choisie par x.

Cependant, il y a une contrepartie à faire ce choix en utilisant les ensembles de contraintes : toutes les contraintes peuvent être insatisfiables et par conséquent toutes les valeurs sont possibles, exceptée la valeur courante. Ceci implique de doter les agents de capacités de dialogue entre élu et invités.

5.4 Dialogue au cours d'une session

Pour un voisinage donné, une affectation est utile dans deux situations :
- l'affectation améliore *immédiatement* la situation du voisinage ;
- l'affectation améliore *potentiellement* la situation d'un voisin, en lui permettant, par exemple, d'affecter une valeur qui diminue le nombre de contraintes insatisfaites, si le voisin est élu dans le futur.

Soit $S_x(f, p)$ le nombre de contraintes impératives satisfaites de l'agent x avec la valeur (f, p). Les données échangées pour représenter l'utilité d'une valeur (f_x, p_x) d'un agent élu x du point de vue d'un voisin y est :

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

Le coût de ce calcul est initialement élevé : pour chaque valeur de l'élu, une matrice du nombre de contraintes satisfaites doit être calculée. Mais une fois calculée la première fois, elle ne nécessite plus que des mises à jour. Avec cette utilité pour chaque valeur et chaque voisin, l'élu x sélectionne les valeurs qui maximisent la somme des utilités de ses voisins :

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

S'il existe encore des valeurs équivalentes, x procède au même filtrage en sélectionnant uniquement les valeurs parmi VNe_x. Ceci représente un équilibre entre altruisme et égoïsme qui définit la notion de coopération, comme présenté dans [10]. Enfin, S'il existe encore des valeurs équivalentes après cette sélection, une valeur est choisie aléatoirement.

5.5 Exemple

La table 1 présente un exemple de déroulement pour le FAPP présenté à la fig. 1. L'état initial est ici $(f_{A_1}, f_{A_2}, f_{A_3}, f_{A_4}) = (3, 3, 3, 3)$. Trois

	Étape 1		Étape 2		Étape 3		Fin
A1	***3*** [1,0,2,1]	1,2,4,***5***,6	5 [1,0,2,2]	0	5 [0,0,1,3]	1 (5)	5
A1-A2	[2,1] (2,4)		[2,2] (2,4)		[1,0] (5)		
A1-A4	[0,1]		[0,2]		[0,3]		
A2	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
A2-A1	[2,1] (2,4)		[2,2] (4,6)		[2,0] (4,6)		
A2-A3	[1,1] (6)		[1,2] (6)		[1,0] (6)		
A3	3 [1,1,2,1]	n/a	3 [1,1,2,1]	0	3 [0,1,1,2]	1 (3)	3
A3-A2	[1,1] (6)		[1,1] (6)		[1,0] (3)		
A3-A4	[2,1] (1,5)		[2,1] (1,5)		[2,2] (1,5)		
A4	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
A4-A1	[0,1]		[1,2] (1)		[1,2] (1)		
A4-A3	[2,1] (1,5)		[2,2] (1,5)		[2,2] (1,5)		

TAB. 1 – Trace pour le FAPP de la fig. 1

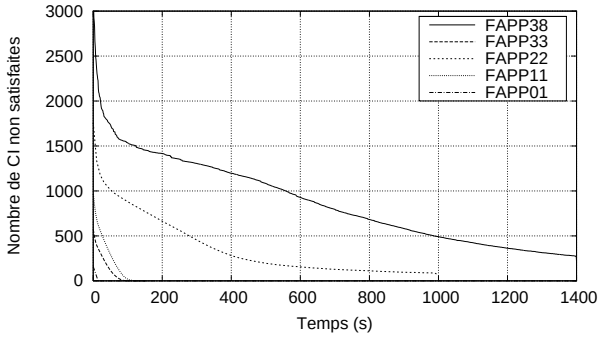


FIG. 3 – Convergence vers des solutions

étapes sont nécessaires pour atteindre une solution. Comme tous les agents sont dans le même voisinage, il n'y a pas de résolution en parallèle, contrairement à des problèmes plus grands. Pour chaque étape, les données sont présentées comme suit. Une *partie décision* (colonne à gauche de chaque pas) pour chaque agent est composée de la fréquence suivie par le vecteur de difficulté, et, pour chaque contrainte (notée $A_x - A_y$), un vecteur *[possibilité,âge]* suivi par les fréquences qui peuvent satisfaire la contrainte. Par exemple, initialement, la fréquence de A_1 est 3, sa difficulté est $[1, 0, 2, 1]$, et la contrainte $A_1 - A_2$ du point de vue de A_1 a deux possibilités $(2, 4)$ et son âge est 1. Une *partie invitation* (colonne à droite de chaque étape) pour l'agent élu montre les fréquences qu'il croit possibles. Ses voisins envoient, pour chacune de ces fréquences, la somme des améliorations possibles et l'amélioration immédiate, avec la fréquence qui conduit à cette amélioration (si possible). Par exemple, à l'étape 1, A_1 est élu (fréquence en italique) et présente l'ensemble des fréquences possibles à ses voisins : $VEl_{A_1} = \{1, 2, 4, 5, 6\}$. A_2 envoie $\{0, 0, 0, 1(6), 0\}$ à A_1 afin de l'informer que si A_1 choisit la fréquence 5 alors A_2 à une amélioration possible de 1 en choisissant la fréquence 6. À l'étape 2, A_1 est désactivé (fond grisé) car il vient juste d'assigner une nouvelle valeur.

À l'étape 1, deux agents sont équivalents : A_1 et A_4. A_1 est choisi aléatoirement et son critère d'élection est alors $cs^{A_1} = Eq$. A_1 examine la contrainte qu'il partage avec A_4 : $A_1 - A_4$. Comme il n'y a aucune possibilité pour satisfaire la contrainte, A_1 n'est pas capable de choisir une bonne valeur, et il sélectionne tout le domaine à l'exception de sa valeur courante : $VEl_{A_1} = \{1, 2, 4, 5, 6\}$. La réponse de son voisinage est que le meilleur choix est 5. Ceci ne laisse qu'une seule possibilité d'affectation pour A_2 en 6 et pour A_4 en 1.

À l'étape 2, A_1 est désactivé et A_2 et A_4 sont élus grâce au critère *Im*. Comme les agents ne peuvent être invités qu'à une seule session en même temps, A_1, par exemple, rejette l'invitation de A_4, qui attendra jusqu'à ce que la session d'affectation de A_2 se termine pour envoyer une nouvelle invitation. A_2 effectue l'affectation, en fonction du critère *Im*, et examine les valeurs qui impliquent la plus grande amélioration, i.e. 6 qui satisfait les deux contraintes de A_2. Le dialogue avec les voisins est inutile car il n'y a qu'une seule valeur possible : 6.

À l'étape 3, A_4 est élu malgré les changements de difficulté de ses voisins A_1 et A_3, et réalise le même raisonnement que A_2 en choisissant pour sa part la fréquence 1 qui vérifie les deux contraintes d'A_4. Une fois l'affectation effectuée, les agents sont tous résolus : la solution est trouvée.

6 Résultats et analyse

Les différentes instances référencées dans cet article proviennent du challenge ROADEF 2001[1], et ont été lancées plusieurs fois sur une seule machine, avec les limites de temps de la table 2.

6.1 Convergence

La convergence vers des solutions est présentée dans la fig. 3. La diminution importante en début de résolution est due au fait que certaines contraintes sont satisfaites dès l'initialisation aléatoire. Ces contraintes sont souvent des contraintes $C3$ ou $C5$, plus faciles à satisfaire.

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

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

TAB. 2 – Caractéristiques des instances FAPP

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

TAB. 3 – Pourcentage de CI insatisfaites

Instance	01	11	22	33	38
Affectations	107,24	547,02	964,81	338,35	1146,18
Affect./CI	0,64	0,56	0,54	0,59	0,37
Affect./Temps(s)	17,87	3,42	0,96	2,26	0,82
σ	5,33%	6,89%	5,36%	2,28%	3,88%

TAB. 4 – Étude des affectations

Instance	01	11	22	33	38
Agents	200	1000	1750	650	2500
Messages	2335	15841	33247	8348	48727
Messages/Agent	11,7	15,8	19	12,8	19,5
% annulation	4,21	6,02	13,56	5,22	10,28

TAB. 5 – Trafic de messages

La diminution apparaissant une fois le temps limite atteint correspond à la terminaison des agents qui traitent les messages restant avant de terminer, sans pour autant initier de nouvelles sessions d'affectation : ils ont atteint une situation stable. Mais les résultats de l'instance 22 montre une stagnation : un minimum local est atteint, et il reste, en moyenne, $3,49\%$ de CI insatisfaites, comme le montre la table 3. Un autre problème est le temps de résolution d'un problème non complet (sans CEM), qui dépend du nombre d'agents. Ainsi, même sur de petites instances, l'approche ne permet pas de trouver des solutions exactes systématiquement.

6.2 Affectation et trafic de messages

La table 4 montre qu'il y a peu d'affectations. En considérant la taille de l'espace de recherche, l'exploration est réduite. En effet, l'idée derrière la notion de coopération est de faire de bons choix en amont pour assurer une convergence rapide, au lieu d'explorer exhaustivement l'espace de recherche : la coopération est une *heuristique*. Le faible rapport entre le nombre d'affectations et le nombre de CI confirme que le processus de résolution est fortement guidé. La déviation standard σ est faible également, de l'ordre de 5%, ce qui souligne la capacité

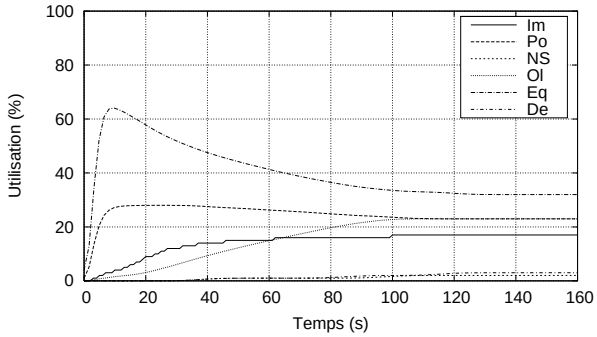


FIG. 4 – Distribution des critères pour FAPP11

d'adaptation de l'algorithme, indépendamment de l'état initial, et par conséquent l'adaptation à des changements en cours de résolution. Toutefois, la vitesse d'affectation est faible, et ce rapport diminue avec un grand nombre d'agents. Mais, à chaque instant, il y a plus d'une session d'affectation, et comme le comportement est complètement parallélisable, le temps sur des environnements distribués est bien meilleur, même en prenant en compte le trafic de messages, présenté dans la fig. 5. De plus, il y a peu de situations conflictuelles (annulation de sessions).

6.3 Pertinence des critères de décision

Parmi les critères présentés dans la section 5, la fig. 4 montre lesquels sont utilisés par les agents lors de la résolution. La valeur pour *De* (lorsque les voisins sont déjà élus) est très faible. Mais la critère *NS* (nombre de CI insatisfaites) n'est pas souvent déclenché, contrairement à *Ol*, *Im* et *Po*. Enfin, 30% des agents en moyenne évaluent leurs voisins comme équivalents (*Eq*).

7 Discussion

Comparaison aux approches existantes. Le critère de terminaison est uniquement une limite de temps car les agents sont incapables de terminer par eux-mêmes, à cause de leur vue limitée et de l'absence de hiérarchie, contrairement à ADOPT [8], mais comme dans ERA [6]. Comme ce travail n'attaque que la partie impérative du FAPP, il est difficile de le comparer aux approches le résolvant, dans son intégralité. Mais, plusieurs comparaisons ont été faites en transformant le FAPP en problème de coloration de graphe, car ADOPT présentait des résultats sur ce problème. Les résultats ne sont pas convaincants (2 fois moins efficace) en partie à cause de la différence de topologie de ces deux

CSP. Les instances de la coloration de graphe concernent des domaines à 3 valeurs, quand les domaines du FAPP en contiennent 100 fois plus. De plus, dans la coloration de graphe, il y a peu de variables (quelques dizaines), alors que le FAPP nécessite au moins 200 agents. Enfin, la coloration de graphe implique de nombreuses contraintes par agents et donc les notions de voisinage et de session d'affectation sont remises en cause.

Évaluation de la difficulté. Une première observation est que les critères sont génériques : ils ne sont pas spécifiques au FAPP, même si *Po* est proche des notions du FAPP. De plus, le concept de paire de trajets [3] pour gérer des cas particuliers de trajets partageant des contraintes $C2$ ou $C4$, n'apparaît pas dans la décision, même si elle impacte sur *Po*. Enfin, le tri des critères reste une question ouverte, dont la validité est difficile à observer. Même s'il semble pertinent de positionner *Im* en tête (cf. 6.3), les autres critères peuvent être discutables et devront être approfondis.

Contraintes souples et optimisation. S'attaquer aux contraintes CEM est une des perspectives directes de ce travail. Ces contraintes sont très nombreuses et augmentent la taille des voisinages, mettant sur la sellette le protocole impliquant un grand nombres d'invités dans les sessions. Une solution sera éventuellement de considérer un voisinage par type de contrainte (CEM et CI). Ensuite, un autre problème sera d'inclure les CEM dans la mesure de difficulté. S'il semble pertinent de les positionner après les critères sur les CI pour être plus décisif, ils ne devront pas corrompre les solutions pour les CI.

8 Conclusion

Nous avons présenté un SMA utilisant des mécanismes auto-organisateurs pour résoudre un problème d'affectation de fréquences. L'algorithme distribué proposé est guidé par le comportement coopératif des agents qui élisent et négocient lors de sessions d'affectations pour améliorer la satisfaction globale, en raisonnant sur leurs difficultés. L'apport principal est de proposer des systèmes adaptatifs capables de répondre aux changements. Le système trouve des solutions pour de large instances, même si les performances ne sont pas élevées, comparé aux solutions centralisées. Deux principales perspectives sont envisagées : étudier d'autres modèles de comportement coopératifs et critères de décision, et étendre le système pour résoudre des problèmes d'optimisation avec contraintes souples, comme les CEM du FAPP.

Remerciements

Nous remercions Florian Cornet pour sa contribution au développement du solveur, ainsi que Gérard Verfaillie pour ses conseils avisés.

Références

[1] D. Capera, C. Bernon, and P. Glize. Étude d'un processus d'allocation coopératif de ressources entre agents pour la gestion de production . In *7e Congrès de la Société Française de Recherche Opérationnelle et d'Aide à la Décision (ROADEF'06)*, pages 369–383. PUV, 2006.

[2] T. Defaix. FAPP : Frequency Assignment with Polarization Problem. 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] J.-P. Georgé, M.-P. Gleizes, and P. Glize. Conception de systèmes adaptatifs à fonctionnalité émergente : la théorie des AMAS. *RIA*, 17(4) :591–626, 2003.

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

[8] 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.

[9] G. Picard, C. Bernon, and M.-P. Gleizes. Emergent Timetabling Organization. In *4th International Central and Eastern European Conference on Multi-Agent Systems (CEEMAS'05)*, volume 3690 of *LNAI*, pages 440–449. Springer, 2005.

[10] 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.

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

[12] 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.