N° d'ordre: 01/2008-D/IN

RÉPUBLIQUE ALGÉRIENNE DÉMOCRATIQUE ET POPULAIRE
MINISTÈRE DE L'ENSEIGNEMENT SUPÉRIEUR ET DE LA RECHERCHE SCIENTIFIQUE

UNIVERSITÉ DES SCIENCES ET DE LA TECHNOLOGIE HOUARI BOUMEDIÈNE

FACULTÉ D'ÉLECTRONIQUE ET D'INFORMATIQUE

THÈSE

Présentée pour l'obtention du diplôme de DOCTORAT
EN : INFORMATIQUE

Spécialité : Programmation et systèmes

par : Dalila BOUGHACI

Thème :

APPROCHES METAHEURISTIQUES POUR LES PROBLEMES DIFFICILES (MAX-SAT et PDG)

Soutenue publiquement le 23 Février 2008, à l'USTHB devant le jury :

Meziane. AIDER	Professeur de l'USTHB	Président
Habiba. DRIAS	Professeur de l'USTHB	Directrice de thèse
Belaïd.BENHAMOU	Maître de Conférences, HDR de l'U. Provence	Examinateur
Riadh. BABA-ALI	Maître de Conférences de l'USTHB	Examinateur
Hamid. AZZOUNE	Maître de Conférences de l'USTHB	Examinateur
Saliha. OUKID	Maître de Conférences de l'U. Blida	Examinatrice

Remerciements

Je tiens à remercier Madame Habiba Drias, Professeur à l'USTHB, qui a dirigé ma thèse. Je la remercie vivement pour sa constante sollicitude, son soutien moral et son assistance matérielle pour mener à bien l'élaboration de cette thèse.

J'adresse mes sincères remerciements à Monsieur Belaïd Benhamou, Maître de conférences (HDR) de l'Univesité de Provence, pour m'avoir accueilli au sein de son équipe et pour m'avoir suivi et conseillé dans mon travail.

Je remercie tout particulièrement Monsieur le Professeur Meziane Aider, qui me fait l'honneur de présider ce jury.

Je remercie Madame Saliha Oukid, Maître de conférences de l'Université de Blida. Messieurs Riadh Baba-Ali et Hamid Azzoune, Maîtres de conférences de l'USTHB, d'avoir bien voulu faire partie de mon jury.

Que toute personne ayant contribué de prés ou de loin à l'élaboration de ma thèse, trouve ici ma parfaite gratitude.

Table des matières

Introduction Générale

Ces dernières années, les métaheuristiques ont connu un succès grandissant dans les domaines de la recherche opérationnelle et de l'intelligence artificielle. Elles ont prouvé leur efficacité et leur performance dans la résolution de nombreux problèmes d'optimisation combinatoire à savoir le problème du voyageur de commerce, les problèmes d'affectation, les problèmes d'ordonnancement ainsi que le problème de satisfiabilité.

Les métaheuristiques se caractérisent par leur conception et leur philosophie. Elles reposent sur des outils mathématiques comme les fonctions à optimiser et de contrôle dont l'objectif consiste à éviter les solutions de mauvaise qualité. La recherche taboue, le recuit simulé, les algorithmes génétiques et la recherche dispersée font partie intégrante de ces algorithmes.

Dans le but d'étudier l'apport de métaheuristiques dans la résolution de problèmes complexes, deux problèmes *NP-Complets* sont considérés dans cette thèse, à savoir : le problème de satisfiabilité (*SAT en abrégé*) et le problème de la détermination du gagnant dans les enchères combinatoires (*PDG*).

Notre thèse comporte deux grandes parties, dont le point commun est la résolution de problèmes NP-complets.

0.1 Partie 1

Le problème *SAT* de la logique booléenne est le premier problème démontré *NP-Complet* [Cook, 1971 ; Garey et Johnson, 1979]. C'est la satisfaisabilité d'une formule propositionnelle donnée sous forme clausale : conjonction de clauses, où chaque clause est une disjonction de littéraux, et un littéral est une variable ou sa négation. Le problème est de décider alors s'il y a une affectation de valeurs de vérité aux variables propositionnelles qui rend la formule vraie. Dans le cas où cette affectation n'existe pas, l'ensemble de clauses est dit non satisfaisable et une variante du problème appelée *MAX-SAT* est introduite. Cette variante consiste à chercher une affectation de valeurs de vérité qui maximise le nombre de clauses satisfaites.

Le problème *SAT* est crucial en intelligence artificielle, notamment en démonstration automatique où tout test de déduction se ramène à un problème de satisfaisabilité. D'autres formes de raisonnement incluant le raisonnement par défaut, le diagnostic, la planification et l'interprétation d'images font un appel direct à la satisfaisabilité.

Vu le rôle important que présente le problème *SAT*, plusieurs algorithmes ont été développés pour le résoudre. Ces algorithmes peuvent être divisés en deux classes principales [Hoos et Stutzle, 2005] :

- La première classe englobe les algorithmes complets dont le principe essentiel consiste généralement à énumérer, souvent de manière implicite, l'ensemble des solutions de l'espace de recherche. La plupart de ces algorithmes sont basés principalement sur la procédure de Davis-Putnam-Loveland [Davis et Putnam, 1960, Davis *et al.*, 1962].
  Parmi les algorithmes complets utilisés pour résoudre le problème *SAT*, nous citons : Satz [Li et Anbulagan, 1997], Sato [Zhang, 1997], et Chaff [Moskewicz *et al.*, 2001]. Quelques algorithmes complets sont proposés pour le problème *MAX-SAT* par [Borchers et Furman, 1999 ; Alsinet *et al.*, 2003 ; Xing et Zhang, 2005 ; Li *et al.*, 2006].
  Les méthodes complètes ont permis de trouver des solutions exactes pour des problèmes de taille raisonnable. Cependant, ces méthodes rencontrent généralement des difficultés face aux applications de taille importante.
- La deuxième classe englobe les méthodes incomplètes qui sont basées sur la recherche locale ou les algorithmes évolutionnaires. Le recuit simulé [Hansen et Jaumard 1990], GSAT [Selman *et al.*, 1992], Walksat [Selman *et al.*, 1994], Novelty [Mcallester *et al.*, 1997], la recherche tabou [Mazure *et al.*, 1997], la recherche locale guidée [Mills et Tsang, 2000], AdaptNovelty [Hoos, 2002] et G^2wSAT[Li *et al.*, 2005] sont des exemples d'algorithmes de recherche locale.
  Parmi les approches évolutionnaires proposées pour le problème *SAT*, nous citons : la recherche dispersée [Drias et Khabzaoui, 2001], les algorithmes de colonies de fourmis (ACS)[Drias et Ibri, 2003 ; Drias *et al.*, 2003], les algorithmes d'essaims d'abeilles [Drias *et al.*, 2005], les algorithmes génétiques [Hao *et al.*, 2003] et les algorithmes mémétiques [Marchiori et Rossi, 1999 ; Gottlieb *et al.*, 2002 ; Boughaci *et al.*, 2004 ; Lardeux *et al.*, 2006].
  La recherche locale et les algorithmes évolutionnaires sont des exemples des algorithmes incomplets. Ce sont des approches capables de trouver une solution approchée en un temps raisonnable en particulier pour les instances de très grande taille [Lardeux *et al.*, 2005].

Pour plus de détails sur les méthodes de résolution de problèmes *SAT* et *MAX-SAT*, le lecteur pourra se référer à [Hoos et Stutzle, 2005].

Nous nous intéressons, dans la première partie de cette thèse au problème *MAX-SAT*. Les différents travaux réalisés au cours de cette première partie de cette thèse ont mené à plusieurs approches métaheuristiques pour le problème *MAX-SAT*.

Les principaux algorithmes que nous avons proposés sont basés sur les algorithmes évolutionnaires [Boughaci *et al.*, 2008 ; Boughaci *et al.*, 2007a ; Boughaci *et al.*, 2007b ; Boughaci *et al.*, 2005 ; Boughaci et Drias, 2005a ; Boughaci et Drias, 2005c ; Boughaci *et al.*, 2004 ; Boughaci et Drias, 2004a].

Pour contribuer à la résolution du problème *MAX-SAT* par les approches évolutionnaires, nous proposons, tout d'abord, une nouvelle stratégie de sélection qui se base sur la diversité et la qualité (*fitness*) pour choisir une collection d'individus qui vont participer à la phase de reproduction et donner une descendance. Ensuite, nous utilisons un opérateur de combinaison spécifique au problème *MAX-SAT* pour générer de nouveaux enfants qui sont améliorés par une recherche locale stochastique (SLS).

Les trois composantes proposées sont incorporées, premièrement, dans un algorithme génétique pour améliorer sa performance. Ensuite, une variante de recherche dispersée, utilisant les trois composantes déjà citées, est étudiée pour le problème *MAX-SAT*.

Plusieurs méthodes évolutionnaires ont été étudiées pour le problème *MAX-SAT* avec une contribution primaire de la part de Habiba Drias. Le premier algorithme de recherche dispersée pour le problème *MAX-SAT* a été développé par [Drias et Khabzaoui, 2001]. L'approche est basée sur les combinaisons linéaires permettant de produire de nouvelles solutions qui sont améliorées par une recherche locale. L'algorithme proposé a été utilisé pour résoudre la classe de problème de *Jnh*. D'autres approches évolutionnaires ont été étudiées pour la classe *Jnh*, nous citons : l'algorithme de colonies de fourmis (ACS)[Drias et Ibri, 2003 ; [Drias *et al.*, 2003] et l'algorithme d'essaims d'abeilles [Drias *et al.*, 2005].

Nos approches présentent des similitudes avec les algorithmes évolutionnaires déjà proposés [Marchiori et Rossi, 1999 ; Drias et Khabzaoui , 2001 ; Lardeux *et al.*, 2006] et qui remplacent la phase de mutation par une recherche locale. Nos méthodes se distinguent par leur stratégie de sélection qui considère la fitness et la diversité pour choisir les individus qui vont survivre et qui serviront à produire d'autres générations, leur recherche locale basée sur Walksat, et leur manière de gérer la population d'individus basée sur la fitness et la diversité. Notre objectif est d'associer à chaque génération une collection composée de deux types d'individus : ceux ayant une forte fitness, et ceux qui sont les plus dispersés dans l'espace de recherche. Ceci permet d'éviter la convergence prématurée connue comme étant l'obstacle majeur des algorithmes génétiques.

L'opérateur de croisement spécifique au problème *MAX-SAT* que nous utilisons pour créer de nouveaux enfants est similaire à celui proposé par [Lardeux *et al.*, 2006] dans le sens où il restaure la consistance des clauses falsifiées simultanément par les parents et essaie de minimiser le nombre de clauses falsifiées par le fils résultant des parents. Toutefois, nous exploitons différemment cet opérateur : d'une part, son application est tributaire d'une probabilité représentant le taux de croisement, d'autre part il combine deux parents appartenant à une collection d'individus meilleurs et diversifiés. Le nouveau fils est ensuite amélioré par la recherche locale stochastique et rajouté à la collection d'individus dans le cas où il améliore la qualité ou la diversité de la collection courante. Autrement, il sera écarté.

Dans le but de tester et de prouver l'efficacité de nos approches, une étude comparative avec quelques algorithmes de l'état de l'art concernant *MAX-SAT* à savoir Walksat [Selman *et al.*, 1994], FlipGA [Marchiori et Rossi, 1999] et GASAT [Lardeux *et al.*, 2006] est établie dans cette première partie de la thèse.

0.2 Partie 2

Dans la deuxième partie de cette thèse, nous nous intéressons au problème de la détermination du gagnant (*PDG*) dans les enchères combinatoires : un ensemble d'objets est soumis à la vente face à plusieurs acheteurs, chaque acheteur, pour des raisons de complémentarité entre les objets, désire acheter un sous-ensemble d'objets qui lui est propre, et dont il fournit une estimation. Les conflits entre les acheteurs naissent des éventuelles intersections entre les sous-ensembles. Le vendeur doit alors résoudre un problème d'optimisation combinatoire NP-complet [Rothkopf *et al.*, 1998] pour réaliser la vente qui lui rapportera le plus.

Plusieurs méthodes ont été proposées pour résoudre le problème de la détermination du gagnant. Parmi les méthodes complètes pour le PDG, nous citons : Branch-on-Items (BoI) [Sandholm et Suri, 2000], Branch-on-Bids (BoB)[Sandholm et Suri, 2000], et (CABoB) [Sandholm *et al.*, 2001]. CASS (Combinatorial Auction Structural Search) est un algorithme Branch-and-Bound pour le PDG proposé par [Fujishima *et al.*, 1999]. Dans [Leyton-Brown *et al.*, 2000a], les auteurs ont proposé CAMUS (Combinatorial Auctions Multi-Unit Search). [Rothkopf *et al.*, 1998] ont proposé une approche de programmation dynamique. Nisan [Nisan, 2000] a proposé une méthode de programmation linéaire. [Anderson *et al.*, 2000] ont développé un autre algorithme exact basé sur la programmation en nombre entier. Hollande et O'sullivan ont employé la programmation par contrainte pour résoudre une enchère combinatoire particulière de Vickrey [Hollande et O'sullivan, 2004].

D'autres part, quelques méthodes incomplètes ont été étudiées pour le PDG. Nous citons : le recuit simulé hybridé avec un Branch-and-Bound [Guo *et al.*, 2004 ; Guo *et al.*, 2006] et la recherche locale Casanova [Hoos et Boutilier, 2000].

Le processus de recherche dans Casanova est similaire à celui d'adaptNovelty, la méthode de recherche locale proposée par le même auteur pour résoudre le problème de la satisfiabilité [Hoos, 2002].

Récemment, [Guo *et al.*, 2006] ont proposé une heuristique de recuit simulé combinée avec un algorithme de type Branch-and-Bound. Les résultats sont très encourageants et la nouvelle méthode surpasse clairement Casanova. L'heuristique proposée fait appel tout d'abord à un prétraitement pour exclure les offres qui peuvent mener aux solutions suboptimales, ce qui permettra d'améliorer le temps de réponse du processus de recherche.

De notre part, nous proposons, dans la deuxième partie de cette thèse, une solution basée sur la recherche taboue pour résoudre le problème de la détermination du gagnant (PDG).

La recherche taboue est l'une des méthodes métaheuristiques. Elle a été utilisée avec un grand succès pour résoudre divers problèmes d'optimisation. Sa capacité de résoudre divers problèmes nous a motivé à proposer une approche basée sur le processus de recherche taboue pour résoudre le problème PDG. La méthode que nous proposons utilise le codage de clef aléatoire, RK (Random Key Encoding) introduit par *Bean* pour les problèmes d'ordonnancement [Bean, 1994]. Ce codage nous permet de manipuler des solutions réalisables. Par conséquent, aucune pénalité additionnelle n'est nécessaire pour éliminer les solutions inadmissibles.

Plus précisement, notre recherche taboue démarre d'une solution créée aléatoirement et selon le codage RK, puis la meilleure allocation voisine est sélectionnée pour être la prochaine solution courante. Pour produire des solutions voisines, nous définissons deux mouvements qui sont : "On-Bid" et "On-Item".

Afin d'échapper aux allocations déjà visitées, notre stratégie de recherche utilise une liste pour maintenir les numéros d'offres récemment choisies. Ces numéros sont dits "mouvements tabous". Quand un mouvement tabou appliqué à une allocation courante donne une meilleure solution ; nous acceptons ce mouvement malgré son statut tabou par critère d'aspiration. Une étape de diversification est lancée pour explorer de nouvelles régions, ce qui évitera la stagnation du processus de recherche. La recherche s'arrête après un certain nombre d'itérations fixé d'une manière empirique.

Nous notons que les offres sont en conflit si elles partagent un objet. Pour ne manipuler que des allocations réalisables tout au long du processus de recherche taboue, nous définissons un graphe de conflit où les sommets sont les offres et les arcs relient les offres qui ne peuvent pas être acceptées ensemble. Ce graphe est utile pour enlever toutes les offres incompatibles qui peuvent se produire dans les

allocations courantes quand de nouvelles offres sont considérées.

Des expérimentations numériques sont réalisées sur des benchmarks de diverses tailles dans le but de tester et de prouver l'efficacité de notre approche.

D'autre part, nous simulons le mécanisme d'enchères combinatoires au moyen d'un système multiagents. Des agents vendeurs et des agents acheteurs interagissent selon leurs besoins dans un environnement d'un marché électronique afin de vendre et d'acheter des objets. Dans un tel environnement, les agents vendeurs et acheteurs ont la possibilité de lancer des offres sur un ensemble d'objets provenant d'une base finie. L'agent vendeur reçoit les demandes des agents acheteurs pour des paquets d'objets différents. Son but étant d'allouer ces objets de sorte à maximiser ses gains. Il s'agit d'un problème d'optimisation, c'est pourquoi nous proposons un agent de recherche taboue pour évaluer les offres afin de trouver une solution optimale au problème.

0.3 Organisation du document

Cette thèse est organisée comme suit :

Le premier chapitre est consacré aux généralités et aux principaux concepts du problème SAT.

Le deuxième chapitre présente les mécanismes d'enchères existants.

Le troisième chapitre est consacré à une contribution de la résolution du problème MAX-SAT. Nous présentons tout d'abord nos approches évolutionnaires que nous avons conçues et mises en oeuvre. Ensuite, nous exposons les différents résultats numériques réalisés sur plusieurs types de benchmarks.

Le quatrième chapitre est consacré à la résolution du problème de la détermination du gagnant PDG dans les enchères combinatoires. Une recherche taboue est proposée pour le PDG et des expérimentations numériques sont réalisées sur des benchmarks de diverses tailles dans le but de tester et de prouver l'efficacité de notre approche.

La première partie du cinquième chapitre décrit le modèle d'agents proposé pour simuler le mécanisme d'enchères combinatoires. La deuxième partie de ce chapitre montre la possibilité d'intégrer un système multiagents et une métaheuristique de recherche taboue au sein d'une place de marché virtuel.

Enfin, la conclusion générale porte sur les résultats obtenus et dresse quelques perspectives sur la suite de notre travail.

Chapitre 1

Le problème de satisfiabilité SAT

1.1 Introduction

Le problème de satisfiabilité (SAT) est l'ancêtre des problèmes *NP-Complets* [Cook 1971 ; Garey et Johnson, 1979 ; Drias, 1993 ; Drias, 1998 ; Drias, 1999].

Beaucoup de problèmes *NP-Complets* appartiennent à divers domaines, tels que la théorie des graphes (le problème du voyageur de commerce, le problème du circuit hamiltonien et le problème de coloration d'un graphe), la géométrie, les jeux, la théorie des langages et l'ordonnancement des tâches. Ces problèmes peuvent être formulés comme un problème de satisfiabilité ou une de ses variantes.

D'autre part, le problème SAT a un impact sur le domaine de l'intelligence artificielle notamment dans l'élimination de l'incohérence des bases de connaissances dans les systèmes experts, où les implications de la base de connaissance peuvent être transformées sous forme de clauses en utilisant des formules bien connues de l'algèbre de Boole.

Dans ce qui suit, nous allons définir d'une manière formelle le problème de satisfiabilité SAT. Nous citons certaines de ses variantes et nous présentons quelques méthodes de résolution.

1.2 Complexité

En théorie de la complexité, les problèmes sont regroupés en classes de complexité en fonction des ressources mises en oeuvre par les algorithmes qui existent pour les résoudre [Papadimitrion, 1994].

Les deux principales classes de complexité sont :

- La classe P : est la classe des problèmes de décision résolubles en temps polynomial.
- La classe NP : est la classe des problèmes de décision pour lequels la réponse

oui peut être décidée par un algorithme non- déterministe en un temps polynomial.

Parmi les problèmes de la classe NP, certains problèmes sont plus difficiles que d'autres, ce sont les problèmes NP-Complets. Le problème SAT est NP-Complet [Cook, 1971].

1.3 Notation et formulation du problème SAT

Soient :
- X un ensemble de n variables booléennes.
- C un ensemble de m clauses dont chacune est la disjonction d'un ensemble de littéraux associés aux variables de X, tel que chaque littéral est une variable x_i ou sa négation $\bar{x_i}$. On définit formellement le problème SAT comme une formule logique écrite sous forme normale conjonctive (conjonction de clauses) et on cherche une affectation de valeurs de vérité aux variables de X qui rend cette formule vraie.

$$F = \bigwedge_{i=1}^{m} C_i \quad avec \quad C_i = \bigvee_{k=1}^{k_i} L_{iki}, \quad L_{iki} \in \{x_j, \bar{x_j}\}, \quad j = 1, ..n$$

Une instance de SAT est présentée comme suit :

$$C_1 = L_{11} \vee L_{12} ... \vee L_{1k1}$$
$$C_2 = L_{21} \vee L_{22} ... \vee L_{2k2}$$
$$.$$
$$.$$
$$.$$
$$C_m = L_{m1} \vee L_{m2} ... \vee L_{mkm}$$

Où :
- n est le nombre de variables booléennes x_i.
- m est le nombre de clauses.
- k_i est la longueur de la clause C_i, c'est-à-dire le nombre de littéraux figurant dans cette clause.
- L_{iki} est un littéral.
- k_1, k_2 ... k_m sont respectivement les longueurs des clauses C_1, C_2 .... C_m.

Une instance est satisfaite quand toutes ses clauses sont vraies, sinon elle sera contradictoire. Le but est de trouver une affectation de valeurs de vérité aux variables, c'est-à-dire pour chaque variable x_i affecter la valeur vraie =1 ou faux=0, qui satisfasse la conjonction de clauses.

Quand il n'existe pas d'affectation qui rend la formule F vraie, on s'intéresse à d'autres variantes du problème *SAT*.

1.4 Quelques variantes du problème SAT

Parmi les variantes du problème *SAT*, nous citons :

- **Le problème *SAT* de décision :** qui permet de répondre par "oui" si la donnée est satisfaite, par "non" si celle- ci est contradictoire.
- **Le problème *SAT* de recherche des solutions d'une donnée *SAT* :** qui consiste en l'exhibition de certaines solutions de la donnée *SAT*. La résolution de ce problème permet de répondre au problème de décision.
- **Le problème de dénombrement des solutions d'une donnée *SAT* :** qui consiste à déterminer le nombre de solutions de l'instance *SAT*. La résolution de ce problème permet aussi de répondre au problème de décision associé à *SAT*.

A ces variantes, on peut rajouter les problèmes *K-SAT*, *MAX-SAT* et *MAX-W-SAT* [Garey et Johnson, 1979] qui sont présentés comme suit :

- **Le problème *K-SAT* :** c'est une variante du problème SAT où toutes les clauses ont au maximum K littéraux.
- **Le problème *MAX-SAT* :** dans un tel problème on cherche une affectation booléenne qui maximise le nombre de clauses satisfaites.
- **Le problème *MAX-W-SAT* :** qui est la satisfiabilité maximale pondérée. C'est une variante de MAX-SAT qui associe à chaque clause C_i un poids W_i. Le but consiste à trouver une affectation aux variables qui maximise la somme des poids des clauses satisfaites.

Remarque :

Le problème *K-SAT* est non polynomial pour $K \succeq 3$, mais il devient polynomial pour $K = 2$ alors que le problème *MAX-SAT* reste non polynomial même quand les clauses contiennent deux littéraux seulement.

1.5 Méthodes de résolution

Le problème *SAT* est crucial en intelligence artificielle, notamment en démonstration automatique où tout test de déduction se ramène à un problème de satisfiabilité. D'autres formes de raisonnement incluant le raisonnement par

défaut, le diagnostic, la planification et l'interprétation d'images font un appel direct à la satisfiabilité.

Vu l'importance du problème *SAT*, plusieurs méthodes ont été développées pour le résoudre. Ces méthodes se divisent en deux catégories : les méthodes complètes et les méthodes incomplètes [Hoos et Stutzle, 2005 ; Hoos et Boutilier, 2000],

- **Méthodes complètes :** elles permettent de trouver une solution exacte en utilisant soit la programmation linéaire ou bien la programmation dynamique ou les méthodes arborescentes.
  Pour les problèmes *SAT*, les principaux algorithmes complets sont basés sur la procédure de Davis et Putnam [Davis et Putnam, 1960 ; Davis *et al*, 1962] qui est une méthode de recherche énumérative. Des techniques sont proposées par la suite pour renforcer les algorithmes complets. Parmi ces techniques, nous avons : la détection de symétries [Benhamou et Sais, 1992] et la détection d'équivalences [Li, 2000].
- **Méthodes incomplètes :** les méthodes incomplètes sont basées sur les algorithmes de recherche locale et les algorithmes évolutionnaire. La recherche taboue [Mazure *et al*, 1997], le recuit simulé [Hansen et Jaumard 1990], la recherche locale guidée [Mills et Tsang, 2000], les algorithmes génétiques [Frank, 1994 ; Hao *et al*, 2003), ASAP [Gottlieb *et al*, 2002], la recherche dispersée [Drias et Khabzaoui, 2001 ; Boughaci et Drias, 2005], les algorithmes de colonies de fourmis [Drias et Ibri, 2003 ; Drias *et al.*,2003], les algorithmes d'essaims d'abeilles [Drias *et al.*, 2005] et les algorithmes mémétiques [Marchiori et Rossi, 1999 ; Boughaci *et al*, 2004 ; Lardeux *et al*, 2006] sont des exemples des algorithmes incomplets. Ce sont des approches capables de trouver une solution approchée en un temps raisonnable en particulier pour les instances de très grande taille.

1.5.1 Méthodes complètes

Les méthodes complètes permettent de construire une solution exacte en utilisant soit la programmation linéaire ou bien la programmation dynamique ou les méthodes arborescentes. Les algorithmes complets pour le problème SAT sont basés principalement sur la procédure Davis-Putnam. Ils diffèrent essentiellement dans la règle de branchement utilisée pour explorer l'arbre de recherche.

Parmi les algorithmes complets donnant de bons résultats, nous citons satz[Li and Anbulgan, 1997] et zchaff[Moskewicz *et al*, 2001].

Contrairement au problème de décision, le problème MAX-SAT a été moins travaillé. Parmi les algorithmes complets dédiés au problème MAX-SAT, nous citons : [Borchers et Furman, 1999 ; Alsinet *et al.*, 2003 ; Xing et Zhang, 2005 ; Li *et al.*, 2006].

1.5.1.1 Davis et Putnam

Le principe de la procédure de Davis et Putnam (DP en abrégé) revient à réduire l'instance de SAT jusqu'à l'obtention d'une sous instance, à partir de laquelle il devient possible de répondre à la satisfiabilité [Davis et Putnam, 1960 ; Davis *et al*, 1962] . L'algorithme de la procédure de Davis et Putnam, donné dans l'algorithme1, est basé sur les deux règles suivantes :

- **La règle de la clause unitaire :** une clause unitaire est une clause qui contient un seul littéral. Par convention, une clause vide est une clause qui ne contient aucun littéral, et elle est forcément fausse. La règle de la clause unitaire est appliquée comme suit : considérer les clauses unitaires et fixer leur littéral à vrai, puis supprimer toutes les clauses qui contiennent ce littéral lit, ainsi que toutes les occurrences négatives de lit dans les clauses restantes. Cette règle est appliquée jusqu'à ce que la donnée soit vide (indiquant que la donnée SAT est satisfiable), ou qu'elle contient une clause vide (indiquant que la donnée SAT est non satisfiable).
- **La règle d'éclatement :** dans le cas où la règle de la clause unitaire n'est pas applicable, l'algorithme de la procédure de Davis et Putnam s'appuie sur la règle d'éclatement. Cette dernière consiste à choisir une variable u, qui n'a pas de valeur de vérité et lui affecter une valeur. Si l'appel de la procédure de Davis et Putnam sur la formule simplifiée retourne non satisfiable, la variable u est réinitialisée à la valeur opposée et la procédure DP est appliquée de nouveau.

Algorithm 1 : Davis et Putnam.

1: Soit un ensemble de clauses C défini sur un ensemble de variables X
2: **Si** C est vide **alors** retourner "satisfiable"
3: **Si** C contient une clause vide **alors** retourner "non satisfiable"
4: **Règle de clause unité :**
Si C contient une clause unité cl **alors** affecter à la variable mentionnée la valeur de vérité qui satisfait cl, et retourner le résultat de l'appel de DP sur la formule simplifiée.
5: **Règle d'éclatement "Splitting" :**
Sélectionner une variable x de X qui n'a pas de valeur de vérité
Affecter une valeur de vérité à cette variable, et appliquer DP sur la formule simplifiée ;
Si cet appel retourne "satisfiable" **alors** retourner "satisfiable"
Sinon initialiser x à la valeur opposée à celle assignée auparavant et retourner le résultat de l'appel de DP sur la formule simplifiée de nouveau.

1.5.2 Méthodes incomplètes

Les méthodes incomplètes sont basées généralement sur la recherche locale ou les algorithmes évolutionnaires ou hybrides. Elles sont très utiles pour les problèmes d'optimisation, permettant de trouver des solutions approchées pour les instances de très grande taille.

1.5.2.1 Recherche locale

La recherche locale démarre d'une solution initiale possible et essaie de l'améliorer, en cherchant une solution meilleure dans le voisinage courant. Un voisinage d'une solution X correspond à des éléments adjacents à X dont chacun est atteint par un changement dans la configuration courante. Le processus de recherche est réitéré jusqu'à ce qu'aucune amélioration dans la solution courante ne pourrait être faite. L'algorithme général de la recherche locale est donné dans l'algorithme2.

Algorithm 2 : La recherche locale.

1: Soit une solution initiale S
2: Amelior =vrai,
3: **Répéter**
4: S' = Meilleur Voisin(X)
5: **Si** coût(S) > coût(S') **alors** S = S'
6: **sinon** Amelior =faux
7: **Jusqu'à** Amelior= faux.

1.5.2.2 GSAT

La procédure de base GSAT commence par une solution aléatoire, puis procède de façon répétitive à changer l'affectation de la variable qui mène à la plus grande baisse dans le nombre total des clauses non satisfaites. Cette procédure est donnée dans l'algorithme3.

Les paramètres $MAX-ITER$ et $MAX-ESS$ déterminent respectivement combien de transitions la procédure entreprendra avant d'abandonner et recommencer, et combien de temps cette recherche peut être recommencée avant de quitter. La procédure GSAT permet de trouver une instanciation pour laquelle on a le maximum de clauses satisfaites. Néanmoins, elle pourrait échouer à trouver une affectation alors qu'elle existe. Pour cet effet, deux extensions de la procédure GSAT ont été introduites : Walksat [Selman *et al.*, 1994] et Novelty [Mcallester *et al.*, 1997].

1.5.2.3 Walksat

La procédure Walksat [Selman *et al.*, 1994] est une variante améliorée de GSAT utilisant des principes de recherche locale. La principale différence entre GSAT et Walksat est le choix de la variable à inverser. A chaque itération, Walksat sélectionne aléatoirement une clause fausse, ensuite la variable à flipper [1] est choisie par deux critères différents :

- Avec une probabilité p, choisir une variable x dont le flippage génère le moins de clauses fausses parmi toutes les variables de la clause fausse sélectionnée. Cette stratégie est appelée l'heuristique "*best*".
- Avec une probabilité 1-p, choisir aléatoirement une variable x de la clause fausse.

De nouvelles techniques pour choisir la variable x à inverser ont été étudiées : Novelty, tabu et adaptNovelty. La technique tabu consiste à interdire les variables récemment inversées. Novelty est basée sur le même principe que *best* mais elle prend aussi en compte le critère d'ancienneté qui permet d'éviter les configurations déjà explorées. La procédure adpatNovelty [Hoos, 2002] est basée sur l'heuristique Novelty améliorée par l'ajustement dynamique de la probabilité p.

Algorithm 3 : La Procédure GSAT.

1: Soit un ensemble de clauses C
   // $MAX-ITER$: est le nombre d'itérations
   // $MAX-ESS$: est le nombre d'essais
   // n : est le nombre de variables
2: **Pour** i :=1 **à** $MAX-ESS$
3: **Faire**
4: Soit X une affectation aléatoire
5: **Pour** i := 1 **à** $Max-ITER$
6: **Faire**
7: **Si** X satisfait C **alors** retourner X et arrêter
8: Soit x_i une variable dont le changement de l'affectation donne la plus grande baisse dans le nombre total de clauses de C qui ne sont pas satisfaites par X
9: X= X avec la valeur de x_i inversée.
10: **Fait**
11: **Fait**

[1] le flippage d'une variable consiste à inverser sa valeur de vérité de 1 à 0 ou de 0 à 1.

1.6 Recherche taboue

La recherche taboue est une métaheuristique introduite par Fred Glover [Glover 1986]. C'est une des méthodes efficaces pour la résolution de grand nombre de problèmes d'optimisation combinatoires [Glover, 1989 ; Glover, 1996]. Elle a été appliquée avec un grand succès à plusieurs domaines, nous citons : le problème de satisfiabilité [Mazure *et al.*, 1997 ; Boughaci et drias, 2005], l'évaluation d'offre pour le commerce électronique [Boughaci et Drias,2005], le problème d'affectation quadratique [Pardalos *et al.*, 1994], et le problème de routage [Greistorfer, 2002].

La recherche taboue est une méthode itérative. Elle démarre d'une solution initiale aléatoire, puis la meilleure solution voisine, est choisie pour être la solution courante. Pour éviter les solutions déjà visitées, la méthode utilise une liste taboue pour conserver une information sur les dernières itérations de l'algorithme ce qui permettra de se prémunir contre les cycles courts.

Les principaux paramètres de contrôle de la recherche taboue sont :

1. Le voisinage : la recherche taboue nécessite que pour toute solution s, il existe un sous-ensemble appelé $V(s)$ qui consiste en les solutions réalisables qui peuvent être atteintes par s en un seul mouvement. Un mouvement est un changement dans la configuration courante.
2. La liste taboue : pour éviter de cycler autour des solutions déjà visitées, la recherche taboue interdit les mouvements qui conduisent vers ces solutions. Une liste LT appelée : liste taboue est introduite et dans laquelle toute solution non permise est recensée.
3. La fonction d'aspiration : la recherche taboue fait intervenir un nouvel ingrédient appelé fonction d'aspiration et définie sur toutes les valeurs de la *fonction objectif*. Elle permet de revenir sur une solution taboue, s, si cette dernière mène à une solution de haute qualité, ceci afin d'exploiter une nouvelle région voisine de s.
4. Le critère d'arrêt : le processus itératif peut être interrompu après un certain nombre d'itérations ou quand on est très proche de la solution optimale f^*.

L'organigramme illustrant le processus tabou est donné dans la figure 1.1.

1.7 Recuit simulé

Le recuit simulé est une méthode inspirée de la technique du "recuit" qui consiste à chauffer un métal jusqu'à l'état liquide, puis on le refroidit très lentement en marquant des paliers de température de durées suffisantes. La stratégie de la baisse contrôlée de la température conduit à un état solide cristallisé, qui est un état stable, correspondant à un minimum d'énergie.

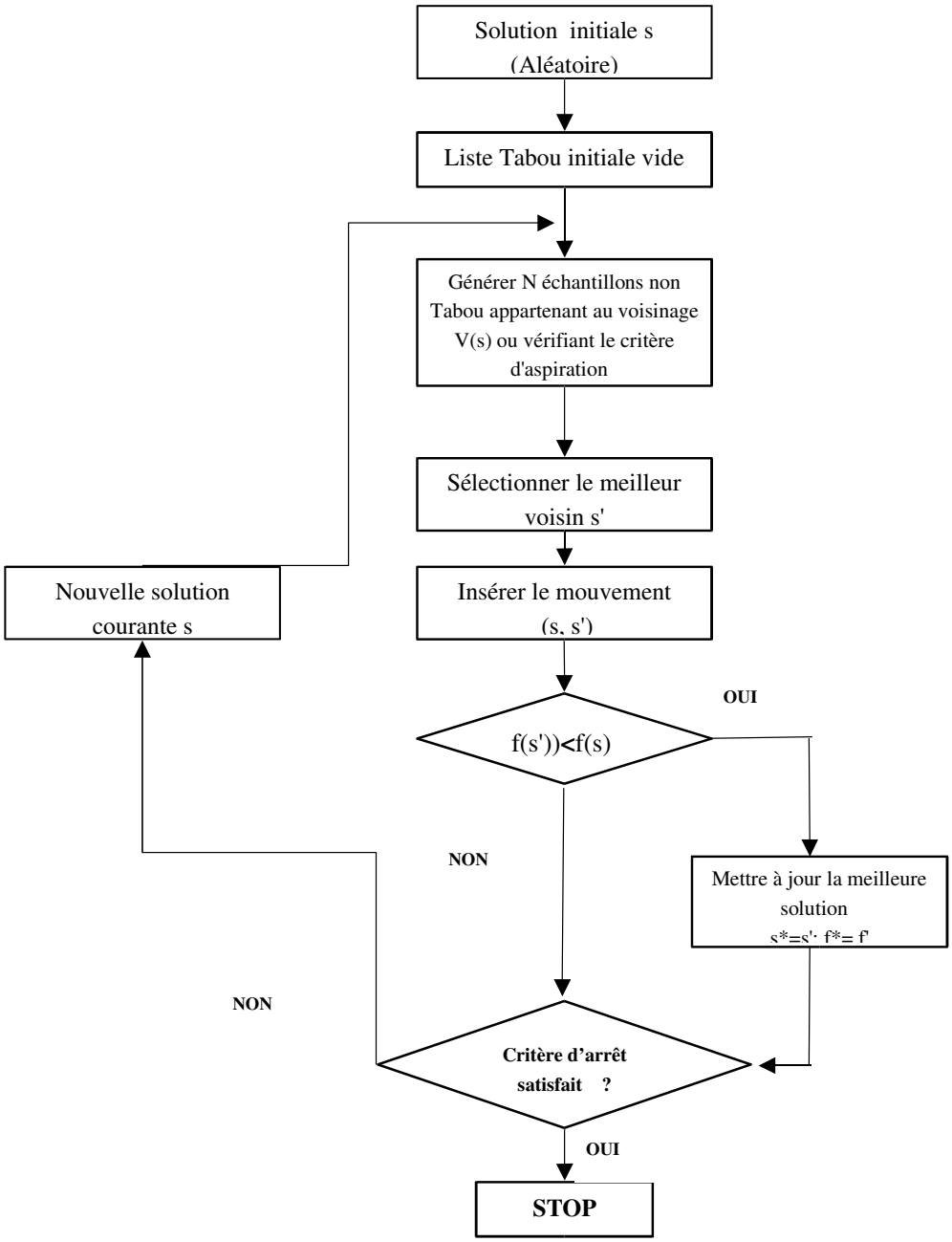

FIG. 1.1 – L'organigramme de la recherche taboue

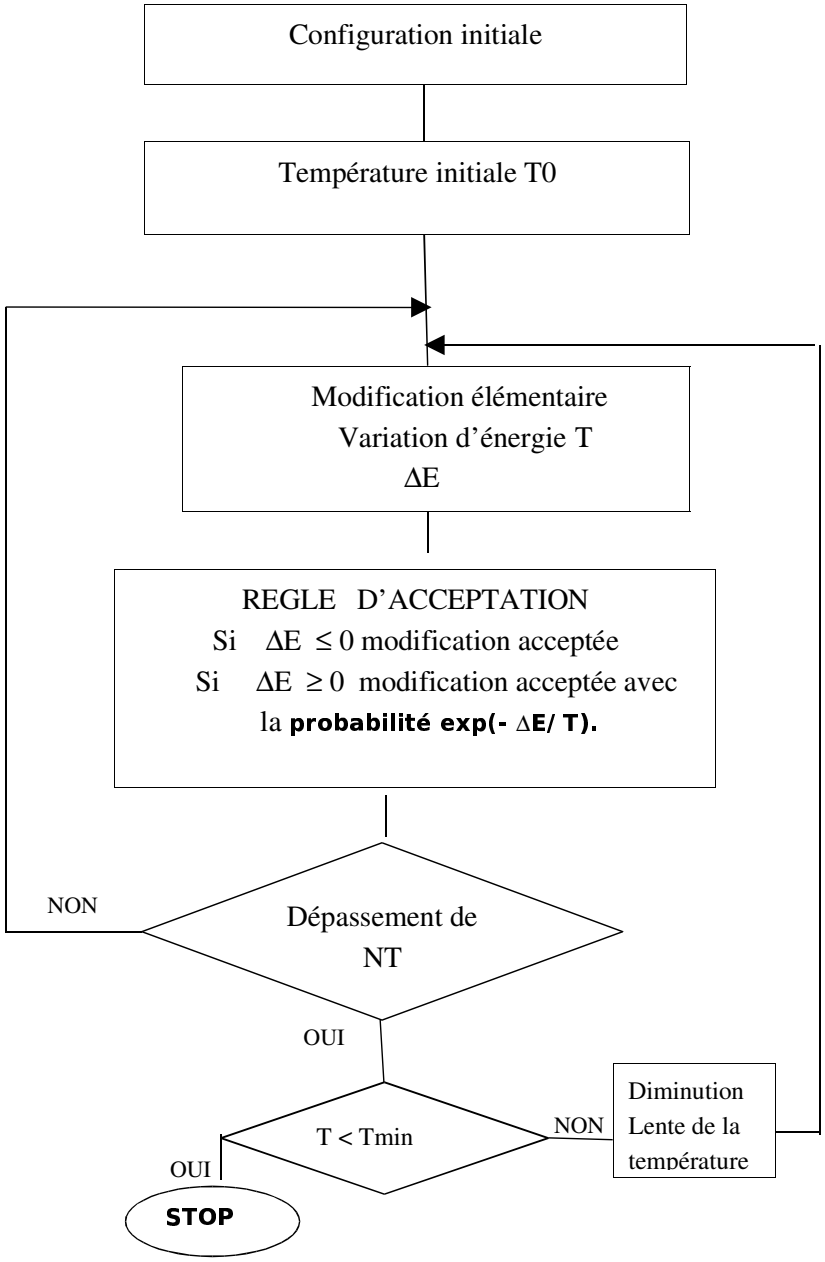

FIG. 1.2 – L'organigramme du recuit simulé

La méthode de recuit simulé consiste à transposer ce procédé à la résolution d'un problème d'optimisation combinatoire : la fonction de coût du problème, analogue à l'énergie du métal, est alors minimisée, avec l'introduction d'une "températureă"qui est dans ce cas un simple paramètre de contrôle de l'algorithme jouant le rôle de la température du système physique dans la technique du recuit [Siarry, 1995]. L'organigramme du recuit simulé est présenté dans la figure 1.2.

1.7.1 Description de la méthode du recuit simulé

Dans cette méthode, on part d'une configuration initiale représentant une solution réalisable du problème et d'une température T. Ensuite, on fait subir à la configuration une modification élémentaire, si cette transformation a pour effet de diminuer la fonction coût du problème, elle est acceptée. Si au contraire elle provoque une augmentation $\triangle E$ de la fonction coût, elle est acceptée tout de même avec la probabilité exp(-$\triangle E$) où exp dénote la fonction mathématique exponentielle. En appliquant itérativement cette règle d'acceptation, on engendre une séquence de configurations. Après qu'un certain nombre d'itérations a été atteint (qui dépend de la taille du problème), on réduit la température T et on effectue une nouvelle série d'itérations à cette nouvelle température. On répète le cycle de refroidissement et des transitions jusqu'à un certain critère d'arrêt qui correspond à une température finale faible(proche de zéro).

D'après cet organigramme, on comprend le rôle confié à la température. A haute température, exp(-$\triangle E/T$) est voisin de 1, donc la plupart des configurations sont acceptées, l'algorithme équivaut à une simple marche aléatoire dans l'espace de recherche. A basse température, exp(-$\triangle$ E/T) est voisin de 0, donc la plupart des configurations augmentant la fonction coût (l'énergie) sont refusées. A température intermédiaire, l'algorithme autorise de temps en temps des transformations qui dégradent la fonction coût, il laisse ainsi au système une chance de s'extraire d'un minimum local [Siarry, 1995].

1.8 Algorithmes génétiques

Un algorithme génétique (AG) [Holland, 1975 ; Goldberg, 1989] repose sur le principe de l'évolution naturelle d'organismes en respectant les phénomènes d'hérédité et la loi de survie énoncée par Darwin, selon lesquels, dans une population d'individus, ce sont les plus forts, c'est à dire les mieux adaptés à leur environnement qui survivent et pourront donner une descendance.

Le processus d'évolution se fait par deux mécanismes : la sélection naturelle et la reproduction.

1.8.1 Représentation chromosomique

Le chromosome code une solution au problème, il représente un point de l'espace de recherche. Dans le cas du codage binaire, utilisé dans les premières versions des algorithmes génétiques, un chromosome est représenté par un vecteur dont les éléments appartiennent à 0,1. Si la fonction à optimiser possède n variables, le chromosome sera constitué de la concaténation des n codages binaires. Il faut noter qu'il peut parfois être très difficile, de coder des solutions de cette manière. Le choix du codage a un impact sur les performances de l'algorithme génétique. En effet on doit choisir un codage qui n'est pas complexe dans le sens où le temps du décodage est minimal.

1.8.2 Population initiale

La population initiale d'individus/solutions est l'ensemble des individus qui représentent des solutions d'un problème donné. Cette dernière est générée aléatoirement.

1.8.3 Fonction d'évaluation

La fonction d'adaptation, décode un individu puis elle lui associe un coût. Un AG tend à maximiser ce coût, en permettant aux chromosomes ayant une grande valeur d'être retenus puis croisés (étape de reproduction).

1.8.4 Opérateurs génétiques

Les opérateurs génétiques définissent la manière dont les individus se recombinent et s'agencent pendant la phase de reproduction.

- **Sélection :** la phase de sélection spécifie les individus de la population qui doivent survivre. La méthode de base, appelée roue de loterie, attribue à chaque individu V_i une probabilité de survie P_i proportionnelle à son adaptation dans la population.
  Lors de la phase de sélection, les individus sont sélectionnés aléatoirement en respectant les probabilités P_i associées pour former la population de la nouvelle génération. Ceci s'effectue par le calcul d'une probabilité de sélection cumulée Q_i, telle que :
  $Q_i = \sum_{j=1}^{i} P_j, P_i = \frac{adaptationV_i}{\sum_{j=1}^{n} adaptationV_j}$, où n est la taille de la population.
  Puis, on génère aléatoirement un réel r sur l'intervalle [0,1].
  L'individu V_i est sélectionné lorsque $Q_i - 1 \prec \mathrm{r} \preceq Q_i$. De ce fait les individus les mieux adaptés sont sélectionnés plusieurs fois et les plus faibles rarement

voire jamais. Ce processus est réitéré n fois.

- **Croisement** : le croisement mélange les chromosomes des individus parents pour créer le code génétique d'un individu fils.
  Soient deux individus V_1 et V_2 juxtaposées, le croisement les coupe en un point choisi aléatoirement et produit deux nouveaux individus V_1' et V_2' après avoir échangé les parties coupées. Les deux individus V_1 et V_2 participant à cette opération sont assimilés aux parents, et les deux chaînes résultantes aux descendants.
- **Mutation :** la mutation est un changement aléatoire d'une ou plusieurs positions d'un individu. L'opérateur de croisement devient moins efficace avec le temps, car les individus deviennent similaires. C'est à ce moment que le phénomène de mutation prend toute son importance. Chaque position de l'ensemble des chaînes de la population a une probabilité P_m de subir une mutation à chaque génération. Cet opérateur permet de sortir des optima locaux.

Exemple 1 *L'opérateur de croisement uni-point : c'est un croisement qui utilise un point de coupure.*

$$\mathbf{1000} \mid \mathit{1011} \otimes \mathit{1101} \mid \mathbf{1001} = \mathbf{1000} \mid \mathbf{1001}$$

$\otimes$ représente l'opérateur de croisement.

Exemple 2 *L'opérateur de croisement multi-points qui choisit aléatoirement plusieurs points de croisement dans un chromosome puis échange les deux chromosomes des parents entre ces points pour produire le nouveau fils. Nous donnons ci-dessous un exemple d'un croisement à deux-points.*

$$\mathbf{10} \mid \mathit{0010} \mid \mathbf{11} \otimes \mathit{11} \mid \mathbf{0110} \mid \mathit{01} = \mathbf{10} \mid \mathbf{0010} \mid \mathbf{01}$$

Exemple 3 *L'opérateur de croisement uniforme qui décide quel parent contribuera à la production du fils, les bits sont copiés de premier ou de deuxième parent d'une manière aléatoire selon une certaine probabilité.*

$$\mathit{10001011} \otimes \mathit{11011001} = \mathbf{11011011}$$

Exemple 4 *Le croisement arithmétique qui effectue une certaine opération arithmétique pour produire le nouveau chromosome.*

$$10001011 \otimes 11011001 = \mathbf{10001101} \text{ (AND).}$$

Exemple 5 *La mutation qui est un opérateur produisant un individu modifié. Une mutation élémentaire consiste à inverser un gene (bit).*

$$100010\ \mathbf{1}\ 1 \Longrightarrow 100010\ \mathbf{0}\ 1$$

Les différentes étapes de l'AG sont données dans l'algorithme4.

Algorithm 4 : Algorithme Génétique.

1: Générer aléatoirement une population initiale d'individus.
2: Evaluation : Affecter à chaque individu son coût.
3: **Répéter**
4: **Sélection** : Etablir une liste de paires d'individus susceptibles de se reproduire, le critère de sélection favorise les meilleurs.
5: **Reproduction** : Appliquer les opérateurs génétiques à toutes les paires d'individus sélectionnées.
6: **Evaluation** : Affecter à chaque individu son coût.
7: **Remplacement** : Produire la nouvelle population en remplaçant les individus de manière à favoriser encore les meilleurs.
8: **Jusqu'à** un certain critère d'arrêt.

1.9 Recherche dispersée

La recherche dispersée [Glover 1999 ; Laguna et Glover, 1999 ; Laguna *et al.*, 1999] appelée en anglais *scatter search* est une métaheuristique basée population. Elle démarre d'une population initiale variée, en explorant des zones diverses de l'espace de recherche, et tente de l'améliorer tout au long du processus de résolution. L'approche permet de produire de nouvelles solutions à partir d'un ensemble de solutions appelées solutions de référence qui sont sélectionnées de la population de départ. La production de nouvelles solutions est faite en utilisant un opérateur de combinaison permettant d'obtenir des solutions admissibles. Les différentes étapes de la recherche dispersée sont données dans l'algorithme5.

Algorithm 5 : Recherche dispersée.

1: Générer une population initiale variée
2: Sélectionner un ensemble de solutions meilleures et diversifiées qui vont construire les solutions de référence
3: Appliquer la méthode de combinaison sur l'ensemble de référence pour construire des nouvelles solutions.
4: Améliorer les solutions combinées moyennant une recherche locale

1.9.1 Eléments de base de la recherche dispersée

1.9.1.1 Population initiale

La population initiale est l'ensemble des individus qui représentent des solutions d'un problème donné. Contrairement aux algorithmes génétiques où les solutions de départ sont générées aléatoirement, la recherche dispersée utilise une stratégie de diversification, pour avoir une population variée qui permettra d'explorer des zones diverses de l'espace de recherche.

A chaque solution un coût correspondant à la *fonction objectif* est associé. La population de départ est ordonnée suivant la *fonction objectif* en commençant par les meilleurs individus ayant un coût assez élevé.

1.9.1.2 Construction des solutions de référence R

Dans cette phase on spécifie les individus de la population initiale qui doivent se combiner et produire une nouvelle génération.

En effet le choix d'une solution pour l'ensemble R "solutions de références" est un point essentiel dans la recherche dispersée, pour lequel une bonne réflexion est nécessaire. Il existe deux stratégies de choix de l'ensemble R : Celle qui consiste à choisir un ensemble de solutions et de les évaluer suivant la *fonction objectif* pour pouvoir déduire les solutions de haute qualité, l'autre consiste à choisir un ensemble de solutions et de les évaluer suivant la disparité pour pouvoir déduire les solutions diverses. Par conséquent, on aura deux sous-ensembles R_1 de taille B_1 et R_2 de taille B_2, tel que les B_1 solutions sont de haute qualité et les B_2 solutions sont les points les plus éloignés les uns des autres dans l'espace de recherche.

1.9.1.3 Renouvellement de la population

Le renouvellement de la population, autrement dit la création d'une nouvelle génération, est obtenu par itération de l'algorithme de recherche dispersée qui va créer de nouvelles solutions et en détruire d'autres. A partir de l'ensemble R, la recherche dispersée génère des sous-ensembles appelés aussi points élites qui vont construire des solutions combinées en utilisant un opérateur de combinaison.

La recombinaison consiste à grouper les éléments de R par paire, par triplet, et ainsi de suite, c'est-à-dire avec un ensemble R de taille B on peut générer 4 types de solutions définis comme suit :

- **Type 1** : Il consiste à grouper les éléments de R par paire, puis, pour chaque paire de solutions appliquer l'opérateur de combinaison, améliorer et insérer la solution générée dans R.
- **Type 2** : Les solutions seront combinées trois à trois, en prenant un élément de type1 et on lui rajoute une meilleure solution qui n'apparaît pas dans le couple de solutions courant.

- **Type 3** : Les solutions seront combinées quatre à quatre. Pour chaque élément de type2, lui rajouter une bonne solution, puis, pour chaque quadruplet obtenu, appliquer l'opérateur de combinaison
- **Type 4 :** : Les i meilleurs points de R seront groupés pour donner d'autres Solutions combinées, où $i = 5$ jusqu'à B, c'est-à-dire le type 4 comprend des sous ensembles de taille 5,6...jusqu'à B meilleurs éléments.

1.9.1.4 Cycle d'amélioration

Dans cette phase, on part d'une configuration représentant une solution réalisable du problème obtenue par la méthode de combinaison. Ensuite, on fait subir à cette solution combinée des modifications élémentaires (série de transformations qui a pour effet d'améliorer le coût de la solution combinée). Afin d'avoir une solution meilleure, on applique itérativement cette méthode d'amélioration qui engendra une séquence de solutions. On répète le cycle d'amélioration et des transitions jusqu'à un certain critère d'arrêt qui correspond à un certain nombre d'itérations

1.9.1.5 Opérateur de combinaison

L'aveuglance exagérée des opérateurs génétiques classiques est à l'origine de plusieurs problèmes, tels que la convergence prématurée de l'algorithme génétique. Donc, il est intéressant d'utiliser un opérateur plus évolué prenant en considération le contexte du problème.

La combinaison structurée est un opérateur permettant de produire des solutions admissibles, à partir d'un ensemble de solutions appelées solutions élites.

Dans la recherche dispersée, la combinaison dépend de la *fonction objectif* à optimiser "contexte du problème" ainsi que de la représentation utilisée pour modéliser les solutions du problème.

1.9.2 Recherche dispersée pour le problème MAX-SAT

Le premier algorithme de recherche dispersée pour le problème *MAX-SAT* a été développé par [Drias et Khabzaoui , 2001]. L'approche démarre d'une population de solutions diversifiées. Ensuite, un certain nombre de solutions de meilleure qualité sont choisies de la population pour construire l'ensemble de référence. Les solutions de références subissent des combinaisons linéarires pour produire de nouvelles solutions qui sont améliorées par une recherche locale. Ce procéssus est réitéré jusqu'à ce que l'ensemble de référence ne change plus. Toutefois, la recherche dispersée peut être relancée pour un certain nombre de générations fixé d'une manière empirique.

1.10 Métaheuristique de colonies de fourmis

La métaheuristique de colonies de fourmis est une approche inspirée du comportement des fourmis. La mthode a été initialement proposée par [Dorigo, 1992 ; Dorigo *et al.*, 2006] pour la recherche de chemins optimaux dans un graphe.

Depuis les débuts de la méthode de colonies de formis son emploi s'est étendu à plusieurs problèmes d'optimisation. Parmi ces problèmes, nous citons le problème *MAX-SAT* où [Drias *et al.*, 2003] ont proposé un algorithme de fourmis spécifique au problème *MAX-SAT*. Il s'inspire du comportement des fourmis recherchant un chemin entre leur colonie et une source de nourriture.

L'approche est une méthode évolutionnaires combosée de deux phases : une phase de coopération et une phase d'adaptation. La phase de coopération consiste à examiner toutes les solutions de la population pour créer une mémoire globale. La phase d'adaptation individuelle fait appel à une méthode constructive qui utilise l'information contenue dans la mémoire globale pour générer de solutions admissibles. La phase d'adaptation individuelle correspond au travail réalisé par une fourmi isolée alors que la coopération correspond à l'ensemble de fourmis coopérants pour touver la bonne solution globale.

1.11 Métaheuristique d'essaims d'abeilles

La métaheuristique d'essaims d'abeilles est une méthode d'optimisation inspirée à l'origine du monde du vivant. Elle simule le déplacement d'un groupe d'abeilles.

Comme les colonies de formis, la méthode d'essaims d'abeilles se base sur la collaboration des individus entre eux. Elle s'appuie aussi sur le concept d'auto-organisation qui veut qu'un groupe d'individus peu intelligents peut posséder une organisation globale complexe.

L'algorithme commence par positionner chaque abeille (*généralement de façon aléatoire*) dans l'espace de recherche du problème. Chaque itération fait bouger les abeilles en fonction de trois composantes qui sont : sa vitesse actuelle, sa meilleure solution et sa meilleure solution obtenue dans son voisinage.

Un algorithme d'essaims d'abeilles pour le problème *MAX-SAT* a été proposé par [Drias *et al.*, 2005].

1.12 Algorithmes mémétiques

Les algorithmes mémétiques [Moscato,1989 ; SIMA, 2007] sont des algorithmes évolutionnaires basés population. Ils prenent leur nom de mot "meme" introduit

par R.Dawkins dans son livre "The Selfish gene"(1976). Le meme est l'unité d'information transmise entre les individus durant l'évolution culturelle.

Autrement dit, le meme est non seulement le matériel génétique transmis d'une génération à une autre mais aussi les concepts et les idées. L'évolution des idées dans un algorithme mémétique est obtenue par l'ajout d'une phase d'intensification forte dans le processus d'évolution (*généralement une recherche locale*).

Contrairement aux algorithmes génétiques qui imitent l'évolution biologique, les algorithmes mémétiques font évoluer les idées (memes). Un meme peut être modifié et amélioré chez l'individu alors qu'un gene ne peut être modifié durant la vie d'un individu. L'amélioration de meme dans un algorithme mémétique est obtenue par l'introduction de la recherche locale dans l'algorithme génétique.

Un algorithme mémétique est une hybridation entre l'algorithme génétique et la recherche locale. Plusieurs algorithmes de recherche locale ont été proposés. Il existe aussi plusieurs façons de concevoir un algorithme génétique. L'hybridation entre ces deux classes d'algorithmes permet d'envisager plusieurs combinaisons possibles.

Plusieurs algorithmes mémétiques ont été proposés pour le problème SAT. Parmi eux, nous citons : FlipGA[Marchiori et Rossi, 1999], GASAT[Lardeux *et al.*, 2006], et ASAP[Rossi*et al.*,2000].

L'algorithme FlipGA [Marchiori et Rossi, 1999] est un algorithme génétique qui utilise un croisement uniforme et qui remplace la phase de mutation par une recherche locale.

L'algorithme ASAP (Adaptative Evolutionary Algorithm for Satisfiability Problem) [Rossi*et al.*,2000] est une variate améliorée de FlipGA qui ajuste le taux de mutation (recherche locale) d'une manière dynamique, c'est-à-dire la probabilité (μ) d'inverser une variable est évoluée dans un intervalle où les bornes sont fixée empiriquement à [0 ; 0.5]. Le mécanisme d'adaptabilité utilisé dans ASAP permet de faire varier la probabilité (μ) ainsi que d'interdire certains flips.

GASAT est un algorithme génétique hybride proposé récemment par [Lardeux *et al.*, 2006]. Il utilise une recherche taboue comme technique locale d'amélioration et un croisement spécifique au problème SAT pour produire de nouveaux enfants et dont l'objectif est de minimiser le nombre de clauses falsifiées par les parents.

Nous avons proposés plusieurs algorithmes mémétiques pour le problème SAT. Parmi eux, nous citons : GAV, SGAV et SSV qui vont être détaillés dans le chapitre 3 de cette thèse.

Un exemple d'un algorithme mémétique peut être donnée dans l'algorithme6.

Algorithm 6 : Algorithme Mémétique.

1: Générer aléatoirement une population initiale d'individus.
2: Pour chaque individu appliquer la recherche locale pour améliorer sa fitness.
3: **Répéter**
4: **Reproduction** : Appliquer l'opérateur de croisement pour produire les enfants
5: **Amélioration** : Appliquer la recherche locale sur les enfants pour améliorer leur fitness.
6: **Remplacement** : Produire la nouvelle population en remplaçant les individus de manière à favoriser les meilleurs.
7: **Jusqu'à** (un certain critère d'arrêt).

1.13 Conclusion

Dans ce chapitre, nous avons étudié le problème de la satisfiabilité, certaines de ses variantes, ainsi que quelques méthodes existantes pour sa résolution.

Dans le chapitre qui suit, nous nous intéresserons au problème d'enchères combinatoires (PDG) qui est un problème difficile. Le *PDG* est un problème d'optimisation combinatoire assez complexe comme le problème MAX-SAT, une solution basée sur la recherche taboue est proposée dans le chapitre 4 alors que le chapitre 3 propose quelques métaheuristiques pour le problème MAX-SAT.

Chapitre 2

Le problème de la détermination du gagnant dans les enchères combinatoires

2.1 Introduction

Une enchère est un mécanisme qui permet d'allouer les ressources (objets, services) aux acteurs du commerce électronique (vendeurs/acheteurs) en se basant sur des règles prédéfinies. Ces règles définissent le processus d'échange de propositions, la détermination du gagnant et l'accord final.

Parmi les enchères les plus largement dominantes, nous citons : les enchères anglaises, les enchères hollandaises, les enchères à enveloppe scellées, et les enchères de Vickrey [McAfee et McMillan, 1987].

Face à ces enchères qui portent uniquement sur le prix et qui manipulent qu'un seul objet, d'autres mécanismes ont été proposés tels que les enchères combinatoires.

Dans ce qui suit, nous commençons tout d'abord par les enchères de base. Ensuite, nous détaillons le mécanisme d'enchères combinatoire, le modèle qui nous intéresse dans ce travail.

2.2 Enchères de base

Un mécanisme classique de vente aux enchères consiste en un vendeur souhaitant maximiser son gain et un ensemble d'acheteurs souhaitant minimiser leur perte en fonction de l'estime qu'ils portent à l'objet de la vente. Les enchères de base peuvent être données comme suit [McAfee et McMillan, 1987] :

1. L'enchère anglaise : c'est une enchère ouverte ou publique à prix ascendant.

Les enchérisseurs annoncent ouvertement leurs offres et augmentent le prix progressivement. Les enchères prennent fin quand personne ne surenchérit sur la dernière offre annoncée, le gagnant est le dernier enchérisseur, il s'acquitte du montant annoncé.

2. L'enchère hollandaise : il s'agit d'une enchère ouverte (publique) à prix descendants. Dans ce type d'enchère, un commissaire annonce le prix et le domaine progressivement. Les enchères cessent quand un enchérisseur accepte le prix annoncé. Le gagnant règle la somme à laquelle il s'est porté acquéreur.
3. L'enchère scellée au premier prix : c'est une enchère à enveloppe scellées, signifiant que les offres des participants sont cachées les uns aux autres. Le gagnant de l'enchère, celui ayant fait la meilleure offre, doit payer le montant exact de son offre. L'enchère se déroule en un seul tour, les participants ne peuvent proposer à soumettre. Un inconvénient major de ce type d'enchère est le fait que ces offres soumises par les participants sont inférieures à leur évaluation.
4. L'enchère scellés au deuxième prix : elle est appelée aussi enchère de Vickrey. Cette stratégie d'enchères a été proposée, pour la première fois, par Vickrey en 1961. Dans ce type d'enchères les offres sont cachées aux différents participants. Le gagnant ayant soumis la meilleure offre doit payer non pas le montant de son offre, mais le deuxième meilleur prix, proposé pour l'objet à vendre au cours de l'enchère.

2.3 Enchères multi-objets

Contrairement aux enchères portant uniquement sur le prix, d'autres modèles d'enchères ont été proposés. Nous citons :

1. Les enchères multi-attributs portant sur plusieurs caractéristiques d'un produit incluant non seulement son prix, mais aussi sa qualité, les conditions de livraison, etc. Les acheteurs défissent leurs préférences sur l'objet recherché et les vendeurs sont en concurrence sur tous les attributs spécifiés par l'acheteur.

2. Les enchères à multiples exemplaires portant sur un ensemble d'objets. Les enchères combinatoires sont des enchères à multiples exemplaires où les propositions peuvent s'effectuer sur une partie de l'ensemble d'objets.

D'une manière générale et en se basant sur les mécanismes d'enchères de base, on distingue deux manières pour faire une enchère multi-objets : soit d'une manière séquentielle (lancer les objets un par un) ou bien d'une manière simultanée

(lancer tous les objets à la fois). Nous donnons un exemple réel d'une enchère multi-objets, celle de la FCC[1].

2.3.1 Enchère de la FCC

Ce type d'enchère a été utilisé en juillet 1994 pour la mise en vente des droits sur les fréquences de téléphones cellulaires par la Federal Communication Commission. Dans cette enchère, chaque cellule a été mise en vente individuellement, cependant il était clair qu'un acheteur avait tout intérêt à opérer des cellules proches. Les enchères de la FCC concernent alors des rondes d'enchères simultanées ascendantes discriminantes ouvertes. Une ronde d'enchères est similaire à une enchère traditionnelle anglaise sauf qu'au lieu de vendre les objets les uns après les autres, les objets sont vendus tous en même temps (FCC).

2.3.2 Enchères combinatoires

Une enchère combinatoire est le mécanisme où les vendeurs et les acheteurs ont la possibilité de lancer des offres non plus simplement sur des objets isolés, mais sur un ensemble d'objets provenant d'une base finie. C'est une enchère multi-objets permettant la soumission d'offres sur une combinaison d'objets. Le prix proposé par un enchérisseur pour un objet dépendent alors des autres objets. Ce qui permet d'exprimer les préférences des enchérisseurs et rend possible une allocation plus efficace des biens.

Les enchères combinatoires ont plusieurs applications notamment en économie, la théorie des jeux et l'allocation des ressources dans les systèmes multi-agents [Vries et Vohra, 2003 ; Sandholm, 2006 ; Sandholm, 1999].

Une enchère combinatoire est une enchère multi-objets permettant la soumission d'offres sur une combinaison d'objets. Les prix proposés par un enchérisseur pour un objet dépendent alors des autres objets. On distingue deux cas possibles [Rothkopf, 1998 ; Leyton-Brown *et al.*, 2000] :

1. le cas où les objets sont complémentaires. Par exemple un acheteur peut évaluer une unité centrale au prix x et un écran au prix y, mais il évalue la paire unité centrale et écran au prix $z \succ x + y$. Ici, les objets sont complémentaires.
2. Le cas où les objets sont substituts l'un de lŠautre : le cas où un acheteur peut évaluer une seule copie d'un livre à un prix x mais deux copies du même livre à un prix $z \prec 2x$. Dans ce cas, les deux objets sont substituts l'un de l'autre.

[1] http ://wireless.fcc.gov/auctions.

2.3.3 Modélisation du problème

Le problème de la détermination du gagnant dans les enchères combinatoires peut être énoncé comme suit :

Considérons un ensemble de m objets, M= $\{1, 2 \ldots m\}$ à vendre aux enchères et un ensemble de n offres d'achat N= $\{1, 2 \ldots n\}$. Chaque offre j est un tuple $<S_j, P_j>$, où S_j ($S_j \sqsubseteq M$) est l'ensemble d'objets couverts par l'offre j, P_j étant son prix ($P_j \succeq 0$).

Soit $a_{(m \times n)}$ une matrice binaire de m lignes et n colonnes tel que a_{ij}=1 si l'objet $i \in S_j$, 0 autrement.

De plus, soient les variables de décision x_j, x_j=1 si l'offre j est acceptée (une offre gagnante), 0 autrement (une offre perdante).

Le problème de la détermination du gagnant consiste à trouver les offres gagnantes qui maximisent le gain du vendeur sous la contrainte que chaque objet ne peut être affecté qu'à un seul acheteur. Le PDG peut être formalisé comme suit [Sandholm, 2006 ; Guo *et al.*, 2006] :

$$\textbf{Maximiser} \quad \sum_{j \in N} P_j.x_j \tag{2.1}$$

$$\textbf{Sous les contraintes} \ : \ \sum_{j \in N} a_{ij}.x_j \leq 1, \ \ i \in M \tag{2.2}$$

$$N = \{1, 2, \ldots, n\}, \ M = \{1, 2, \ldots, m\}, \ x_j \in \{0, 1\} \tag{2.3}$$

Où la fonction objectif (1) permet de maximiser le gain du vendeur calculé par la somme des prix des offres gagnantes et les contraintes interdisent qu'un objet figure dans deux offres différentes.

Exemple 6 *Exemple illustratif :*

Cet exemple permettra de mieux comprendre les notations utilisées dans la modélisation du problème PDG. Considérons un ensemble de cinq objets $\{1, 2, 3, 4, 5\}$ *à vendre aux enchères et quatre offres d'achat. Chaque offre est représentée par un couple* (S_j, P_j) *où* P_j *indique le prix que l'enchérisseur propose à payer pour son offre* j *contenant l'ensemble d' objets* S_j.

Soient les offres d'achat suivantes :

- *Offre 1 :* $\{(1, 2), 350\}$
- *Offre 2 :* $\{(1, 2, 3), 300\}$

- *Offre 3 :* $\{(3,4,5),400\}$
- *Offre 4 :* $\{(4,5),200\}$

L'offre 1 par exemple contient un ensemble de deux objets (1, 2) qui vaut 350. L'offre 3 vaut 400 pour un ensemble de trois objets (3,4,5).

Les offres 1 et 3 peuvent constituer une allocation gagnante maximisant le gain du vendeur et dont le prix total de la vente vaut 750.

2.3.4 Les enchères combinées et ses utilisations

Les enchères combinatoires permettent une allocation meilleure des objets selon les besoins spécifiques des enchiresseurs. Néanmoins, la difficulté de modéliser et de formaliser ce type d'enchère est apparaît dans le nombre de combinaisons d'objets possible. Pour cela, ce types d'enchères n'a été utilisé que rarement, dans des applications bien spécifiques [Nisan, 2000]. Les enchères de la FCC sont un exemple réel de cette catégorie d'enchères combinatoires.

Les premiers travaux pour implémenter les enchères combinatoires étaient effectués en 1982 par Rassenti, Smith et Bulfin. Ils ont développé un mécanisme pour l'allocation des tranches temporelles dans les aéroports entre les différentes compagnies aériennes.

D'autres implémentations des enchères combinatoires à été effectuées à l'instar du mécanisme de sélection adaptative par l'utilisateură(AUSM) introduit par *Banks* et ses collègues [Banks *et al.*, 1989]. Le mécanisme AUSM est un mécanisme décentralisé, à offres continues communiquées via une sorte de babillard électronique. Cette implémentation permet la soumission d'offres sur des paquets d'objets choisis par les participants.

2.3.5 Travaux antérieurs

Plusieurs méthodes ont été proposées pour résoudre le problème de la détermination du gagnant [Sandholm, 2006]. Ces méthodes peuvent être divisées en deux catégories : les méthodes complètes et les méthodes incomplètes.

Dans ce qui suit, nous donnons un aperçu sur quelques méthodes de résolution du *PDG*.

Sandholm a développé plusieurs algorithmes pour trouver la solution optimale pour le *PDG*. Parmi eux, nous citons : Branch-on-Items (BoI) [Sandholm et Suri, 2000], Branch-on-Bids (BoB)[Sandholm et Suri, 2000], et (CABoB) [Sandholm *et al.*, 2001]. CASS (Combinatorial Auction Structural Search) est un algorithme Branch-and-Bound pour le PDG proposé par [Fujishima *et al.*, 1999].

Dans [Leyton-Brown *et al.*, 2000a], les auteurs ont proposé CAMUS (Combinatorial Auctions Multi-Unit Search). [Rothkopf *et al.*, 1998] ont proposé une approche de programmation dynamique. Nisan [Nisan, 2000] a proposé une méthode de programmation linéaire. [Anderson *et al.*, 2000] ont développé un autre algorithme exact basé sur la programmation en nombre entier. Hollande et O'sullivan ont employé la programmation par contrainte pour résoudre une enchère combinatoire particulière de Vickrey [Hollande et O'sullivan, 2004].

D'autres part, quelques méthodes incomplètes ont été étudiées pour le PDG. Nous citons : le recuit simulé hybridé avec un Branch-and- Bound [Guo *et al.*, 2004 ; Guo *et al.*, 2006] et la recherche locale Casanova [Hoos et Boutilier, 2000].

2.3.5.1 Casanova

Casanova est une méthode de recherche locale proposée par Hoos et Boutilier [Hoos et Boutilier, 2000]. La méthode se résume comme suit : l'algorithme démarre d'une allocation vide ou toutes les offres sont considérées initialement non satisfaites. Ensuite, le processus de recherche sélectionne à chaque itération une offre non satisfaite à inclure dans l'allocation et enlève toute offre incompatible pouvant se produire dans l'allocation courante quand de nouvelles offres sont considérées (deviennent satisfaites). A chacun étape, une offre est sélectionnée comme suit :

1. Avec une probabilité wp (*Walk probability*) une offre non satisfaite est sélectionnée aléatoirement.
2. Avec une probabilité $1 - wp$, les offres sont classées selon leur profit qui correspond au prix de l'offre divisé par le nombre d'objets couverts par l'offre. Ensuite et avec une probabilité np (*Novelty probability*) la meilleure offre, B_1 ayant le plus grand profit est sélectionnée pour être incluse dans l'allocation courante. Autrement et avec une probabilité $1-np$, la deuxième meilleure offre B_2 est choisie pour inclusion dans l'allocation courante.

Le processus de recherche est réitéré pour un certain nombre d'itérations. Ce processus est similaire à celui d'adaptNovelty, la méthode de recherche locale proposée par le même auteur pour résoudre le problème de la satisfiabilité [Hoos, 2002].

2.3.5.2 Recuit Simulé hybride

Récemment, [Guo *et al.*, 2006] ont proposé une heuristique de recuit simulé combinée avec un algorithme de type Branch-and-Bound. Les résults sont très encourageants et la nouvelle méthode surpasse clairement Casanova. Nous notons que l'heuristique proposée fait appel tout d'abord à un prétraitement pour

exclure les offres qui peuvent mener aux solutions suboptimales, ce qui permettra d'améliorer le temps de réponse du processus de recherche.

Le processus de recherche consiste en trois composantes. La première composante est un algorithme Branch-and-Bound appliqué aux sous ensemble d'objets des offres de l'allocation courante. La deuxième composante est une recherche locale utilisée pour sélectionner les meilleures offres à inclure dans l'allocation courante. Une fonction de pénalité est utilisée pour éliminer les offres incompatibles. La troisième composante est un mouvement (échange) qui permet de sélectionner d'une manière aléatoire l'offre à considérer dans l'allocation.

La méthode démarre d'une allocation vide. Ensuite la phase de prétraitement est lancée pour exclure les offres qui peuvent mener aux solutions sub-optimales.

L'algorithme Branch-and-Bound est exécuté avec une probabilité p_1 =0.2, la recherche locale est lancée avec une probabilité p_2=0.7, et le mouvement échange est exécuté avec une probabilité 1-p_1-p_2.

Le processus de recherche qui consiste en un algorithme de Branch-and-Bound, une recherche locale et un mouvement d'échange est répété pour certain nombre d'itérations.

L'algorithme général de l'approche locale est donné dans l'algorithme7.

D'après l'algorithme7, on part d'une allocation initiale S et d'une température T. Ensuite on génère une nouvelle allocation voisine S'. La solution voisine est générée selon trois méthodes :

1. Avec une probabilité p_1, S' est générée par l'algorithme de Branch-and-bound(S) ;

2. Avec une probabilité p_2, S' est générée par l'algorithme de recherche locale(S) ;

3. Avec une probabilité 1-p_1-p_2, S' est générée par le mouvement échange(S) ;

Si l'allocation voisine augmente la fonction coût du problème, elle est acceptée, si au contraire elle provoque une diminution δ de la fonction coût, elle est acceptée tout de même avec la probabilité p= $e^{\delta/T}$ ou e dénote la fonction mathématique exponentielle. En appliquant itérativement cette règle d'acceptation, on engendre une séquence d'allocations. Après qu'un certain nombre d'itérations a été atteint, on réduit la température T et on effectue une nouvelle série d'itérations à cette nouvelle température.

Algorithm 7 : Le recuit simulé hybride pour le PDG.

1: // S est l'allocation (solution) qu'on cherche et $best\_value$ est son coût.
2: // S' est l'allocation voisine
3: // T_{max} est la température initiale
4: // α est le taux de décroissance de la température.
5: $S = \phi$; $best\_value = 0$; $T = T_{max}$;
6: pré-traitement() ;
7: **Tant que** (le nombre maximum des itérations n'est pas atteint)
8: **Faire**
// Le système se trouve à la température T et possède l'énergie
// (valeur de la fonction coût relative à l'allocation courante) $value$
9: Avec une probabilité p_1, S' = Branch-and-bound(S) ;
10: Avec une probabilité p_2, S' = recherche locale(S) ;
11: Avec une probabilité 1-p_1-p_2, S' = échange(S) ;
12: $\delta = value\ (S')$ - $value\ (S)$;
13: **Si** $\delta \succeq 0$ **alors** $S = S'$
14: **Sinon** $p = e^{\delta/T}$;
15: Avec probabilité p, $S = S'$;
16: **FinSi**
17: **Si** $value\ (S) > best\_value$ **alors** $best\_value = value\ (S)$;
18: **FinSi**
19: $T = T \times \alpha$;
20: **Fait ;**

2.4 Conclusion

Le PDG est un problème complexe, une solution basée sur la recherche taboue est proposée dans le chapitre 4. Le chapitre suivant propose plusieurs méthodes métaheuristiques pour le problème *MAX-SAT*.

Parite 1. Approches Évolutionnaires pour le problème MAX-SAT

Les travaux présentés dans cette première partie ont fait l'objet de publications dans les revues :

- Journal of Mathematical Modelling and Algorithms [Boughaci *et al.*, 2008],
- Journal of Research on Computing Sciences (RCS) [Boughaci *et al.*, 2005],
- International Transaction on Computer Science and Engineering [Boughaci et Drias (c), 2005],
- International Journal of Information Technology [Boughaci et Drias (a), 2004].

et les conférences :

- IICAI'07[][Boughaci *et al.*, 2007a],
- COSI'07 [Boughaci *et al.*, 2007b],
- RJCIA'07 [Boughaci *et al.*, 2007c],
- META'06 [Boughaci *et al.*, 2006a],
- Roadef'06 [Boughaci *et al.*2006b],
- WEA'05 [Boughaci et Drias a, 2005a],
- MICAI'05 [Boughaci *et al.*, 2005c],
- ICCI'04 [Boughaci et Drias, 2004d],
- ACM SAC'04 [Boughaci et Drias, 2004b],
- IEEE CIS'04 [Boughaci *et al.*2004].
- IEEE GCC'04 [Boughaci et Drias 2004c]

Chapitre 3

Approches évolutionnaires pour le problème MAX-SAT

3.1 Introduction

Dans ce chapitre, nous présentons deux approches évolutionnaires pour le problème MAX-SAT. Tout d'abord, nous proposons une nouvelle stratégie de sélection basée sur la diversité et la qualité pour choisir une collection de solutions appelées solutions de référence. Ces dernières vont participer à la phase de reproduction et donner une descendance. Ensuite, nous utilisons un opérateur de combinaison spécifique au problème MAX-SAT pour générer de nouvelles solutions qui sont améliorées par une recherche locale stochastique (SLS). Les trois composantes proposées sont incorporées, premièrement, dans un algorithme génétique pour améliorer sa performance. Ensuite, une variante de recherhce dispersée, utilisant les trois composantes déjà citées, est étudiée pour le problème MAX-SAT. Plusieurs expérimentations numériques sont réalisées sur des instances MAX-SAT dans le but de tester et de prouver l'efficacité de nos approches.

3.2 Recherche locale stochastique (SLS)

La technique de recherche locale que nous proposons est basée sur les concepts bien connus de Walksat [Selman *et al.*, 1994]. A chaque itération, la variable à inverser est choisie selon l'un des trois critères suivants où dp et wp sont deux probabilités à utiliser :

- Le premier critère consiste à choisir la variable à inverser d'une manière aléatoire avec une probabilité fixe $wp > 0$.
- Le deuxième critère consiste à choisir aléatoirement la variable à inverser d'une clause non satisfaite cl choisie d'une manière aléatoire. L'étape est

exécutée avec une probabilité fixe $(dp - wp)/(1 - wp)$.
- Le troisième critère consiste à choisir la meilleure variable (celle maximisant le nombre de clauses satisfaites une fois renversée) pour être renversée. Cette étape correspond à un GSAT standard.

Le processus de SLS est esquissé dans l'algorithme8.

Algorithm 8 : La méthode d'amélioration de SLS.

Entrée : la formule MAX-SAT , l'individu V, maxflips, dp, wp
Sortie : un individu amélioré V
r = un nombre aléatoire entre 0 et 1
Pour flip=1 **à** maxflips
faire
Si ($r \prec wp$) **alors** *(étape 1 :).* $flipvar$ = choisir aléatoirement une variable
sinon *(étape2 :)* **Si** ($r \prec dp$) **alors**
cl= choisir aléatoirement une clause non satisfaite
$flipvar$ = choisir aléatoirement une variable de cl
sinon *(étape3).*
$flipvar$ = choisir la meilleure variable ;
Finsi
Finsi
$V = V$ avec $flipvar$ renversée.
Finpour

3.3 Un algorithme génétique spécifique au problème MAX-SAT

L'algorithme génétique spécifique au problème MAX-SAT (SGAV) que nous proposons ici démarre d'une population P d'individus créée aléatoirement. Ensuite, une collection B d'individus est choisie parmi la population courante pour participer à la phase de reproduction et produire d'autres individus.

La collection B contient, d'une part, un certain nombre B_1 de bons individus choisis selon leur fitness. D'autre part, un certain nombre B_2 d'individus sont tirés de la population restante $P - B$ et rajoutés à la collection B pour la compléter. Les B_2 individus sont les individus les plus éloignés des meilleurs individus de B, ils sont appelés individus diversifiés. La diversité d'un individu est mesurée par la distance de Hamming. La nouvelle stratégie de sélection que nous proposons aide l'algorithme à maintenir à chaque génération une population diversifiée et de meilleure qualité ce qui réduit principalement la convergence prématurée qui est une caractéristique inhérente dans l'algorithme génétique classique.

Après avoir choisi un ensemble d'individus bons et diversifiés, la phase de reproduction est lancée. Une fois que deux parents sont choisis, leurs chromosomes sont combinés pour construire un enfant. La combinaison utilise un opérateur spécifique au problème MAX-SAT.

Afin de localiser plus efficacement une solution, la phase de mutation est remplacée par une recherche locale stochastique. À la différence de l'algorithme génétique classique qui favorise toujours les meilleurs, notre approche gère la population selon deux critères : la qualité et la diversité. Si le nouvel individu améliore la fitness de B alors on l'ajoute aux meilleures solutions B_1 et la solution la plus mauvaise de B est enlevée. Autrement, si le nouvel individu améliore la diversité de la collection courante, alors l'individu le moins diversifié dans la collection est remplacé par le nouveau.

Le processus génétique est réitéré pour un certain nombre de générations fixé d'une manière empirique.

Les composants de notre approche sont détaillés dans ce qui suit :

3.3.1 Représentation d'un individu

Un individu décode une solution. Cette dernière est représentée par une chaîne binaire X (un vecteur X de taille n), dont chaque composant X_i reçoit la valeur 0 (fausse) ou 1 (vrai). Elle représente une affectation de valeurs de vérité aux n variables.

3.3.2 Fonction d'évaluation

La qualité d'une solution est mesurée par le nombre de clauses satisfaites par la solution.

3.3.3 Fonction de diversité

La mesure de distance dépend de la représentation de la solution. Dans notre problème où la solution est représentée par une chaîne binaire, nous utilisons la distance de Hamming pour exprimer le nombre de bits différents entre deux solutions.

La diversité d'une solution est définie par la distance entre cette solution et l'ensemble de solutions de la collection B. Elle correspond à la valeur minimum des distances de Hamming entre la solution et les solutions du B.

Exemples :

- La distance de Hamming entre X= 0100 et Y = 0011 est égale à 3.
- La distance de Hamming entre X= 0100 et l'ensemble de deux solutions (Y = 0011 et Z=0101) est égale à 1 (elle est égale au minimum entre la distance entre X et Y et la distance entre X et Z).

3.3.4 Stratégie de choix de collection

La nouvelle stratégie de sélection que nous proposons constitue un moyen pour réaliser une interaction efficace entre la diversification et l'amélioration. Un ensemble B de solutions diversifiées et de haute qualité est sélectionné pour construire une nouvelle génération. La construction de l'ensemble B est faite selon les étapes ci-dessous :

- Sélectionner les B_1 meilleures solutions de P pour pouvoir déduire les solutions de haute qualité choisies selon leur fonction d'évaluation.
- Pour chaque solution V appartenant à $P - B$, calculer la diversité de V.
- Sélectionner B_2 solutions de $P - B$ ayant une grande diversité.
- Rajouter les B_2 solutions sélectionnées à l'ensemble B pour le compléter.

On obtient une collection B de B_1 meilleures solutions et B_2 diverses solutions.

3.3.5 Opérateur de combinaison

Définir un bon opérateur de combinaison est une étape délicate dans l'implémentation de notre variante. L'opérateur de combinaison que nous utilisons dépend de la *fonction objectif* utilisée ainsi que de la représentation binaire de la solution. Notre objectif est d'essayer de réduire au minimum le nombre de clauses falsifiées. L'opérateur de combinaison prend deux parents et produit un nouvel enfant (appelé Child).

Algorithm 9 : L'opérateur de combinaison.

Entrée : deux parents, $Parent1$ et $Parent2$
Sortie : un enfant, $Child$
(* initialement toutes les variables dans l'enfant sont non valuées*)
Pour toute (clauses non satisfaites et falsifiées par Parent1 et Parent2)
Faire
Soit, cl une clause falsifiée
Sélectionner de cl la meilleure variable pour la fixer dans l'enfant
Fixer la variable choisie de manière à satisfaire la clause cl
Fait
Pour chaque (variable non valuée var, dans $Child$)
Faire
r = un nombre aléatoire entre 0 et 1
Si ($r < 0.5$) **alors** $Child_{var} = Parent1_{var}$
sinon $Child_{var} = Parent2_{var}$
Finsi
Fait

Pour construire l'enfant, l'opérateur, premièrement, examine parmi les clauses non satisfaites, celles falsifiées simultanément par les deux parents, puis choisit de la clause courante falsifiée une meilleure variable (c.-à-d. celle réduisant au minimum le nombre de clauses non satisfaites une fois renversée) à fixer dans l'enfant pour satisfaire la clause (rendre la clause vraie).

Ensuite et après avoir examiné toutes les clauses non satisfaites, nous obtenons une affectation partielle qui peut être complétée en appliquant un opérateur de croisement uniforme. L'opérateur de croisement uniforme décide quel parent contribuera à la production de l'enfant, les bits sont copiés de premier ou de deuxième parent d'une manière aléatoire selon une certaine probabilité.

L'opérateur de combinaison est donné dans l'algorithme9.

3.3.6 L'algorithme génétique spécifique au problème MAX-SAT

L'algorithme génétique spécifique au problème MAX-SAT est donné dans l'algorithme10.

Algorithm 10 : L'algorithme génétique spécifique pour MAX-SAT.

1: **Entrée** : une instance MAX-SAT.
2: **Sortie** : une affectation maximisant le nombre de clauses satisfaites.
3: Générer aléatoirement une population initiale P ;
4: Sélectionner une collection d'individus meilleurs et dispersés B de P
5: **Pour** I=1 au nombre maximum d'essais (Maxtries)
6: **Faire**
7: **Tant que** (le nombre maximum de générations n'est pas atteint)
8: **Faire**
9: **Répéter**
10: Sélectionner deux parents d'individus de B ;
11: Générer aléatoirement un nombre r entre 0 et 1 ;
12: **Si** ($r \prec$ le taux de croisement) **alors** Appliquer l'opérateur de combinaison aux parents pour obtenir un nouvel enfant V ;
13: Appliquer SLS sur V ;
14: **Si** (V améliore la qualité de B) **alors** Ajouter V aux B_1 meilleurs individus ;
15: Enlever de B le mauvais individu ;
16: **Sinon Si** (V améliore la diversité de B) **alors** Ajouter V aux B_2 individus dispersés ;
17: Enlever de B l'individu le moins dispersé ;
18: **Finsi**
19: **Finsi**
20: **Finsi**
21: Jusqu'au (toutes les combinaisons des parents sont examinées)
22: **Fait**
23: Reproduire une nouvelle population et sélectionner une nouvelle collection B ;
24: **Fait**
25: Retourner le meilleur individu trouvé ainsi que sa fitness.

3.4 Une variante de recherche dispersée pour MAX-SAT

La recherche dispersée est une méta-heuristique à stratégies d'évolution. Elle intègre les idées majeures des méthodes évolutionnistes (population de solutions, recombinaison de solutions).

Un algorithme génétique [Goldberg 1989] est aussi une approche évolutionniste, il présente plusieurs inconvénients liés à sa structure classique [Talbi, 1995 ; Rana et Whitley, 1998]. Nous citons :

- L'incapacité du processus de recherche d'exploiter au mieux certaines régions particulières. Ceci est dû au fait que les opérateurs génétiques de mutation et de croisement agissent tous les deux d'une manière aveugle.
- La convergence prématurée après un certain nombre d'itérations dû au fait que les individus deviennent de plus en plus identiques au fur et à mesure que le nombre de générations augmente.

Dans le but de pallier ces principaux inconvénients, la recherche dispersée s'appuie sur un principe qui consiste à générer la population de départ à l'aide d'une stratégie de diversification.

La variante de recherche dispersée que nous avons proposée (SSV), démarre d'une population initiale variée. Comme dans l'algorithme génétique proposé (SGAV), le processus d'évolution dans SSV se fait par deux mécanismes : la nouvelle stratégie de sélection et la reproduction.

La SSV présente des similitudes avec SGAV déjà présenté en *paragraph* 3.3 et qui utilise la nouvelle stratégie de sélection basée sur la fitness et la diversité pour choisir les individus qui vont survivre et qui serviront à produire d'autres générations. La phase de reproduction qui utilise l'opérateur de combinaison spécifique au problème MAX-SAT pour générer de nouveaux enfants qui sont améliorés par la recherche locale stochastique (SLS). La SSV se distingue par sa méthode de diversification utilisée pour générée la population de départ et sa manière d'appliquer l'opérateur de combinaison. Dans SGAV l'application de l'opérateur de combinaison est tributaire d'une probabilité représentant le taux de croisement alors que dans SSV la combinaison est réitérée pour un certain nombre de combinaisons fixé empiriquement.

Notre SSV présente des similitudes avec SS proposée par [Drias et Khabzaoui, 2001] et qui se base sur la qualité et la diversité pour explorer l'espace de recherche. Notre SSV se distingue par sa recherche locale basée sur Walksat et son opérateur de croisement spécifique au problème *MAX-SAT* qui permet de restaurer la consistance des clauses falsifiées simultanément par les parents et essaie de minimiser le nombre de clauses falsifiées par le fils résultant des parents.

D'autre part, SS[Drias et Khabzaoui, 2001] traite la classe *jnh* du problème *MAX-SAT* alors que notre SSV étudie plusieurs classes du problème *MAX-SAT* à savoir : *Parity, ais, Encoded 2-colouring, Dubois, Jnh, Pigeon hole, Mat, difp, glassy, Color, logistics, hgen, lran, MAX-2-SAT et MAX-3-SAT.*

3.4.1 Population initiale

La population initiale est l'ensemble des configurations qui représentent des solutions du problème. Contrairement aux algorithmes génétiques où les solutions de départ sont générées aléatoirement, la recherche dispersée utilise une stratégie

	1	1	1	1	1	h=1
	1	0	1	0	1	h=2
Type 1	1	0	0	1	0	h=3
	1	0	0	0	1	h=4
	0	0	0	0	0	
Type 2	0	1	0	1	0	
	0	1	1	0	1	
	0	1	1	1	0	

FIG. 3.1 – Exemple d'une population de taille 8.

de diversification pour avoir une population variée qui permettra d'explorer des zones diverses de l'espace de recherche. A chaque solution un coût correspondant à la *fonction objectif* est associé.

La méthode de diversification utilisée dans la génération de la population initiale, démarre d'une solution initiale X générée aléatoirement. Ensuite, elle crée deux types de solutions Y et Z en utilisant les formules suivantes :

Solutions de type 1
Y_1 = 1- X_1
$Y_{1+k\times h} = 1 - X_{1+k\times h}$, $k = 1,...k^*, \quad k^* \leq n/h, \quad 1 \leq h < n.$
Solutions de type 2 : Z est le complément à 1 de Y.
Exemple :
Soient $n = 5$ et $X = 0, 0, 0, 0, 0$. On cherche à générer une population P de taille 8 en utilisant les formules ci-dessus. On obtient la Population donnée dans la Figure 3.1.

3.4.2 L'algorithme de la recherche dispersée pour MAX-SAT

L'algorithme de la variante de recherche dispersée pour le problème MAX-SAT est donné dans l'algorithme**??**.

Algorithm 11 : L'algorithme de la recherche dispersée pour MAX-SAT.

Entrée : une instance MAX-SAT.
Sortie : une affectation booléenne maximisant le nombre de clauses satisfaites.
// maxgen est le nombre maximum de générations
// maxcombin est le nombre maximum de combinaisons
Générer une population initiale P en utilisant la méthode de diversification ;
That que (gen $<$ maxgen)
Faire
cpt = 0 ;
Sélectionner un ensemble de référence B de P ;
Tant que ((une nouvelle solution est introduite dans B) **et** (cpt$<$ maxcombin))
Faire
cpt = cpt +1 ;
Pour chaque deux solutions de B, utiliser l'opérateur de combinaison pour produire une nouvelle solution V ;
Améliorer la nouvelle solution en utilisant la SLS ;
Si (V améliore la qualité de B) **alors** Ajouter V aux B_1 meilleures solutions ;
Enlever de B la mauvaise solution ;
Sinon Si (V améliore la diversité de B) **alors** Ajouter V aux B_2 solutions dispersées ;
Enlever de B la solution la moins dispersée ;
Finsi
Finsi
Fait
Régénérer une nouvelle population P ;
Fait
Retourner la meilleure solution trouvée.

3.5 Résultats expérimentaux

Dans le but de valider nos approches, des tests ont été effectués sur différents problèmes MAX-SAT. Dans nos expériences, nous avons considéré des instances de *MAX-2-SAT* et *MAX-3-SAT* produites par Borchers et al[1]. En plus de ces instances, nous avons considéré plusieurs instances de la librairie de *SATLIB*[2].

Nos algorithmes ont été implémentés en C sous Linux et sous la machine AMD Duron 800 MHz avec 512 MO de RAM.

[1]http ://www.nmt.edu/ borchers/maxsat.html
[2]http ://www.cs.ubc.ca/ hoos/SATLIB/benchm.html.

Cette section énumère les solutions trouvées par nos variantes (SGAV[3], SSV[4]) appliquées au problème MAX-SAT.

Afin de bien évaluer la qualité des solutions trouvées, nous avons implémenté les algorithmes de comparaison suivants :

- GA : est un algorithme génétique standard utilisant un croisement uni-point, une mutation standard et sans recherche locale.
- GAV : est une variante d'un algorithme génétique utilisant la nouvelle stratégie de sélection que nous avons proposée, une recherche locale stochastique à la place de mutation et le croisement uniforme.

Dans le but de tester et de prouver l'efficacité de nos approches, une étude comparative avec quelques algorithmes de l'état de l'art concernant MAX-SAT à savoir Walksat, FlipGA et GASAT est établie dans ce chapitre.

3.5.1 Réglage des paramètres

L'efficacité d'une approche dépend largement d'un bon ajustement de ses paramètres. Ceux-ci sont fixés suivant des expérimentations effectuées et les valeurs prises sont celles pour lesquelles il existe un compromis entre la qualité de la solution obtenue par l'approche et le temps d'exécution de l'algorithme.

Pour nos approches SGAV et SSV, la taille de la population, la taille de l'ensemble B, le nombre d'itérations de la recherche locale SLS (maxflips), les deux paramètres de la recherche locale (dp et wp), le nombre d'exécutions de l'algorithme, et le nombre maximum de générations sont les paramètres à ajuster.

Suite à plusieurs tests, et en tenant compte du modèle de problèmes à considérer, les paramètres des algorithmes GA, SSV, SGAV et GAV sont fixés comme suit :

Le Modèle 1 contient des instances aléatoires de *MAX-2-SAT* et *MAX-3-SAT*. Les résultats trouvés pour ces problèmes sont donnés dans les Tables 3.1 jusqu'à 3.6 où les différents paramètres utilisés sont :

- Les paramètres de GA classique sont : le nombre maximum de générations=1000, la taille de la population =100, le taux de croisement (uni-point) = 0.6 et le taux de mutation= 0.1. L'algorithme GA a été relancé 10 fois.
- Les paramètres de SSV sont : le nombre maximum de générations=5, la taille de la population=50, la taille de l'ensemble de référence B est (B_1= 3, B_2= 3), et maxcombin=5. Le nombre des itérations (maxflips) de SLS

[3]SGAV : est un algorithme génétique utilisant le croisement spécifique, la nouvelle stratégie de sélection et la recherche locale SLS que nous avons proposés.

[4]SSV : est une variante de recherche dispersée utilisant le croisement spécifique, la nouvelle stratégie de sélection et la recherche locale SLS.

est fixé à vars × 10, wp=0.3 et dp=0.6. La SSV a été lancée 1 fois car la population de départ est toujours la même.
- Les paramètres de GAV sont : Maxtries= 10, le nombre maximum de générations=50, la taille de population =50, la taille de collection B (B_1= 3, B_2= 3), le taux de croisement uniforme= 0.6. Maxflips de la recherche locale stochastique SLS est fixé au nombre de variables de l'instance MAX-SAT, wp=0.3 et dp=0.6.
- Les paramètres de SGAV sont : Maxtries= 10, le nombre maximum de générations=50, la taille de population =50, la taille de collection B (B_1= 3, B_2= 3), le taux de croisement spécifique = 0.6. Maxflips de la recherche locale stochastique SLS est fixé au nombre de variables de l'instance MAX-SAT, wp=0.3 et dp=0.6.

Le Modèle 2 : contient des instances de *SATLIB*. Plusieurs classes de problèmes sont considérées à savoir : *Parity, ais, Encoded 2-colouring, Dubois, jnh, AIM, Pigeon hole, Mat, difp, glassy, Color, logistics, hgen, et lran*. Plus de détails sur ces classes de problèmes peuvent être trouvés sur le site Web de la librairie de SATLIB[5].

Pour certaines instances (*lran, Parirty, Mat, difp, hgen et Color*), nous avons augmenté la taille de la population, la taille de l'ensemble B ainsi que le nombre d'itérations de la recherche locale pour avoir de bons résultats. Les Tables 3.7 jusqu'au 3.13 donnent quelques résultats trouvés où les paramètres sont fixés comme suit :
- Les paramètres de GA classique sont : le nombre maximum de générations=1000, le taux de croisement (uni-point) = 0.6 et le taux de mutation= 0.1. L'algorithme GA a été relancé 10 fois.
- Les paramètres de SSV sont : le nombre maximum de générations=5, la taille de l'ensemble de référence B est (B_1= 5, B_2= 5), et maxcombin=5. Le nombre des itérations (maxflips) de SLS est fixé à vars×50, wp=0.3 et dp=0.6. L'algorithme SSV a été lancé 1 fois.
- Les paramètres de GAV sont : Maxtries= 10, le nombre maximum de générations =100, la taille de collection B (B_1= 5, B_2=5), le taux de croisement uniforme= 0.6. Maxflips de la recherche locale stochastique SLS est fixé au nombre de variables de l'instance MAX-SAT ×50, wp=0.3 et dp=0.6.
- Les paramètres de SGAV sont : Maxtries= 10, le nombre maximum de générations=10, la taille de collection B (B_1= 5, B_2= 5), le taux de croisement spécifique = 0.6. Maxflips de la recherche locale stochastique SLS est fixé au nombre de variables de l'instance MAX-SAT ×50, wp=0.3 et dp=0.6.

[5]http ://www.cs.ubc.ca/ hoos/SATLIB/benchm.html.

Pour les classes *ais, Encoded 2-colouring, Dubois, jnh, AIM et Pigeon hole*, nous avons utilisé une population de 50 solutions tandis que pour les autres classes la taille de population est fixée à 100 solutions.

3.5.2 Résultats

Les résultats obtenus sont donnés dans les tables [6] 3.1, ...3.13. Pour chaque méthode, nous donnons le meilleur nombre de clauses violées dans la meilleure solution trouvée (*sol*) et le temps de calcul en seconde nécessaire pour atteindre cette solution (*time*). U correspond au nombre exact de clauses violées dans l'instance MAX-SAT.

Nous tenons à préciser que le temps d'exécution de chaque algorithme (GAV, SGAV et GA) donné dans les tables correspond au temps total de 10 exécutions.

3.5.2.1 Comparaison avec GA classique

- Instances MAX-2-SAT et MAX-3-SAT

Les Tables 3.1 et 3.2 montrent que les trois variantes (SSV, GAV et SGAV) trouvent l'optimum pour presque toutes les instances *MAX-2-SAT* et *MAX-3-SAT*. Elles sont nettement meilleures que l'algorithme génétique traditionnel qui produit des résultats de qualité médiocre. Ceci est du à sa convergence prématurée.

- Instances de SATLIB

A partir des Tables 3.3, ...3.8, on remarque que les algorithmes GAV, SSV et SGAV trouvent l'optimum pour certaines instances, et pour les autres l'écart à l'optimum n'est pas très grand. Les résultats respectifs sont meilleurs que ceux fournis par l'algorithme GA, concernant la qualité des solutions.

La version traditionnelle de l'algorithme génétique (GA) échoue encore à trouver une solution aux différentes classes de problèmes de MAX - SAT. Ceci confirme la convergence prématurée de l'algorithme (GA). L'efficacité de notre nouvelle approche est expliquée par la bonne combinaison entre la diversification et l'intensification ce qui mène l'algorithme à explorer efficacement l'espace de recherche et localiser une bonne solution.

[6] *vars* (respectivement *cls*) est le nombre de variables (respectivement de clauses) de l'instance MAX-SAT. SAT : *Y* signifie que l'instance est satisfaisable. *N* signifie que l'instance est non satisfaisable.

TAB. 3.1 – Les résultats trouvés par SSV, GAV, SGAV et GA sur les instances *MAX-2-SAT*.

Instances.cnf (#Vars/#Cls)	#U	**SSV** *sol*	time	**GAV** *sol*	time	**SGAV** *sol*	time	GA *sol*	time
p100(50/100)	4	**4**	0.30	**4**	**0.01**	**4**	0.04	15	0.42
p150(50/150)	8	**8**	0.32	**8**	**0.02**	**8**	0.04	24	0.42
p200(50/200)	16	**16**	0.35	**16**	**0.01**	**16**	0.06	34	0.42
p2200(100/200)	5	**5**	0.61	**5**	**0.02**	**5**	0.09	34	0.6
p2300(100/300)	15	**15**	0.66	**15**	**0.04**	**15**	0.11	60	0.6
p2300(150/300)	4	**4**	0.94	**4**	**0.05**	**4**	0.15	57	0.8
p2400(100/400)	29	30	0.72	30	0.04	**29**	**0.13**	83	0.62
p2500(100/500)	44	45	0.78	**44**	**0.04**	**44**	0.16	104	0.62
p250(50/250)	22	**22**	0.38	**22**	**0.02**	**22**	0.07	45	0.42
p2600(100/600)	66	66	0.86	**65**	**0.06**	**65**	0.18	129	0.62
p2600(150/600)	38	39	1.14	39	0.07	**38**	**0.24**	125	0.81
p300(50/300)	32	**32**	0.40	**32**	**0.02**	**32**	0.07	61	0.41
p350(50/350)	41	**41**	0.43	**41**	**0.02**	**41**	0.08	71	0.41
p400(50/400)	45	**45**	0.45	**45**	**0.02**	**45**	0.09	81	0.41
p450(50/450)	63	**63**	0.48	**63**	**0.02**	**63**	0.11	93	0.42
p500(50/500)	66	**66**	0.51	**66**	**0.03**	**66**	0.11	102	0.42

TAB. 3.2 – Les résultats trouvés par SSV, GAV, SGAV et GA sur les instances *MAX-3-SAT*.

Instances.cnf (#Vars/#Cls)	#U	**SSV** *sol*	time	**GAV** *sol*	time	**SGAV** *sol*	time	GA *sol*	time
p3250(50/250)	2	**2**	0.43	**2**	**0.02**	**2**	0.08	18	0.42
p3300(50/300)	4	**4**	0.47	**4**	**0.02**	**4**	0.09	24	0.41
p3350(50/350)	8	**8**	**0.51**	9	**0.03**	**8**	0.10	27	0.42
p3400(50/400)	11	**11**	0.55	12	**0.03**	**11**	**0.12**	35	0.42
p3450(50/450)	15	**15**	0.58	**15**	**0.04**	**15**	0.13	38	0.42
p3500(100/500)	4	**5**	0.89	6	**0.05**	**5**	**0.18**	46	0.62
p3500(50/500)	15	**15**	0.62	**15**	**0.04**	**15**	0.15	46	0.42
p3550(100/550)	5	8	0.93	**6**	**0.06**	**6**	0.20	50	0.61
p3600(100/600)	8	**9**	1.00	**10**	**0.07**	**9**	**0.21**	53	0.62
p3675(150/675)	2	**4**	1.34	**4**	**0.09**	**4**	0.29	62	0.81

TAB. 3.3 – Les résultats trouvés par SSV, GAV, SGAV et GA sur les instances Dubois de SATLIB.

Instances.cnf (#Vars/#Cls)	#U	**SSV** *sol*	time	**GAV** *sol*	time	**SGAV** *sol*	time	GA *sol*	time
dubois20(60/160)	1	**1**	0.43	**1**	**0.02**	**1**	0.06	13	0.46
dubois21(63/168)	1	**1**	0.45	**1**	**0.03**	**1**	0.06	15	0.46
dubois22(66/176)	1	**1**	0.45	**1**	**0.02**	**1**	0.07	15	0.48
dubois23(69/184)	1	**1**	0.49	**1**	**0.02**	**1**	0.08	13	0.49
dubois24(72/192)	1	**1**	0.51	**1**	**0.03**	**1**	0.07	17	0.51
dubois25(75/200)	1	**1**	0.53	**1**	**0.02**	**1**	0.08	15	0.51
dubois26(78/208)	1	**1**	0.54	**1**	**0.03**	**1**	0.08	19	0.53
dubois27(81/216)	1	**1**	0.59	**1**	**0.03**	**1**	0.09	17	0.54
dubois28(84/224)	1	**1**	0.63	**1**	**0.03**	**1**	0.09	21	0.55
dubois29(87/232)	1	**1**	0.61	**1**	**0.03**	**1**	0.09	21	0.56
dubois30(90/240)	1	**1**	0.64	**1**	**0.03**	**1**	0.10	21	0.57
dubois50(150/400)	1	**1**	1.09	3	0.06	**1**	**0.19**	37	0.81

TAB. 3.4 – Les résultats trouvés par SSV, GAV, SGAV et GA sur instances *Encoded 2-colouring*.

Instances.cnf (#Vars/#Cls)	#U	**SSV** *sol*	time	**GAV** *sol*	time	**SGAV** *sol*	time	GA *sol*	time
pret150_25(150/400)	1	**1**	1.07	**1**	**0.06**	**1**	0.18	35	0.79
pret150_40(150/400)	1	**1**	1.06	**1**	**0.06**	**1**	0.18	37	0.8
pret150_60(150/400)	1	**1**	1.07	**1**	**0.06**	**1**	0.19	39	0.8
pret150_75(150/400)	1	**1**	1.07	**1**	**0.06**	**1**	0.19	39	0.8
pret60_25(60/160)	1	**1**	0.40	**1**	**0.03**	**1**	0.06	13	0.45
pret60_40(60/160)	1	**1**	0.40	**1**	**0.02**	**1**	0.05	13	0.45
pret60_60(60/160)	1	**1**	0.40	**1**	**0.01**	**1**	0.06	15	0.45
pret60_75(60/160)	1	**1**	0.41	**1**	**0.01**	**1**	0.06	13	0.46

TAB. 3.5 – Les résultats trouvés par SSV, GAV, SGAV et GA sur instances *AIM*.

Instances.cnf (#Vars/#Cls)	#U	**SSV** *sol*	time	**GAV** *sol*	time	**SGAV** *sol*	time	GA *sol*	time
aim100-1_6no(100/160)	1	**1**	0.63	**1**	**0.03**	**1**	0.09	11	0.60
aim100-1_6no4(100/160)	1	**1**	0.63	**1**	**0.03**	**1**	0.09	14	0.61
aim200-1_6no3(200/320)	1	**1**	1.34	**1**	**0.07**	**1**	0.23	40	1.00
aim200-1_6no4(200/320)	1	**1**	1.36	**1**	**0.08**	**1**	0.24	41	1.01
aim50-2_no2(50/100)	1	**1**	0.32	**1**	**0.01**	**1**	0.04	6	0.42
aim50-2_no3(50/100)	1	**1**	0.32	**1**	**0.01**	**1**	0.05	6	0.41

TAB. 3.6 – Les résultats trouvés par SSV, GAV, SGAV et GA sur instances *Pigeon hole problem*.

Instances.cnf (#Vars/#Cls)	#U	**SSV** *sol*	time	**GAV** *sol*	time	**SGAV** *sol*	time	GA *sol*	time
hole10(110/ 561)	1	**1**	0.83	**1**	**0.05**	**1**	0.17	81	0.65
hole6(42/133)	1	**1**	0.26	**1**	**0.01**	**1**	0.03	14	0.39
hole7(56/204)	1	**1**	0.37	**1**	**0.02**	**1**	0.05	24	0.45
hole8(72/297)	1	**1**	0.50	**1**	**0.03**	**1**	0.08	33	0.5
hole9(90/415)	1	**1**	0.64	**1**	**0.04**	**1**	0.11	57	0.59

TAB. 3.7 – Les résultats trouvés par SSV, GAV, SGAV et GA sur les instances *Parity-16* et *Lran*.

Instances.cnf (#Vars/#Cls)	#U	**SSV** *sol*	time	**GAV** *sol*	time	**SGAV** *sol*	time	GA *sol*	time
par16-1-c(317/1264)	0	4	10.70	13	2.02	**4**	43.63	142	1.69
par16-1(1015/3310)	0	11	75.67	20	13.94	**10**	349.01	499	5.12
par16-2-c(349/1392)	0	4	12.56	13	2.36	**4**	52.77	163	1.89
par16-2(1015/3374)	0	8	76.62	22	14.18	**5**	353.39	507	5.12
par16-3-c(334/1332)	0	5	11.57	14	2.18	**4**	48.55	159	1.80
par16-3(1015/3344)	0	12	76.07	20	14.03	**9**	351.59	509	5.10
par16-4-c(324/1292)	0	6	11.03	14	2.13	**5**	48.01	150	1.70
par16-4(1015/3324)	0	10	75.69	20	14.12	**8**	349.42	521	5.08
par16-5-c(341/1360)	0	6	11.97	13	2.31	**4**	50.19	158	1.80
par16-5(1015/3358)	0	11	76.35	24	13.90	**9**	352.32	522	5.05
f1000(1000/4250)	0	7	90.06	10	16.15	**1**	408.78	474	5.08
f2000(2000/8500)	0	11	337.76	9	60.10	**6**	2670.74	1001	9.90

TAB. 3.8 – Les résultats trouvés par SSV, GAV, SGAV et GA sur les instances *Jnh*.

Instances.cnf (#Vars/#Cls)	#U	**SSV** *sol*	time	**GAV** *sol*	time	**SGAV** *sol*	time	GA *sol*	time
jnh201(100/800)	0	**0**	0.11	**0**	**0.08**	**0**	0.10	31	0.63
jnh202(100/800)	1	**1**	3.41	**1**	0.55	**1**	12.73	36	0.63
jnh203(100/800)	1	**1**	3.39	2	0.55	**1**	12.80	39	0.63
jnh204(100/800)	0	**0**	0.18	**0**	0.17	**0**	0.52	42	0.63
jnh205(100/800)	0	**0**	0.09	**0**	0.08	**0**	0.08	42	0.63
jnh206(100/800)	1	**1**	3.32	**1**	0.54	**1**	12.72	38	0.62
jnh207(100/800)	0	**0**	0.28	1	0.54	**0**	0.14	37	0.64
jnh208(100/800)	1	**1**	3.39	**1**	0.54	**1**	12.73	37	0.64
jnh209(100/800)	0	**0**	0.08	**0**	0.11	**0**	0.14	33	0.64
jnh210(100/800)	0	**0**	0.08	**0**	**0.08**	**0**	0.07	38	0.63

TAB. 3.9 – Comparaison entre SSV, GAV et SGAV sur les instances *glassy* et *Color*.

Instances.cnf (#Vars/#Cls)	**SSV**		**GAV**		**SGAV**	
	sol	time	*sol*	time	*sol*	time
glassy-a(399/1862)	6	18.57	8	3.15	**5**	78.31
glassy-b(399/1862)	9	18.76	12	3.21	**8**	78.44
glassy-c(399/1862)	7	18.68	8	3.19	**6**	78.48
glassy-d(399/1862)	6	18.66	10	3.26	**6**	78.35
glassy-e(399/1862)	6	18.66	9	3.19	**6**	78.36
glassy-f(450/2100)	9	23.25	12	4.08	**8**	98.50
glassy-g(450/2100)	11	23.24	13	4.02	**9**	98.66
glassy-h(450/2100)	9	23.15	12	4.10	**7**	98.17
glassy-i(450/2100)	7	23.20	11	4.09	**7**	98.28
glassy-j(450/2100)	11	23.20	12	4.05	**9**	98.44
color-10-3(300/6475)	**0**	21.28	4	8.24	**0**	17.95
color-18-4(1296/95904)	17	1232.44	45	544.61	**15**	1510.09

3.5.2.2 Comparaison avec SSV, GAV et SGAV

La section ci-après compare les trois approches SSV, GAV et SGAV sur d'autres benchmarks plus difficiles.

D'après les résultats numériques donnés dans les Tables 3.9 . . .3.13, on observe que les algorithmes de SSV et GAV donnent de résultats satisfaisants mais les solutions trouvées par l'algorithme SGAV sont nettement meilleures que celles fournies par SSV et GAV.

Les trois approches arrivent à trouver la solution optimale pour quelques classes de problèmes (*MAX-2-SAT*, *MAX-3-SAT*, classe de *jnh*, classe d'*ais*, *f600*, *color10-3*). Pour les autres classes, nos approches trouvent des solutions satisfaisantes en un temps raisonnable.

En général, les résultats trouvés par SGAV sont nettement meilleurs que ceux fournis par les algorithmes GAV et SSV.

L'efficacité de notre approches est expliquée par la bonne combinaison entre la diversification et l'intensification ce qui mène l'algorithme à explorer efficacement l'espace de recherche et localiser une bonne solution.

Pour le reste de cette section, nous proposons une étude comparative avec certains algorithmes de l'état de l'art concernant MAX-SAT à savoir Walksat, FlipGA et GASAT.

TAB. 3.10 – Comparaison entre SSV, GAV et SGAV sur les instances de *Parity-32*.

Instances.cnf (#Vars/#Cls)	**SSV** *sol*	time	**GAV** *sol*	time	**SGAV** *sol*	time
par32-1-c(1315/5254)	**11**	138.03	49	28.51	**11**	658.82
par32-1(3176/10277)	47	659.41	65	122.55	**35**	1186.74
par32-2-c(1303/5206)	11	135.26	51	27.65	**10**	655.57
par32-2(3176/10253)	42	657.68	64	122.09	**37**	1195.25
par32-3-c(1325/5294)	**10**	139.52	50	28.55	11	679.15
par32-3(3176/10297)	49	660.23	68	122.64	**37**	1181.29
par32-4-c(1333/5326)	11	141.16	41	28.68	**9**	636.54
par32-4(3176/10313)	48	3634.44	58	122.88	**34**	1178.16
par32-5-c(1339/5350)	11	142.72	53	29.15	**11**	671.41
par32-5(3176/10325)	51	661.21	61	122.97	**37**	1174.43

TAB. 3.11 – Comparaison entre SSV, GAV et SGAV sur les instances *ais* et *Lran*.

Instances.cnf (#Vars/#Cls)	**SSV** *sol*	time	**GAV** *sol*	time	**SGAV** *sol*	time
ais10(181/3151)	**0**	1.93	1	2.25	**0**	**1.25**
ais12(265/5666)	1	29.58	1	5.58	**0**	**51,84**
ais6(61/581)	**0**	0.04	**0**	0.09	**0**	0.09
ais8(113/1520)	**0**	0.80	1	0.71	**0**	**1.90**
f1000(1000/4250)	7	90.06	10	16.15	**1**	408.78
f2000(2000/8500)	11	337.76	9	60.10	**6**	2670.74
f600(600/2550)	2	34.07	3	6.17	**0**	102.69

TAB. 3.12 – Comparaison entre SSV, GAV et SGAV sur les instances *Mat* et *difp*.

Instances.cnf (#Vars/#Cls)	**SSV** *sol*	time	**GAV** *sol*	time	**SGAV** *sol*	time
Mat25.shuffled(588/1968)	7	28.68	6	5.09	**3**	125.73
Mat26.shuffled(744/2464)	8	44.06	6	7.96	**2**	199.55
difp_19_0(1201/6563)	14	155.06	38	28.73	**11**	3532.01
difp_19_1(1201/6563)	14	155.52	44	29.22	**10**	764.02
difp_19_3(1201/6563)	15	156.87	38	28.94	**12**	766.35
difp_19_99(1201/6563)	16	156.71	44	29.14	**10**	766.11

TAB. 3.13 – Comparaison entre SSV, GAV et SGAV sur les instances *logistics* et *hgen*.

Instances.cnf (#Vars/#Cls)	**SSV** *sol*	time	**GAV** *sol*	time	**SGAV** *sol*	time
logistics.a(828/6718)	2	98.92	6	17.88	**1**	484.04
logistics.b(843/7301)	2	107.73	4	19.75	**1**	508.05
logistics.c(1141/10719)	3	201.57	6	36.96	**2**	3350.43
logistics.d(4713/21991)	11	2397.31	13	355.36	**7**	639.63
hgen2-a(500/1750)	5	22.77	6	3.95	**2**	97.45
hgen2-b(500/1750)	5	22.72	5	3.89	**3**	97.47
hgen2-c(500/1750)	3	22.88	8	3.86	**2**	97.53
hgen2-d(500/1750)	4	22.83	5	3.96	**2**	97.57
hgen2-e(500/1750)	4	22.76	8	3.96	**3**	97.67

TAB. 3.14 – Comparaison avec GASAT et FlipGA.

Instances (#Vars/#Cls)	**GASAT**		**FlipGA**		**SSV**		**SGAV**	
	fc	sec	**fc**	sec	**fc**	sec	**fc**	sec
f1000(1000/4250)	2.30	45	8.90	47	7	90.06	**1**	40.87
f2000(2000/8500)	7.35	97	16.90	122	11	337.76	**6**	267.07
par16-4-c(324/1292)	5.85	35	10.15	9	6	11.03	**5.00**	4.80
par32-5-c(1711/5722)	19.60	129	27.40	77	**11**	142.72	**11**	67.14
par32-5(6458/13748)	41.25	813	50.65	290	51	661.21	**37**	117.44
Mat25.shuffled(588/1968)	7.60	161	10.10	28	7	28.68	**3**	12.57
Mat26.shuffled(744/2464)	8.00	749	12.89	44	8	44.06	**2**	19.95
difp_19_0(1201/6563)	84.25	661	87.60	109	14	155.06	**11**	353.20
difp_19_99(1201/6563)	81.40	658	87.95	107	16	156.71	**10**	76.61
glassy-a(399/1862)	**5.00**	18	7.60	16	6	18.57	**5.00**	7.83
glassy-b(450/2100)	8.95	86	11.45	21	9	23.15	**7.00**	9.8
hgen2-a(500/1750)	**1.40**	22	6.24	16	5	22.77	2	9.74
hgen2-b(500/1750)	7.00	1.80	22	18	3	22.88	**2**	9.75

3.5.3 Comparaison avec GASAT et FlipGA

GASAT est un algorithme génétique hybride proposé récemment par [Lardeux *et al.*, 2006]. Il utilise une recherche taboue comme technique locale d'amélioration et un opérateur spécifique au problème MAX-SAT. FlipGA est un algorithme mimétique proposé par [Marchiori et Rossi, 1999] et utilisé dans [Lardeux *et al.* 06] pour des études comparatives.

La Table 3.14 compare SSV, SGAV, GASAT et FlipGA.

- Les paramètres de FlipGA sont : une population de 10 individus, un nombre maximum de croisement de 10^2 et un taux de mutation = 0.9.
- Les paramètres de GASAT sont : une population de 100 individus, une sous population de 15 individus, un nombre maximum de croisement est 10^3 et une recherche taboue comme technique locale d'amélioration avec 10^4 itérations pour chaque appel.

Comparant nos approches à FlipGA, SSV et SGAV surpassent clairement FlipGA pour toutes les instances. Comparant nos approches à GASAT, la Table 3.14 montre une légère performance en faveur de notre variante SGAV. SSV et SGAV produisent de meilleures solutions sur les problèmes de *parity*, de *Mat* et de *glassy* comparé à GASAT qui s'exécute mieux sur des instances de *lran* (f1000, f2000).

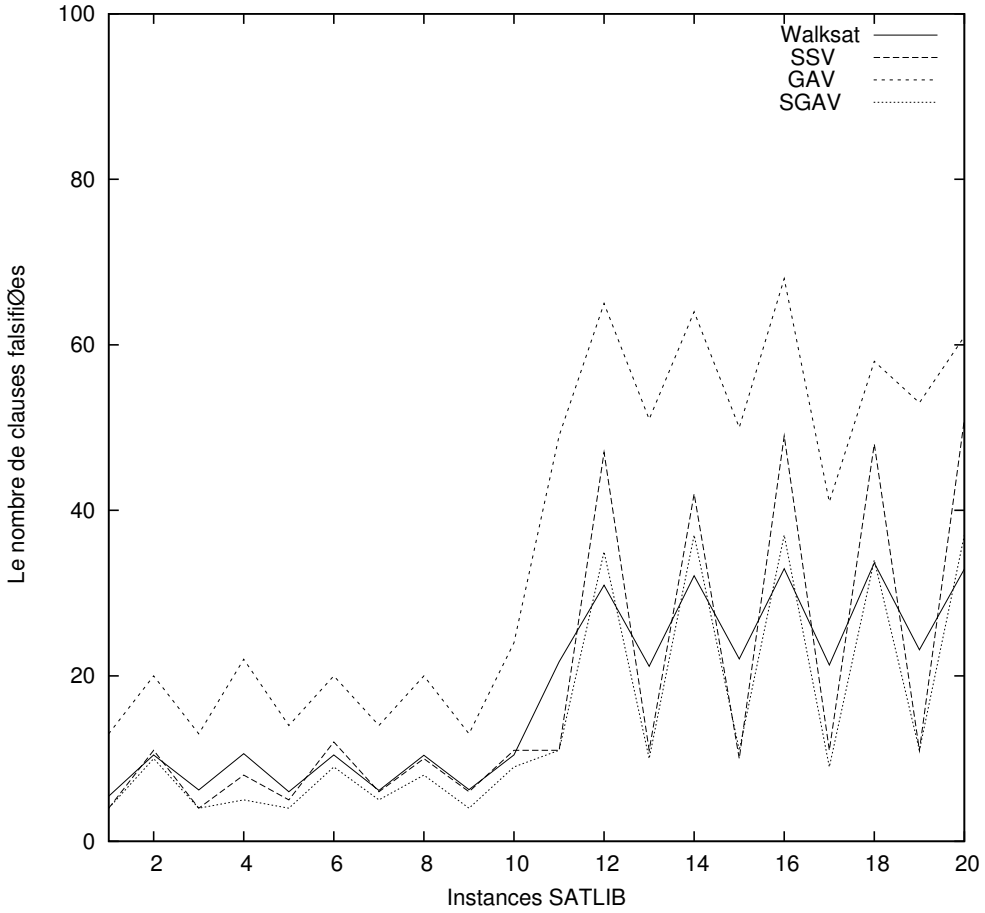

FIG. 3.2 – Comparison entre SSV, GAV, SGAV et Walksat

3.5.4 Comparaison avec WalkSat

Walksat est une recherche locale permettant de trouver une affectation booléenne satisfaisant un ensemble de clauses de variables propositionnel. La Figure 3.2 compare nos variantes (SGAV et SSV) et Walksat. Les résultats concernant Walksat sont tirés de (Zhang et al, 2003) où le Walksat a été employée avec un total de 10 millions de flippage de variables (MAXflips). Les résultats sont pour 20 essais pour chaque instance.

Selon l'histogramme de la Figure 3.2, nous pouvons observer de bonne performance en faveur de nos approches qui trouvent des solutions satisfaisantes. Comme la montre la Figure 3.2, pour la plupart des cas soit la qualité de la solution est dans les trois cas comparable. Soit l'algorithme SGAV fournit des résultats meilleurs que ceux trouvés par Walksat.

TAB. 3.15 – Temps CPU pour des instances *MAX-2-SAT* et *MAX-3-SAT*.

Instances	# U	**SSV**	**SGAV**	Max Solver	BF	WCSP
p100(50/100)	4	0.30	0.04	0.01	0.01	0.01
p150(50/150)	8	0.32	0.04	0.04	0.04	0.01
p200(50/200)	16	0.35	0.06	0.02	4.81	0.02
p250(50/250)	22	0.38	0.07	0.02	28.16	0.02
p300(50/300)	32	0.40	0.07	0.07	394.09	0.19
p350(50/350)	41	0.43	0.08	0.12	2875.61	0.29
p400(50/400)	45	0.45	0.09	0.09	2592.49	0.15
p450(50/450)	63	0.48	0.11	0.65	-	1.52
p500(50/500)	66	0.51	0.11	0.42	-	0.80
p2600(100/600)	65	0.86	0.18	106.22	-	2762.36
p2300(150/300)	4	0.94	0.15	0.06	0.08	1.28
p2450(150/450)	22	1.04	0.20	1.93	-	154.96
p3250(50/250)	2	0.43	0.08	0.03	0.01	0.02
p3300(50/300)	4	0.47	0.09	0.12	0.10	0.18

3.5.5 Comparaison avec les solvers exacts de MAX-SAT

La comparaison entre les méthodes exactes et approchées n'est pas triviale. Dans cette section, nous donnons quelques résultats trouvés par quelques solvers exacts MAX-SAT.

Nous comparons nos approches SGAV et SSV Solvers avec *MaxSolver* [Xing et Zhang, 2005], l'algorithme *BF* développé par Borchers et Furman [Borchers et Furman, 1999] et WCSP (A Weighted CSP-based solver) développé par [Givry *et al.* , 2003]. La table 3.15 donne le temps CPU des méthodes SSV, SGAV ainsi que les fameux solvers MAX-SAT. La marque "-" indique que la méthode correspondante a été arrêtée après 5 heures d'exécution. La solution donnée par les différentes méthodes est l'optimum.

Les Tables 3.15 et 3.16 montrent et confirment l'efficacité de nos approches évolutionnaires dans la résolution du problème MAX-SAT.

3.6 Conclusion

Dans ce chapitre, nous avons proposé quelques approches évolutionnaires pour le problème MAX-SAT. La méthode de base utilise une nouvelle stratégie de sé-

TAB. 3.16 – Temps CPU pour des instances SATLIB.

Instances	# U	**SSV**	**SGAV**	Max Solver	BF	WCSP
aim50no2(50/100)	1	0.32	0.04	0.03	0.02	0.04
aim50no3(50/100)	1	0.32	0.05	0.03	0.02	0.06
aim1001.6no1(100/160)	1	0.63	0.08	649.25	449.55	670.72
pret60-40(60/160)	1	0.40	0.05	3.27	4.68	85.27
pret60-60(60/160)	1	0.40	0.06	3.35	4.69	85.07
pret60-75(60/160)	1	0.41	0.06	4.12	4.62	84.94
dubois25(75/200)	1	0.53	0.08	16.77	99.31	2358.37

lection basée sur la diversité et la fitness pour choisir des individus pour participer à la phase de reproduction et produire d'autres individus. Elle se sert d'un opérateur de croisement spécifique au problème MAX-SAT combiné avec une méthode d'amélioration de recherche locale stochastique (SLS).

Les approches proposées surpassent l'algorithme génétique classique (avec une stratégie de sélection standard, un croisement standard et sans recherche locale).

Les résultats obtenus par SGAV (l'une des variantes proposées) sont très encourageants, montrant l'avantage de fusionner les composants principaux : la stratégie de sélection, l'opérateur de croisement spécifique au problème MAX-SAT et la recherche locale (SLS). Plusieurs expérimentations menées sur des benchmarks et des comparaisons avec des algorithmes génétiques, SSV, Walkasat, FlipGA et GASAT montrent l'efficacité de notre nouvelle approche. En perspective, nous souhaitons regarder l'hybridation d'algorithmes complets et d'algorithmes de recherche locale ou évolutionnaires.

Dans le chapitre qui suit, nous proposons une recherche taboue pour le problème de la détermination du gagnant dans les enchères combinatoires (PDG).

Parite 2. Une recherche Taboue pour le problème de la détermination du gangnat dans les enchères combinatoires (PDG)

Les travaux présentés dans cette deuxième partie ont fait l'objet de publications dans les revues :

- International Journal of Internet and Enterprise Management,
IJIEM [Boughaci et Drias , 2005 b].

et les conférences :

- ACM SAC'07[][Boughaci *et al.*, 2007 d],
- ITCC'05 [Boughaci et Drias , 2005 n],
- CE'04 [Boughaci et Drias 2004 e].
- CE'03 [Boughaci et Drias 2003].

Chapitre 4

Une Recherche Taboue pour les Enchères Combinatoires

4.1 Introduction

Dans ce chapitre, nous proposons une métaheuristique de recherche taboue (RT) pour le problème de la détermination du gagnant (PDG) dans les enchères combinatoires. Le problème PDG est un problème complexe et équivalent au problème d'emballage d'ensemble qui est NP-Complet [Rothkopf *et al.*, 1998]. Des expérimentations numériques sont réalisées sur des benchmarks de diverses tailles dans le but de tester et de prouver l'efficacité de notre approche.

4.2 Présentation de notre approche

La méthode de la recherche taboue a remporté un immense succès dans la résolution de nombreux problèmes d'optimisation combinatoires. Vu son succès, nous allons l'utiliser pour traiter le problème d'enchère combinatoires qui un problème difficile.

4.2.1 Adaptation de la recherche tabou au Problème Etudié

Pour pouvoir appliquer la recherche taboue, on doit passer par les étapes suivantes :

1. Créer aléatoirement une solution initiale.
2. Avoir un opérateur de mouvement permettant de générer des solutions voisines.

3. Générer un ensemble de solutions voisines en utilisant l'opérateur de mouvement.
4. Sélectionner la meilleure solution non taboue.
5. Avoir une liste taboue pour sauvegarder les solutions parcourues et ce pour éviter les optima locaux.
6. Définir un critère d'arrêt.

4.2.1.1 Représentation de la Solution

Une solution est une collection d'offres satisfaisant les contraintes du problème et le cas échéant optimisant la fonction objectif. Nous utilisons un vecteur A de taille variable (ne dépassant pas le nombre d'offres) pour représenter une telle solution, où chaque composante A_i reçoit le numéro de l'offre gagnante.

La solution initiale de la recherche locale que nous proposons est générée aléatoirement suivant le codage RK qui fonctionne comme suit :

nous générons une suite r de n nombres réels aléatoires où n est le nombre d'offres soumises. Pour chaque offre B_j une valeur de clef r_j lui est associée et qui constitue son ordre de sélection. La première offre à choisir et à inclure dans l'allocation est celle ayant la plus grande valeur de clef. Ensuite, l'offre ayant la deuxième plus grande clef est acceptée dans la collection si son acceptation ne crée aucun conflit avec les offres déjà sélectionnées dans la collection courante, autrement elle est rejetée. Et ainsi de suite, jusqu'à ce que les n offres soient examinées. Nous obtenons un sous-ensemble d'offres qui peut être une solution au PDG.

Exemple :

Considérons un ensemble de quatre objets 1, 2, 3, 4 à vendre aux enchères et trois offres B_1, B_2, B_3 soumises par trois soumissionnaires. Chaque offre (B_i, p_i) indique le prix p_i que le soumissionnaire est disposé à payer pour son offre B_i.

Soient les offres d'achat proposées suivantes :

1. Offre 1 : B_1 ((1, 2), 500.25).
2. Offre 2 : B_2 ((1, 2, 3), 678).
3. Offre 3 : B_3 ((3, 4), 200.10).

Pour produire une solution réalisable pour cet exemple, nous suivons les étapes suivantes :

1. Produire d'abord trois nombres réels aléatoires, par exemple, r= 0.65, 0.70, 0.80.

2. La première offre à accepter est l'offre B_3 ayant la plus grande clef (0.80). L'allocation courante reçoit l'offre B_3, A= (B_3).
3. Puis l'offre ayant la deuxième plus grande clef, B_2 est à écarter parce qu'elle est en conflit avec l'offre B_3 actuellement dans A. Elles partagent le même objet 3.
4. Ensuite, l'offre B_1 peut être ajoutée à l'allocation A parce qu'il n'y aura aucun conflit avec les offres dans A. A=(B_3, B_1).

Nous obtenons l'allocation A= (B_3, B_1) qui peut être une solution au problème du PDG. Le prix global est tout simplement la somme des prix des offres gagnantes : 200.10+500.25=700.35.

A partir de l'allocation A, nous pouvons dire que les objets 1 et 2 sont affectés au soumissionnaire 1 et les objets 3 et 4 sont affectés au soumissionnaire 2.

Les offres gagnantes sont : B_3 et B_1. B_2 est une offre perdante.

Pour cet exemple, nous remarquons que tous objets sont vendus.

4.2.1.2 Relation de Voisinage

La metaheuristique taboue est une méthode itérative où chaque itération consiste à améliorer une configuration courante. L'amélioration consiste à effectuer des transformations successives sur la configuration courante.

Un mouvement est un opérateur utilisé pour produire des solutions de voisinage. L'opérateur de mouvement consiste en choisissant une meilleure offre à partir des offres insatisfaites pour être incluse dans l'allocation courante pour obtenir de nouvelles allocations.

Nous proposons deux mouvements qui peuvent être utilisés pour produire le voisinage des solutions : "on-Bid" et "on-Item".

1. Mouvement "on-Bid" : l'espace d'état est défini par l'ensemble des offres insatisfaites qui ne sont pas dans l'allocation courante. Ces offres sont considérées admissibles et peuvent être incluses dans l'allocation courante. La meilleure offre (celle qui maximise le prix global quand on l'ajoute à l'allocation) dans le voisinage actuel est choisie pour être incluse dans l'allocation courante et toutes les offres incompatibles dans l'allocation sont à enlever.
2. Mouvement "on-Item" : L'espace d'état est défini par lensemble des objets qui ne sont pas couverts par les offres dans l'allocation courante. Alors la meilleure offre couvrant tels objets est choisie pour être incluse dans l'allocation courante et toutes les offres incompatibles dans l'allocation sont enlevées.

Après avoir généré toutes les configurations voisines en utilisant l'opérateur de mouvement (inclure une nouvelle offre dans l'allocation courante), la meilleure configuration non taboue est sélectionnée pour être la solution candidate.

4.2.1.3 Liste taboue

Pour éviter des optimums locaux, l'offre récemment choisie reçoit le statut tabou et garde ce statut pour un certain nombre λ des itérations. Le numéro de l'offre est inclus dans la liste taboue qui agit en tant que liste $FIFO$ à taille fixe. Cette offre est interdite pour les prochaines λ itérations. Cependant, elle peut être re-sélectionnée malgré son statut tabou par critère d'aspiration quand elle mène à une solution meilleure.

4.2.1.4 Etape de diversification

Pour explorer de nouvelles régions, une étape de diversification est ajoutée. Elle consiste à choisir aléatoirement une offre insatisfaite pour être incluse dans la meilleure allocation courante et toutes offres incompatibles sont enlevées. La stratégie de diversification est répétée pour n étapes consécutives où n est le nombre d'offres.

4.2.1.5 Graphe de conflit

Pour ne manipuler que des solutions réalisables, nous avons créé un graphe de conflit où les sommets sont les offres et les arcs relient les offres qui ne peuvent pas être acceptées ensemble. Ce graphe permet de détecter les offres en conflit qui partagent au moins un objet. Selon ce graphe nous pouvons assurer et maintenir la faisabilité de nos allocations.

4.2.1.6 Critère d'Arrêt

Le processus de recherche est stoppée après un certain nombre d'itérations (max_iter) fixé d'une manière empirique. Vu que la méthode de recherche taboue démarre d'une solution aléatoire, la recherche peut être recommencée, avant de quitter, pour un certain nombre d'exécution (max_trials).

4.2.1.7 Fonction Objectif

Une solution est caractérisée par une fonction objectif permettant de mesurer sa qualité. La qualité d'une solution est donnée par la somme des prix des offres gagnantes.

$$\sum_{i=0}^{L} Price(B_i) \text{ où } L \text{ est la taille de l'allocation gagnante.}$$

4.2.1.8 L'algorithme de recherche tabou pour le PDG

Une version de l'algorithme de la recherche tabou pour le PDG est donnée dans l'algorithme12.

Algorithm 12 : Recherche taboue pour le PDG.

1: // max_trials est le nombre des exécutions
// max_iter est le nombre maximum des itérations
// $iter\_best$ est le numéro de la meilleure itération.
// A^* est la meilleure solution d'une exécution courante avec le coût F^*
// A^{**} est la meilleure solution trouvée durant toutes les exécutions avec
// le maximum revenue F^{**},
// TL est la liste taboue initialement vide,
2: Créer le graphe de conflit ;
3: $A^{**}= \phi$; $F^{**} = 0$; $iter$ =0 ;
4: **Pour** (i= 1 à max_trials)
5: **Faire**
6: Générer aléatoirement une solution initiale A selon le RK
7: **Tant que** (iter $\prec$ max_iter)
8: **Faire**
9: iter = iter+1 ;
10: Générer les allocations voisines en utilisant les mouvements On-Bid et On-Item ;
11: Sélectionner le meilleur mouvement ;
12: Inclure l'offre choisie dans l'allocation A ;
13: Enlever toutes les offres incompatibles dans A,
14: Insérer mouvement appliqué dans TL ;
15: Evaluer F(A) ;
16: **Si** ($iter$-$iter\_best \preceq 40$) **alors** étape de diversification ;
17: **FinSi**
18: **Si** $F \succ F^*$ **alors** $A* = A$; $F* = F$; $iter\_best$= $iter$;
19: **FinSi**
20: **Fait**
21: **Si** $F^* \succ F^{**}$ **alors** $A^{**} = A^*$; $F^{**} = F^*$; **FinSi**
22: **Fait**
23: Retourner la meilleure solution A^{**} avec le revenue F^{**}

4.3 Résultats expérimentaux

Dans le but de valider notre approche, des tests ont été effectués sur différents problèmes de diverses tailles. Nos algorithmes ont été implémentés en C sous

Linux et sous la machine Pentium- IV 2.8 GHz, 1GO de RAM.

Dans le but de tester et de prouver l'efficacité de notre approche, une étude comparative avec quelques algorithmes de l'état de l'art concernant PDG à savoir *Casanova* et $SAGII$ est établie dans cete thèse.

4.3.1 Benchmarks

Récemment Lau et Goh ont fourni de nouvelles données de diverses tailles ayant jusqu'à 1500 objets et 1500 offres [Lau et Goh, 2002]. Ces benchmarks tiennent compte de plusieurs facteurs tels : le facteur des prix, le facteur de préférence des enchérisseurs et les facteurs de distribution des objets sur les offres.

Dans nos expérimentations, nous utilisons les données pré-générées dans [Lau et Goh, 2002] pour lesquelles le CPLEX n'arrive pas àă trouver la solution optimale en un temps raisonnable [Guo *et al.*, 2006]. Cette batterie de test contenant 500 instances est disponible sur la page Web dont l'URL est : HTTP :/logistics.ust.hk/ zhuyi/instance.zip. Ces instances peuvent être divisées en 5 groupes de problèmes où chaque groupe contient 100 instances, et où m est le nombre d'objets et n est le nombre d'offres. Les cinq groupes sont donnés comme suit :

- REL-1000-500 : contenant 100 instances allant de in101 à in200, et où m=500 et n=1000.
- REL-1000-1000 : contenant 100 instances allant de in201 à in300, et où m=1000 et n=1000.
- REL-500-1000 : contenant 100 instances allant de in401 à in500, et où m=1000 et n=500.
- REL-1500-1000 : contenant 100 instances allant de in501 à in600, et où m=1000 et n=1500.
- REL-1500-1500 : contenant 100 instances allant de in601 à in700, et où m=1500 et n=1500.

Le format du fichier d'entrée est :
- m n
- n lignes. Le premier nombre de chaque ligne i est le prix de l'offre i, suivi d'un certain nombre de nombres entiers, qui sont les objets couverts par l'offre i.

4.3.2 Valeurs des paramètres

Avant de se lancer dans l'étude comparative entre les différentes approches présentées, le réglage de certains paramètres s'impose. Ceux-ci sont fixés suivant les études expérimentales.

TAB. 4.1 – Résultats trouvés par TS pour différent paramètres.

Test set	# *ins*	*maxs _trials*	*max _iter*	*TS*	*time*
REL-500-1000	100	1	50	52120,26	0,13
REL-500-1000	100	100	100	62446,03	21,26
REL-500-1000	100	**500**	**500**	**65286,94**	**91,07**
REL-500-1000	100	1000	500	65599.41	123.03
REL-500-1000	100	1500	500	66258,51	187,83
REL-1000-500	100	1	50	55858,67	0,02
REL-1000-500	100	100	100	69435,71	5,20
REL-1000-500	100	**500**	**500**	**71985,34**	**25,84**
REL-1000-500	100	1000	500	72630.72	48,82
REL-1000-500	100	1500	500	73009,51	68,96

Dans la Table 4.1, nous donnons un exemple des expérimentations faites sur la recherche taboue (TS) pour différents paramètres. Les valeurs prises sont celles pour lesquelles il existe un compromis entre la qualité de la solution obtenue par l'approche et le temps d'exécution de l'algorithme (ligne en gras de la Table 4.1).

D'après la Table 4.1, on remarque que la recherche taboue procure des solutions de bonne qualité. Les paramètres max_iter et max_trials utilisés jouent un rôle dans la qualité des solutions trouvées. Si ceux-ci sont augmentés, la recherche devient efficace mais perd en rapidité d'exécution, et à l'inverse ils diminuent l'efficacité mais accélèrent la recherche.

4.3.3 Résultats

L'algorithme développé a été testé sur un certain nombre de problèmes PDG difficiles. Suite à plusieurs tests, les paramètres de notre TS sont fixés comme suit : le nombre des exécutions (max_trials) est fixé à 500, le nombre maximum des itérations (max_iter) est fixé à 500, et la taille de la liste taboue est fixée à 40. Les résultas correspondant sont donnés dans la série des tables de 4.2 à 4.6.

TAB. 4.2 – Les résultats trouvés par TS sur quelques instances REL 1000-500

Instance	m	n	$time$	sol_TS
in101	1000	500	57.8600	66170.61719
in102	1000	500	63.4360	64716.31641
in103	1000	500	128.6880	66350.99219
in104	1000	500	120.5630	62524.23047
in105	1000	500	105.3600	64312.56641
in106	1000	500	129.4220	64591.70312
in107	1000	500	128.5160	63972.62891
in108	1000	500	119.8440	68776.34375
in109	1000	500	80.9840	64343.07812
in110	1000	500	115.3130	60275.66406
in111	1000	500	107.8750	66546.82812
in112	1000	500	68.3280	68583.75781
in113	1000	500	93.7030	66663.48438
in114	1000	500	69.3750	69452.16406
in115	1000	500	82.4210	62579.47266
in116	1000	500	67.6250	64849.85547
in117	1000	500	67.6560	64229.21484
in118	1000	500	105.2820	65843.74219
in119	1000	500	99.3910	65460.73047
in120	1000	500	71.6560	68216.00000
in190	1000	500	104.0630	66474.48438
in191	1000	500	69.0460	65800.05469
in192	1000	500	100.9060	64281.60156
in193	1000	500	81.5310	66242.07812
in194	1000	500	71.3590	67076.80469
in195	1000	500	86.6250	62715.24219
in196	1000	500	76.8270	64753.22266
in197	1000	500	79.7190	62050.87891
in198	1000	500	86.6560	66956.14062
in199	1000	500	83.6550	64714.54297
in200	1000	500	127.7970	69867.20312

TAB. 4.3 – Les résultats trouvés par TS sur quelques instances REL 1000-1000

Instance	m	n	$time$	sol_TS
in201	1000	1000	85.3750	81557.74219
in202	1000	1000	187.4210	82879.72656
in203	1000	1000	78.3740	79505.18750
in204	1000	1000	140.8750	81969.04688
in205	1000	1000	106.7340	79977.89844
in206	1000	1000	124.0940	79436.56250
in207	1000	1000	78.9690	91033.51562
in208	1000	1000	133.7660	82099.07812
in209	1000	1000	61.9070	80786.38281
in210	1000	1000	164.5310	83350.42188
in211	1000	1000	66.2030	79746.14844
in212	1000	1000	158.8900	80715.77344
in213	1000	1000	117.8440	81740.95312
in214	1000	1000	80.4220	77525.90625
in215	1000	1000	120.9990	85549.66406
in216	1000	1000	89.4060	78777.96875
in217	1000	1000	137.6570	81545.43750
in218	1000	1000	88.5470	86936.78125
in219	1000	1000	134.9380	86064.50781
in220	1000	1000	124.5480	86937.85938
in290	1000	1000	107.4840	81053.27344
in291	1000	1000	51.4220	81294.33594
in292	1000	1000	49.0150	82221.24219
in293	1000	1000	54.6410	76017.25781
in294	1000	1000	49.9840	80903.21094
in295	1000	1000	48.0160	83086.79688
in296	1000	1000	103.7490	84754.59375
in297	1000	1000	53.5000	78341.52344
in298	1000	1000	73.3900	81659.42188
in299	1000	1000	89.2500	84587.35156
in300	1000	1000	78.8290	83349.51562

TAB. 4.4 – Les résultats trouvés par TS sur quelques instances REL 500-1000

Instance	m	n	$time$	sol_TS
in401	500	1000	44.1400	68485.81250
in402	500	1000	23.5780	72820.03125
in403	500	1000	34.1570	74843.96094
in404	500	1000	16.8590	73385.62500
in405	500	1000	15.9060	72674.25000
in406	500	1000	37.1250	71791.03125
in407	500	1000	15.5780	71578.48438
in408	500	1000	27.3750	70144.19531
in409	500	1000	25.4840	67177.35156
in410	500	1000	14.0160	72791.68750
in411	500	1000	28.2970	68548.03906
in412	500	1000	13.7190	75292.63281
in413	500	1000	19.3440	72561.90625
in414	500	1000	40.9850	70344.38281
in415	500	1000	14.1720	78106.73438
in416	500	1000	75.5000	70358.95312
in417	500	1000	39.0780	69895.95312
in418	500	1000	29.7030	67272.32031
in419	500	1000	13.3750	73332.96875
in420	500	1000	23.7810	71381.02344
in490	500	1000	29.0000	71423.73438
in491	500	1000	21.8600	74181.24219
in492	500	1000	17.6250	70296.63281
in493	500	1000	27.5950	72197.60938
in494	500	1000	14.8750	72922.96875
in495	500	1000	25.7970	72919.67969
in496	500	1000	30.3590	73376.74219
in497	500	1000	27.6400	70942.18750
in498	500	1000	13.2030	74213.25000
in499	500	1000	13.6090	71246.87500
in500	500	1000	23.3430	69909.96875

TAB. 4.5 – Les résultats trouvés par TS sur quelques instances REL 1500-1000

Instance	m	n	$time$	sol_TS
in501	1500	1000	98.7180	82216.35938
in502	1500	1000	120.8290	74127.61719
in503	1500	1000	114.1100	77005.81250
in504	1500	1000	155.5460	81903.02344
in505	1500	1000	163.8430	77977.65625
in506	1500	1000	151.9850	74708.78125
in507	1500	1000	121.9380	75916.19531
in508	1500	1000	341.7500	79885.98438
in509	1500	1000	250.3270	76115.86719
in510	1500	1000	134.2190	78841.15625
in511	1500	1000	104.9220	78793.83594
in512	1500	1000	188.3600	79739.67969
in513	1500	1000	189.6870	75425.91406
in514	1500	1000	124.5010	78416.59375
in515	1500	1000	560.6250	78874.25781
in516	1500	1000	683.6720	83663.75000
in517	1500	1000	139.9220	72569.57031
in518	1500	1000	114.1720	80027.38281
in519	1500	1000	109.3900	80892.13281
in520	1500	1000	109.3430	81658.50000
in590	1500	1000	203.1250	78712.15625
in591	1500	1000	109.0630	78177.09375
in592	1500	1000	136.9840	78381.93750
in593	1500	1000	96.7340	80658.93750
in594	1500	1000	325.5470	80985.63281
in595	1500	1000	159.6260	76721.54688
in596	1500	1000	274.0000	75542.06250
in597	1500	1000	194.0790	77777.77344
in598	1500	1000	102.9990	79201.96875
in599	1500	1000	374.0160	77527.33594
in600	1500	1000	302.0160	82555.91406

TAB. 4.6 – Les résultats trouvés par TS sur quelques instances REL 1500-1500

Instance	m	n	$time$	sol_TS
in601	1500	1500	100.7650	97473.85938
in602	1500	1500	155.3430	93873.31250
in603	1500	1500	137.9540	92568.61719
in604	1500	1500	96.7040	92869.78125
in605	1500	1500	175.1410	95787.59375
in606	1500	1500	334.1250	104346.07031
in607	1500	1500	267.7970	98674.39844
in608	1500	1500	95.6250	91554.61719
in609	1500	1500	103.1090	96652.44531
in610	1500	1500	146.0310	99975.09375
in611	1500	1500	213.2500	97051.67188
in612	1500	1500	103.2180	102453.14844
in613	1500	1500	220.4220	92616.91406
in614	1500	1500	269.8910	97510.72656
in615	1500	1500	127.2350	96151.64844
in616	1500	1500	100.6090	100601.55469
in617	1500	1500	247.1250	103684.74219
in618	1500	1500	164.6250	97672.58594
in619	1500	1500	218.7650	97306.30469
in620	1500	1500	206.8280	97060.17969
in690	1500	1500	125.2810	100613.71094
in691	1500	1500	96.0310	98952.00000
in692	1500	1500	95.7040	98353.00000
in693	1500	1500	156.8130	95603.90625
in694	1500	1500	339.3910	101816.08594
in695	1500	1500	202.0000	97687.15625
in696	1500	1500	245.2810	96410.53906
in697	1500	1500	192.6410	91787.14844
in698	1500	1500	305.6880	95572.62500
in699	1500	1500	187.9060	103762.70312
in700	1500	1500	198.6560	101510.20312

TAB. 4.7 – Casanova, SAGII, CPLEX et TS sur quelques benchmarks.

Instance	*Casanova*		*SAGII*		*CPLEX*	*TS*	
	sol	time	sol	time	sol	sol	time
$R_{1000-1000}$	52048.73	113.55	86179.64	45.19	81755.45	82879,73	187,42
$R_{1000-1000}$	51340.27	111.16	83500.82	44.80	80479.80	82099,07	133,76
$R_{1000-1000}$	54998.44	117.16	86747.23	44.58	81563.71	88513,37	63,32
$R_{1000-1000}$	51003.39	132.92	88513.37	44.58	83195.99	91033,51	78,96
$R_{1000-1500}$	53992.12	164.94	85101.43	71.05	79453.93	82216,36	98,72
$R_{1000-1500}$	58365.02	164.66	88086.29	67.56	74623.20	85143,13	202,39
$R_{1000-1500}$	58527.76	171.72	82046.16	69.03	80656.88	80658,94	136,98
$R_{1000-1500}$	55821.04	171.80	82341.22	68.11	78085.08	80985,63	325,54
$R_{1000-1500}$	56001.82	174.25	83772.91	67.09	76334.65	82555,91	302,01
$R_{1500-1500}$	65543.41	161.11	104346.07	90.73	92981.93	102905,26	97,97
$R_{1500-1500}$	64962.00	166.88	106056.08	90.95	98763.83	108113,19	222,17
$R_{1500-1500}$	70140.69	170.56	105699.93	91.09	98763.83	101816,08	339,39
$R_{1500-1500}$	65026.40	171.70	103252.95	90.42	92408.82	103762,70	187,9
$R_{1500-1500}$	62404.80	160.98	105462.71	91.22	94840.40	101510,20	198,65

4.3.4 Comparaisons

La comparaison entre TS, *Casanova*, *SAGII* et CPLEX est donnée dans la Table 4.7. Les instances choisies sont des problèmes difficiles où CPLEX n'arrivait pas à trouver la solution exacte en un temps raisonnable, c'est pourquoi une durée de 3600 secondes au maximum a été donnée à l'algorithme CPLEX pour trouver une solution. La colonne CPLEX de la table 4.7 donne la solution trouvée en 3600 secondes. Pour les autres méthodes *Casanova*, *SAGII* et *TS*, on reporte la solution trouvée par l'algorithme et le temps nécessaire pour trouver cette solution[1].

D'après la table 4.7 représentant les tests effectués sur quelques instances difficiles de PDG, on constate qu'en général, l'algorithme de recherche taboue s'est avéré très efficace par rapport aux autres méthodes, voire parfois excellent. Ceci en ce qui concerne la qualité de la solution obtenue et le temps de calcul qui lui est approprié. La recherche taboue donne des solutions qui sont nettement meilleures que celles trouvées par Casanova. Les résultats trouvés par SAGII sont comparables à ceux donnés par notre approche.

[1]Pour CPLEX qui est une méthode exacte, le temps est fixé à 3600 seconde pour chaque benchmark

TAB. 4.8 – TS vs. Casanova

Test set	*time*	μ $\_TS$	*time*	μ $_{Casanova}$	$\delta\%$
REL-500-1000	91,07	**65286,94**	119.46	37053.78	43,24
REL-1000-500	25,84	**71985,34**	57.74	51248.79	28,80
REL-1000-1000	104,30	**81633,63**	111.42	51990.91	36,31
REL-1000-1500	223,37	**77931,41**	168.24	56406.74	27,62
REL-1500-1500	175,68	**97824,64**	165.92	65661.03	32,88

TAB. 4.9 – TS vs. SAGII

Test set	*time*	μ $\_TS$	*time*	μ $_{SAGII}$	$\delta\%$
REL-500-1000	91,07	**65286,94**	38.06	64922.02	+0,55
REL-1000-500	25,84	71985,34	24.46	73922.10	-0.02
REL-1000-1000	104,30	81633,63	45.37	83728.34	-0.02
REL-1000-1500	223,37	77931,41	68.82	82651.49	-0.05
REL-1500-1500	175,68	97824,64	91.78	101739.64	-0.03

La Table 4.8 (respectivement Table 4.9) donne les différents résultas trouvés par TS et $Casanova$ (respectivement ($SAGII$) où les colonnes μ correspondent à la solution moyenne des 100 problèmes dans chaque groupe, trouvée par chaque méthode. La colonne *time* correspond au temps moyen en second nécessaire pour trouver la solution. La colonne δ est le taux d'amélioration qui égale (μ_{TS} - $\mu_{Casanova}$)/μ_{TS} (respectivement (μ_{TS} - μ_{SAGII})/ μ_{TS}). A partir de la Table 4.8, on remarque que TS donne une amélioration de 27 to 43 par cent dans les résultats en comparaison avec Casanova. Les résultats de TS sont nettement meilleurs que ceux fournis par Casanova, concernant la qualité des solutions et le temps mis pour les trouver.

A partir de la Table 4.9, on remarque que malgré les outils sophistiqués (Branch-and bound et la phase du prétraitement) utilisés dans $SAGII$, notre TS arrive à trouver des solutions comparables à $SAGII$ et la différence entre TS et $SAGII$ n'est pas très grande. En général, on peut dire que les résultats trouvés sont très encourageants et montrent l'intérêt de notre approche.

4.4 Conclusion

Dans ce chapitre, le problème de la détermination du gagnant (PDG) dans les enchères combinatoires a été étudié. Il s'agit d'un problème d'optimisation combinatoire assez complexe. Nous avons proposé une nouvelle approche basée sur la recherche taboue pour trouver une allocation optimale. Plusieurs expérimentations menées sur des benchmarks et des comparaisons avec des algorithmes de l'état de l'art concernant PDG à savoir : $Casanova$ et $SAGII$ montrent l'efficacité de notre nouvelle approche. Dans le chapitre qui suit, nous proposons un système à base d'agents et utilisant la métaheuristique de recherche taboue pour simuler le mécanisme d'enchères combinatoires.

Chapitre 5

Un système multiagents pour simuler le mécanisme d'enchères combinatoires

5.1 Introduction

Un agent est un logiciel autonome, réactif et capable de communiquer avec les autres agents. Un système multiagents est un système qui comprend un ensemble d'agents. Si les agents sont en coordination alors on parle d'un système multiagents collaboratif ou coopératif. Si les agents sont en compétition, on parle alors d'un système multiagents compétitif.

En effet la technologie d'agents se prête bien à la modélisation des problèmes complexes en les distribuant sur un ensemble d'entités intelligentes et autonomes dotées d'une grande liberté de manoeuvre et une dynamique de résolution. La coopération est une notion importante dans les systèmes multiagents. A cette description, nous entamons ce chapitre pour décrire notre approche proposée pour concevoir un système à base d'agents pour simuler le mécanisme d'enchères combinatoires.

Nous présentons, tout d'abord, le concept d'agent intelligent, et nous donnons les notions de base du langage AUML (Agent Unified Modeling Language) utilisé pour modéliser notre architecture. Ensuite, nous détaillons le modèle du système que nous proposons.

5.2 Les agents intelligents

Un agent est une entité conceptuelle capable de réaliser des tâches au profit des utilisateurs. C'est une technologie issue du domaine de l'intelligence artifi-

cielle sous le vocable d'Agents Intelligents et qui s'impose comme un nouveau paradigme de structuration des logiciels.

Un agent est défini par *"une entité (physique ou abstraite) capable d'agir sur elle-même et son environnement, disposant d'une représentation partielle de cet environnement, pouvant communiquer avec d'autres agents et dont le comportement est la conséquence de ses observations, de sa connaissance et des interactions avec les autres agents"*[Ferber, 2002].

Beaucoup de caractéristiques peuvent être associées à un agent. Parmi elles, nous citons :

- L'autonomie : l'agent dispose de ses propres règles de contrôle qui déterminent ses désirs, c'est-à-dire, les objectifs à atteindre et les actions qu'il cherche à réaliser. L'autonomie signifie que l'agent peut réagir sans l'intervention directe de l'humain ou d'autres agents, il contrôle ses propres actions ainsi que son état interne.
- L'adaptabilité : un agent adaptatif est un agent capable de contrôler ses aptitudes selon l'agent avec qu'il interagit.
- La communication : l'agent interagit avec l'utilisateur ou avec un autre agent afin de recevoir des instructions pour accomplir sa tâche, de l'informer du statut et de la réalisation de la tâche. Cette communication s'effectue au moyen d'une interface Utilisateur-Agent ou d'un langage de communication agent.
- L'intelligence : l'agent est capable de prendre les décisions appropriées, de manière autonome.
- La Mobilité : l'agent mobile est capable de se déplacer d'un environnement à un autre. En effet, la mobilité n'est pas systématiquement envisagée mais qui devient nécessaire dès lors que l'on souhaite mouvoir des agents à travers une infrastructure distribuée.

5.2.1 Les agents dans le marché électronique

Les enchères est un mécanisme largement utilisé dans les marchés électroniques. Ce modèle de fonctionnement a été automatisé par l'introduction de la technologie d'agents. En effet, les agents sont utilisés dans l'automatisation de plusieurs services à savoir [Boll *et al.*, 1999] :

- Les services de marché qui supportent les processus d'achat, de vente, d'authentification, d'informations. Ce sont les services avec lequels interagissent l'utilisateur (vendeur/acheteur).
- Les services de base dont le système a besoin comme l'accès aux bases de données des utilisateurs, des produits, des enchères, etc.
- Les services commerciaux qui représentent les modules réalisant quelques

fonctionnalités du marché et qui sont utilisés par les services de marché. Parmi ces services, nous citons : la notification, la recherche de vendeurs, la recherche d'offres.

5.2.1.1 Quelques exemples

Parmi les système de marchè électroniques utilisant les agents et adoptant le mécanisme d'enchère comme modèle de fonctionnement, nous citons : **Kasbah**[Chavez et Maes, 1998], **AuctionBot**[Wurman *et al.*, 1998], et **MOPPET**[Arpinar *et al.*, 2000].

- **Kasbah** : est un système développé au sein du laboratoire MIT. Les agents proposés permettent l'automatisation des services de recherche de produits et le service de négociation entre les participants. Le mécanisme de négociation simple est une sorte d'enchère anglaise.
- **AuctionBot** : est un système d'enchère développé au sein de l'université de Michigan et qui permettent aux agents de participer à des enchères de différents types tel que l'enchère anglaise, l'enchère hollandaise et l'enchère de Vickrey.
- **MOPPET** : est une architecture de place de marché électronique proposér par l'université de Middle East Technical University en turquie. Le système est composé de plusieurs types d'agents qui sont : des agents d'interfaces, des agents de négociation et des agents intermédiaires qui jouents le rôle de lien entre les besoins des acheteurs et les offres de vente des vendeurs.

5.3 Le langage AUML

Le langage UML (Unified Modeling Language) est insuffisant pour modéliser les systèmes à base d'agents. Pour cela, des spécifications techniques ont été mises en place pour supporter la totalité du processus de développement, de la planification, analyse et désigne jusqu'à la construction et la maintenance. Dans ce contexte, AUML (Agent Unified Modeling Language) [1] est une adaptation des notations d'UML pour décrire la modélisation orientée agent. Les modifications proposées à UML sont :

- Un diagramme de classes agents qui est une reformulation du diagramme de classes objets.
- Un diagramme de séquences qui permet une meilleure modélisation des actions et interactions entre agents.
- Un diagramme de collaborations qui complémente le diagramme de séquences en proposant une autre lecture et vision des actions entre agents.

[1] www.auml.org

Afin de modéliser notre système, nous avons utilisé le langage AUML. Agent UML (AUML) qui est d'une manière précise une extension du langage UML pour la technologie d'agents.

5.4 Architecture proposée

Pour simuler le mécanisme d'enchères combinatoires, nous proposons un système à base d'agents. La figure 5.1 montre les composantes essentielles de l'architecture de notre système qui sont :

- un agent administrateur,
- un agent acheteur,
- un agent vendeur,
- un agent base de données
- un agent Tabou.

Le diagramme de classe correspondant est donné dans la Figure 5.2.

5.4.1 Agent administrateur

L'agent administrateur est le superviseur du système. C'est une entité stationnaire qui constitue le noyau de notre système. Ses tâches consite à :

- veiller sur le bon fonctionnement du système.
- lancer l'enchère,
- arrêter l'enchère,
- lancer l'agent tabou,
- faire la coordination entre les agents du système.

La structure de données liée à cet agent peut être résumée comme suit :

- Idf_Agent_Admin : est l'identificateur de l'agent administrateur.
- All_agent_list : est la liste des agents constituant le système.
- Offre_list : est la liste de toutes les offres proposées.
- Début_ench : est la date du début de l'enchère.
- Fin_ench : est la date de fin de l'enchère.

5.4.2 Agent acheteur

L'agent acheteur est l'interface entre le système et l'acheteur humain. Il traite toutes les demandes d'achat formulées par ce dernier. Son rôle consiste en :

- La récupération des mots de passe et les noms des acheteurs connectés au système.
- L'enregistrement des offres formulées par les acheteurs.

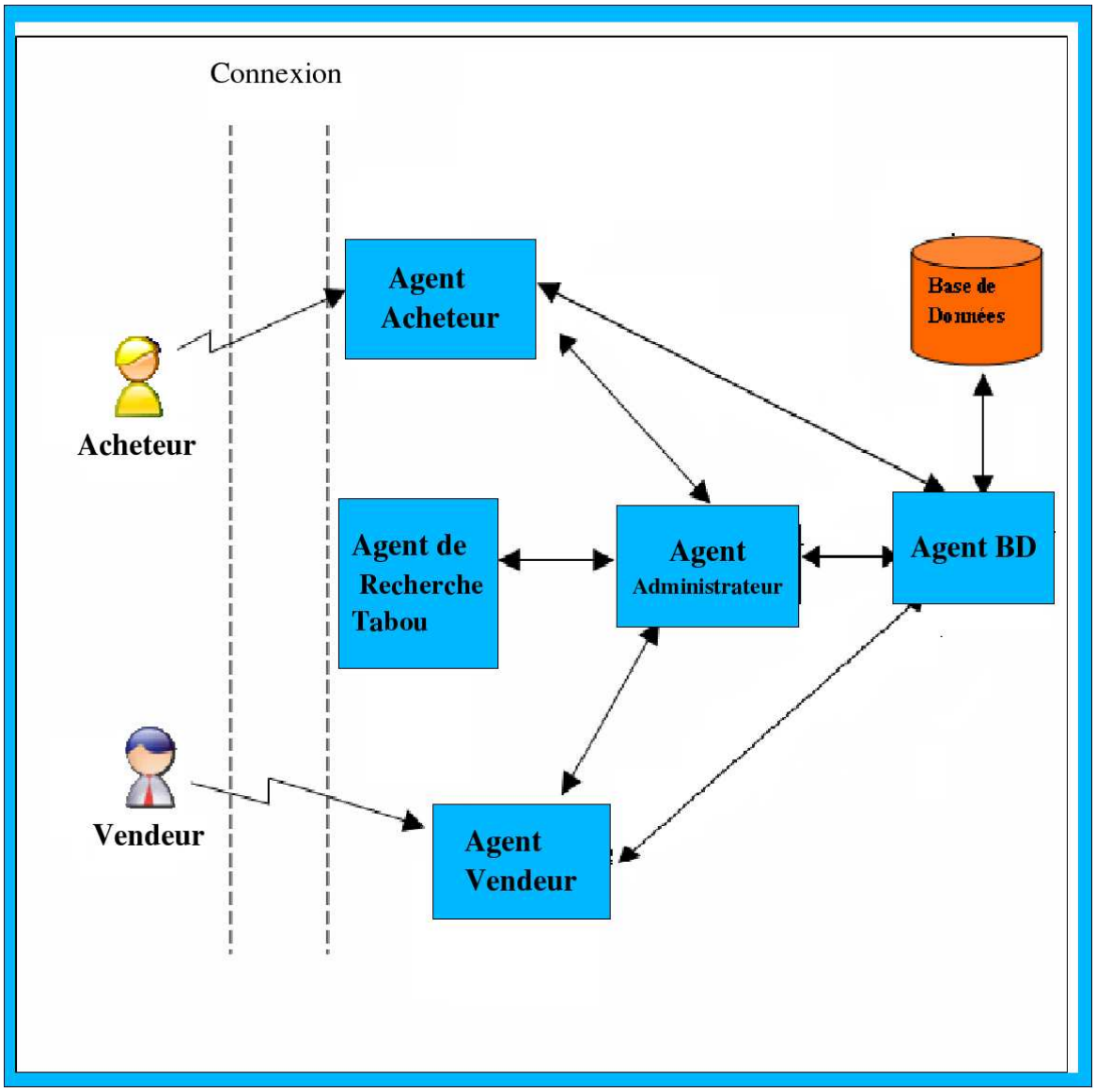

FIG. 5.1 – Architecture proposée du système d'enchères combinatoires

Pour ce faire, l'agent acheteur dispose de la structure de données suivante :

- Idf_Agent_Acheteur : est l'identificateur de l'agent acheteur.
- Offre_Acheteur : est l'offre d'achat de l'acheteur.
- Idf_Acheteur : est l'identificateur de l'acheteur (son mot de passe).
- Nom_Acheteur : est le nom de l'acheteur.
- Idf_Administrateur : l'identificateur de l'agent Administrateur.
- Idf_Ag_BD : l'identificateur de l'agent base de donnée.

5.4.3 Agent vendeur

C'est l'interface entre le système et le vendeur humain. Son rôle principal consiste à lancer l'offre de vente formulée par le vendeur. Parmi ses tâches nous citons encore :

- L'enregistrement des offres formulées par les vendeurs.
- Afficher l'état de l'enchère (meilleure offre, fin de l'enchère, ...).

L'agent vendeur disposera de la structure de données suivante :

- Idf_Agent_Vendeur : est l'identificateur de l'agent vendeur.
- Offre_vendeur : est l'offre de vente du vendeur.
- Idf_vendeur : est l'identificateur du vendeur (son mot de passe).
- Nom_vendeur : est le nom du vendeur.

5.4.4 Agent gestionnaire de base de données

Nous avons jugé utile l'utilisation d'un agent pour maintenir la masse d'informations qui circulent entre les agents constituant notre système. Notre base de données est un ensemble de tables utilisées pour sauvegarder les informations concernant les acheteurs, les vendeurs ainsi que leurs offres.

Afin de faciliter la gestion de la base de données et optimiser le temps de recherche, nous avons utilisé un agent qui gère cette dernière. On aura donc pour la classe agent de base de données les attributs suivants :

- Idf_Ag_BD : est l'identificateur de l'agent base de données (Agent BD).
- Idf_Ag_AD :est l'identificateur de l'agent Administrateur.
- Idf_Ag_V : est l'identificateur de l'agent Vendeur.
- Offre_Ag_V : est l'offre de vente du vendeur.
- Idf_Ag_A : est l'identificateur de l'agent Acheteur.
- Offre_Ag_A : est la proposition d'achat de l'acheteur.

5.4.5 Agent tabou

L'agent tabou est l'agent qui évalue les offres en se basant sur l'algorithme de la recherche taboue. L'objectif consiste à sélectionner les offres gagnantes maximisant le gain du vendeur. L'algorithme utilisé est détaillé dans le chapitre 4.

L'agent tabou disposera des informations suivantes :

- Idf_Ag_tabou : l'identificateur de l'agent tabou.
- Idf_Ag_admin : l'identificateur de l'agent Administrateur.
- Liste_offre : la liste des offres proposées.
- max_trials : est le nombre des exécutions.
- max_iter : est le nombre maximum des itérations.
- $iter\_best$: est le numéro de la meilleure itération.
- A^* : est la meilleure solution d'une exécution courante avec le coût F^*
- A^{**} : est la meilleure solution trouvée durant toutes les exécutions avec le maximum revenue F^{**}.
- TL est la liste taboue.

Il existe d'autres entités dans notre système et qui jouent un rôle important dans son fonctionnement : la classe offre, la classe produit et la classe client.

5.4.6 Classe Offre

La classe offre est la classe qui définie les offres de l'enchère. La classe offre possède les attributs suivants :

- Idf_offr : est l'identificateur de l'offre.
- Date_prop_offre : est la date de proposition de l'offre.
- Date_fin_offre : est la date de fin de l'enchère.
- Idf_client : est l'identificateur de client.
- List_prduit : est la liste des produits de l'offre.
- List_prix : est la liste des prix de l'offre.
- Prix_total : est le prix total de l'offre.
- Type_offre : est le type de l'offre (offre d'achat ou offre de vente).

5.4.7 Classe produit

La classe "produit" est la classe qui décrit le produit. Elle possède les attributs suivants :

- Idf_prod : est l'identificateur du produit.
- Nom : est le nom du produit.
- Marque : est la marque du produit.
- Descrip : est la description du produit (comme : le numéro de série, la date de fabrication,..)

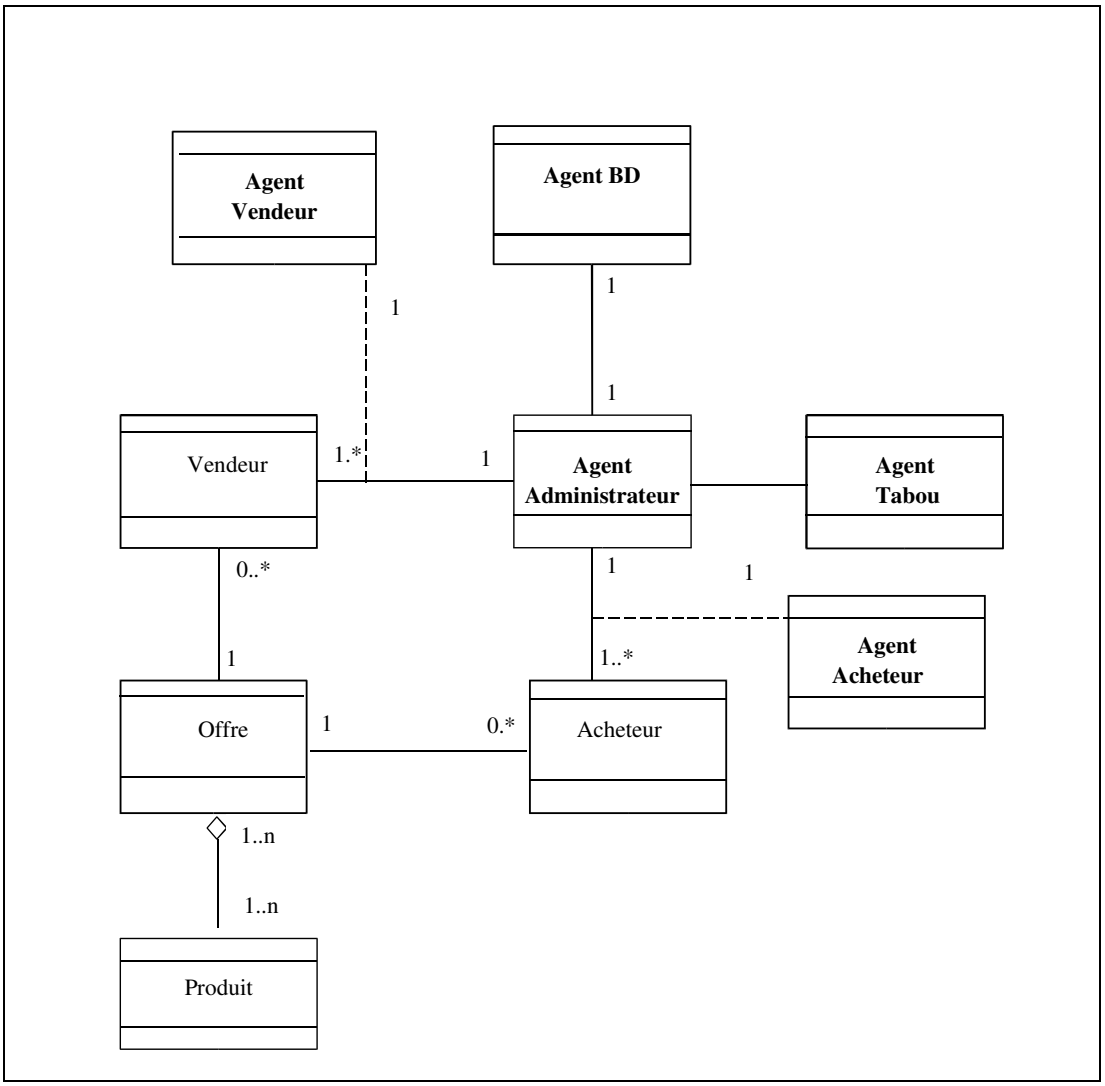

FIG. 5.2 – Diagramme de classe de l'Architecture Proposée

5.4.8 Classe Client

La classe client est une classe comportant le profil de chaque client du site. Un client peut être un vendeur qui souhaite vendre ses objets ou un acheteur cherchant à avoir des biens. La classe client possède la description suivante :

- Idf_client : est l'identificateur du client.
- Mot_passe : est le mot de passe du client.
- Nom : est le nom du client.
- Prénom : est le prénom du client.
- Adresse : est l'adresse physique du client.
- email : est l'adresse email du client
- Organisation : est l'organisation du client(société ou particulier).

5.5 L'étude fonctionnelle

Dans cette partie nous présentons le fonctionnement de notre système en montrant ses activités principales et les diagrammes de séquence correspondants.

Parmi les activités principales du système proposé, nous citons :

- La proposition d'une offre : cette activité permet à un vendeur (respectivement un acheteur) de formuler son offre de vente (respectivement d'achat), sachant qu'une offre concerne des paquets d'objets différents.
- La consultation : un vendeur (respectivement un acheteur) peut avoir une idée sur l'historique des offres ainsi que le résultat des enchères.
- L'inscription sur site : cette activité permet à un client de s'inscrire sur le site. Un client peut être un vendeur cherchant à vendre ses objets ou un acheteur cherchant à acquérir un bien.
- La gestion d'offres : cette option nous permet de modifier, améliorer ou supprimer une offre déjà enregistrée dans le système.
- L'évaluation des offres : l'agent tabou (utilisant le processus de recherche tabou décrit dans le chapitre 4) permet d'évaluer les offres soumises et essaie de déterminer la combinaison gagnante. Le but du vendeur étant de vendre ses objets de sorte à maximiser ses gains. Le problème de la détermination du gagnant PDG consiste alors à trouver les offres d'achat gagnantes maximisant le gain du vendeur.
- L'authentification : l'authentification des clients est assurée par l'agent administrateur, et ce pour assurer la sécurité des participants. L'utilisation des mots de passe permet l'accès uniquement aux clients autorisés.

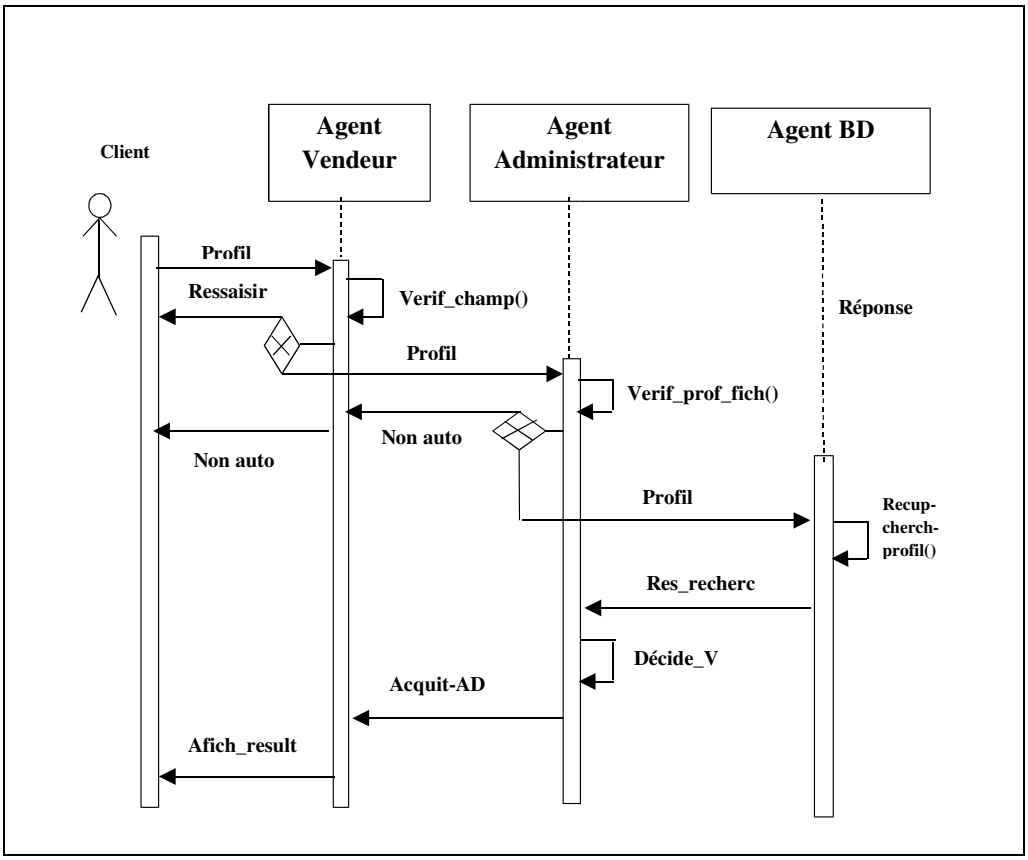

FIG. 5.3 – Diagramme de séquences correspondant à l'authentification

5.5.1 Authentification

Le diagramme de séquence correspondant à l'activité d'authentification est donné dans la Figure 5.3. Comme le montre le diagramme de séquence, l'agent administrateur, l'agent vendeur et l'agent BD collaborent pour accomplir la tâche d'authentification d'un vendeur. le plan d'action de chaque agent est donné comme suit :

Pour l'agent vendeur :

Le plan d'action :

- Verif_champ () : est l'action de vérification nécessaire pour chaque tentative d'entrée.
- Open_session () : est l'action d'ouverture d'une session.

Événement :

- Profil : est le profil envoyé par le client.
- Non auto : est l'évènement qui correspond à un client non autorisé.
- Acquit_AD : est l'autorisation d'accès au site.

Pour l'agent administrateur :
Le plan d'action :
- Verif_prof_fich() : est l'action qui vérifie si un client est autorisé ou non.
- Décide_reponse() : est la possibilité d'ouvrir une session ou non.

Événement :
- Profil : est l'évènement qui permet à l'agent vendeur d'envoyer à l'agent administrateur le mot de passe et l'identificateur du vendeur.
- Res_recherch : est l'évènement qui permet à l'agent BD d'indiquer à l'agent administrateur si un vendeur existe dans la base de données ou non.

Pour l'agent BD :
Le plan d'action :
- Recup_cherch_profil() : est l'action qui permet à l'agent BD d'effectuer une recherche dans sa base de données pour chercher l'identificateur du vendeur.

Événement :
- Profil : est l'évènement qui permet à l'agent administrateur d'envoyer l'identificateur du vendeur.

5.5.2 Proposition d'une offre

Deux types d'offres peuvent être formulées : une offre de vente par un vendeur ou une proposition d'achat par un acheteur. Le diagramme de séquence correspondant à l'offre de vente (cas du vendeur) est donné dans la figure 5.8. L'agent administrateur, l'agent vendeur et l'agent BD participent à cette activité.

Pour l'agent vendeur :
Le plan d'action :
- Recup_off_acht () : est l'action qui récupère l'offre proposée par le vendeur. Elle est déclanchée à l'arrivée de l'évènement : Offr_acht.
- Control_form_offr() : est l'action qui permet à l'agent vendeur de 1vérifier si le formulaire d'offre est bien rempli.

Évènement :
- Send_Requst_off : est l'évènement qui permet au vendeur d'envoyer sa proposition de vente.
- Offre_libre : est évènement permet de proposer une offre indépendamment de l'acheteur dite offre libre.
- Offre_acht : est évènement qui permet de récupérer l'offre de l'acheteur déjà enregistrée.
- Confirmation : est la confirmation de la transaction.

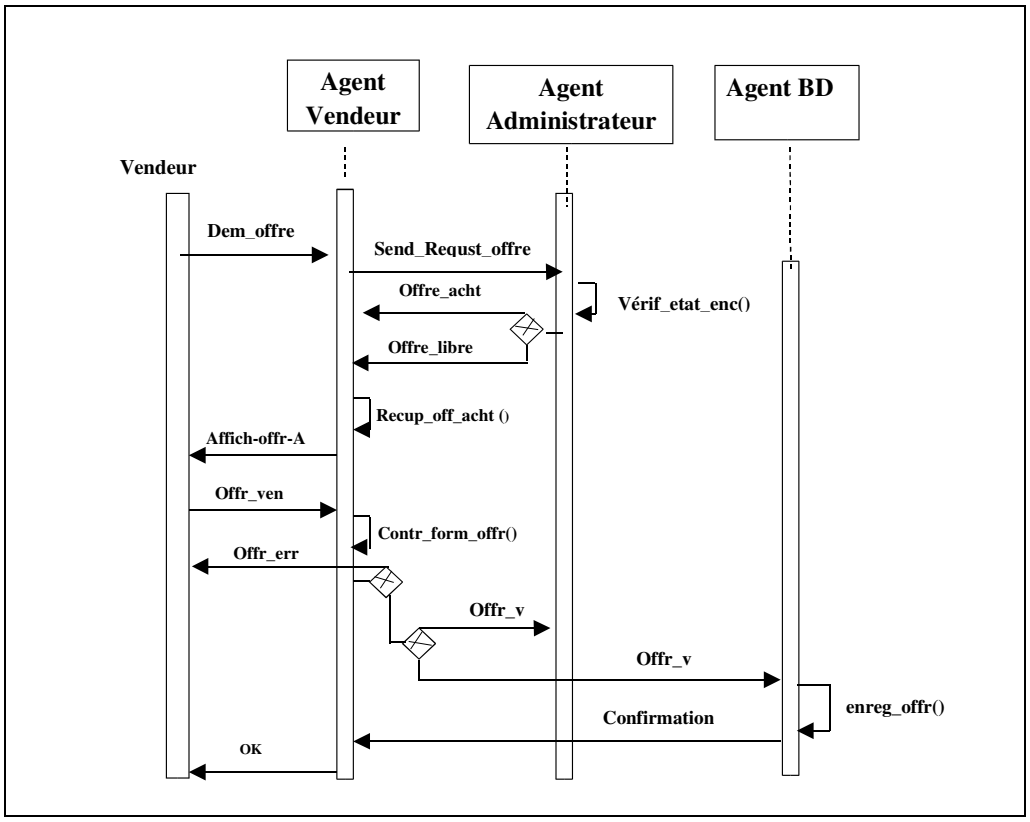

FIG. 5.4 – Diagramme de séquences correspondant à l'offre de vente

Pour l'agent administrateur :
Le plan d'action :
- Vérif_etat_enc() : est l'action qui vérifie l'état de l'enchère, c'est-à-dire fermée ou ouverte.

Évènement :
- Send_Requst_off : est l'évènement qui permet à l'agent vendeur d'envoyer son offre.
- Offr_v : est l'évènement qui permet à l'agent administrateur de recevoir l'offre de vente.

Pour l'agent BD :
Le plan d'action :
- enreg_offre() : est l'action d'enregistrement de l'offre de vente dans la base de données.

Évènement :
- Offre_v :est l'évènement qui permet à l'agent administrateur de recevoir l'offre de vente.

5.5.3 Evaluation des offres

Le diagramme de séquence correspondant à l'activité d'évaluation des offres est donné dans la Figure 5.5.

Pour l'agent administrateur :
Le plan d'action :

- dem_Eval_offr() : est l'action qui permet d'évaluer les offres participantes à l'enchère.

Évènement :
- Send_req_eval : est la demande d'évaluation des offres de l'enchère.
- Send_req_enrg : est la demande d'enregistrement du résultat de l'enchère dans la base de données.

Pour l'agent tabou :
Le plan d'action :
- Eval_offr() : est l'action qui évalue les offres en utilisant le processus de recherche taboue.

Évènement :
- gagnat_offr : est l'évènement indiquant les offres gagnantes.

Pour l'agent BD :
Le plan d'action :
- Enr_eval_offr() : est l'action d'enregistrement de résultat de l'offre.

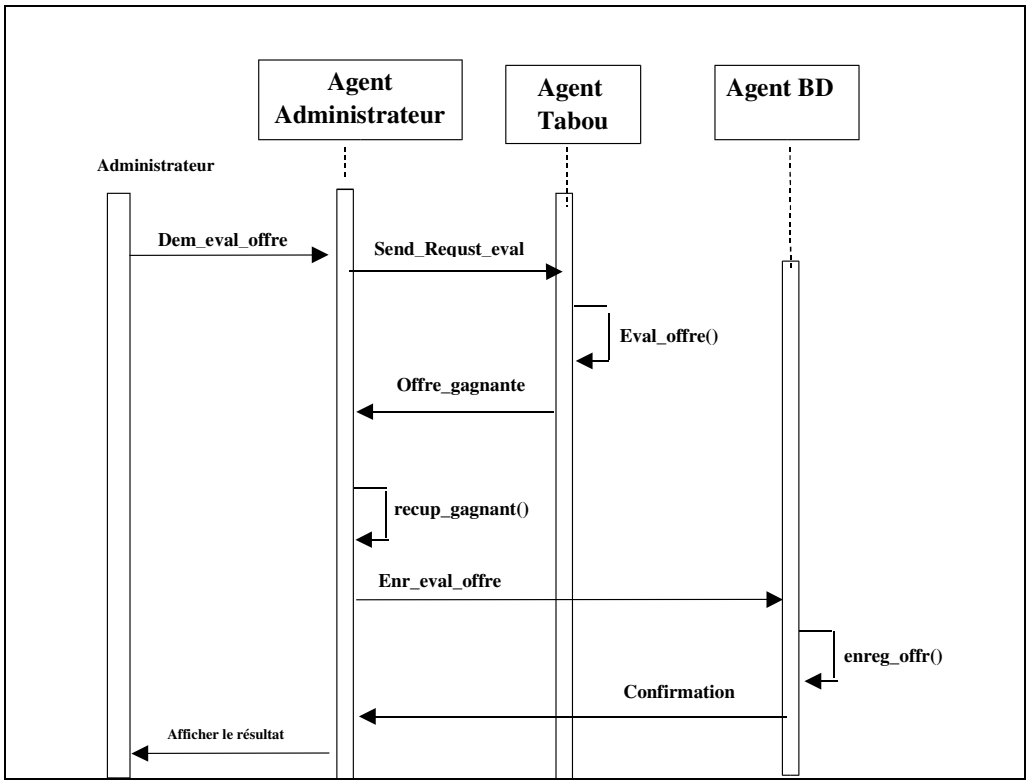

FIG. 5.5 – Diagramme de séquences correspondant à l'évaluation des offres

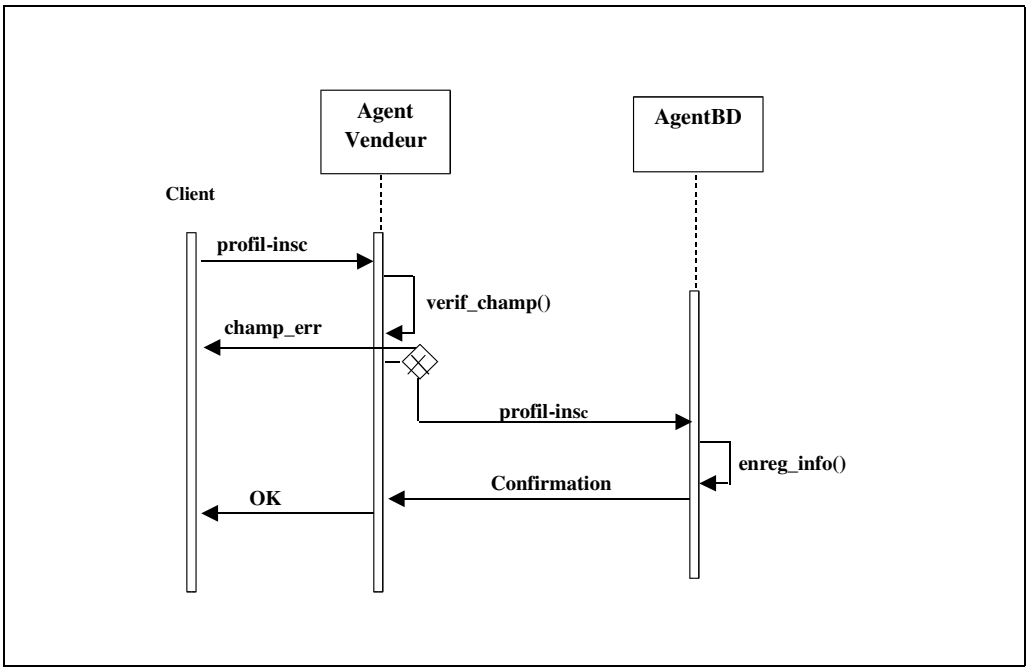

FIG. 5.6 – Diagramme de séquences correspondant à l'inscription

5.5.4 Inscription

Le diagramme de séquence correspondant à l'inscription est donné dans la Figure 5.6

Pour l'agent vendeur :
Le plan d'action : Évènement :

- enreg_info() : est l'action permettant à l'agent BD d'enregistrer le profil d'inscription des participants.

Évènement :

- profil_insc : ce sont les informations nécessaires pour l'inscription.

Pour l'agent BD :
Le plan d'action :

- profil_insc : est le profil d'inscription du client.

Évènement :

- verif_champ() : est l'évènement permettant de vérifier les champs de formulaire d'inscription.

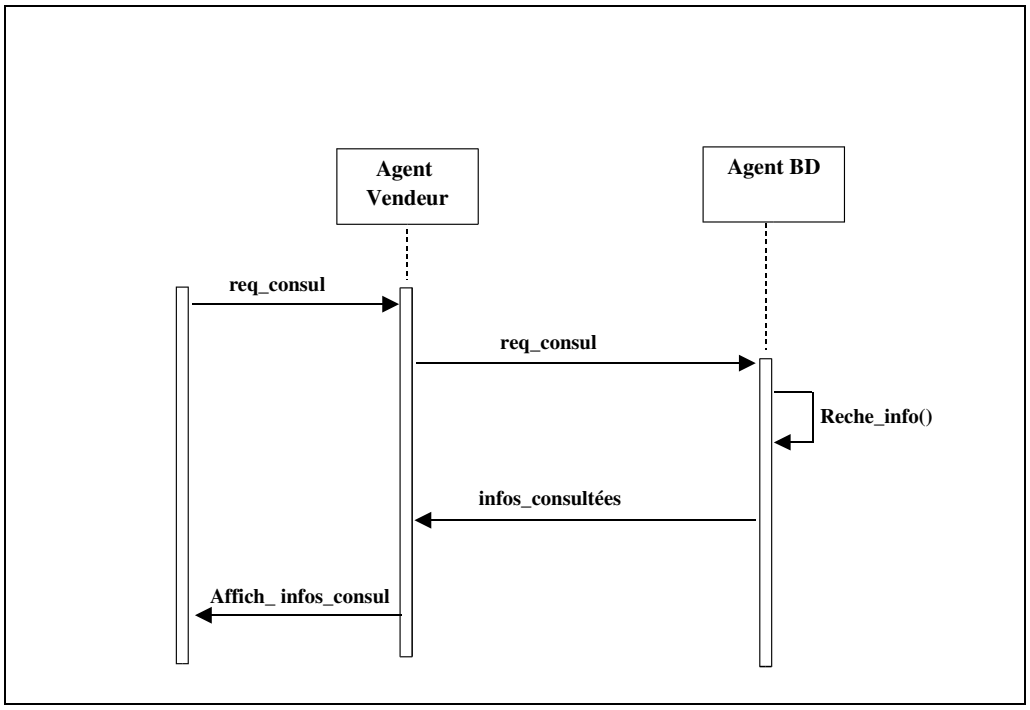

FIG. 5.7 – Diagramme de séquences correspondant à la consultation

5.5.5 Consultation

Le diagramme de séquence correspondant à la consultation est donné dans la Figure 5.7

Pour l'agent vendeur :
Évènement :

- dem_cons : est l'évènement indiquant la demande de consultation
- infos_consultées : est l'évènement indiquant à l'agent BD de faire une recherche des informations demandées

Pour l'agent BD :
Le plan d'action :

- reche_info() : est l'action permettant à l'agent BD de chercher les informations demandées par le vendeur (respectivement acheteur).

Évènement :

- dem_cons : est la demande de consultation envoyée par l'agent vendeur.

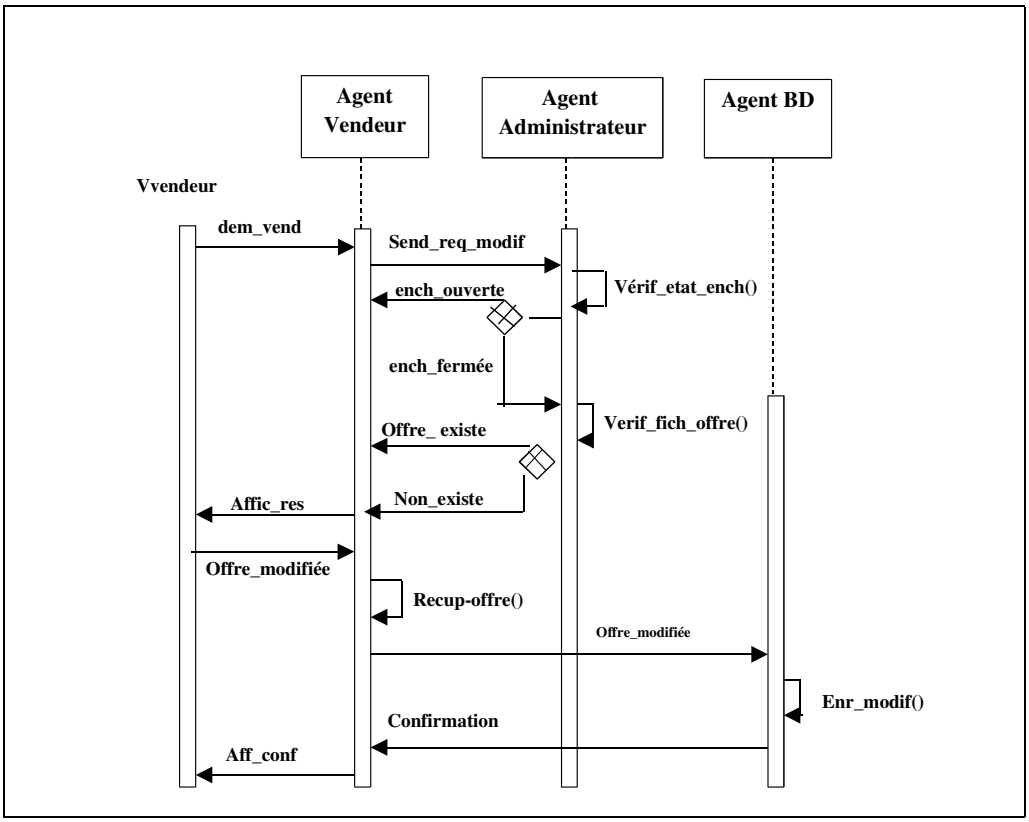

FIG. 5.8 – Diagramme de séquences correspondant à la Mise à jour

5.5.6 Mise à jour d'une offre

Le diagramme de séquence correspondant à l'activité de mise à jour est donné dans la Figure 5.8.

Pour l'agent vendeur :
Le plan d'action :

– Recup_offr() : est l'action qui permet de récupérer une offre.

Évènement :

– dem_vend : est la demande du vendeur pour modifier une offre.
– ench_ouvert : est l'évènement indiquant l'ouverture de l'enchère.
– existe_offre : est l'évènement indiquant l'existence de l'offre.
– not_exist : est l'évènement indiquant la non existence de l'offre.
– Offre_mod : est l'évènement indiquant la modification de l'offre.
– confirmation : est le message de confirmation.

Pour l'agent administrateur :
Le plan d'action :
- Vér_eta_ench() : est l'action qui vérifie l'état de l'enchère (fermée ou ouverte).
- Ver_fich_offr() : est l'action qui vérifie l'existence de l'offre.

Évènement :
- Send_req_mod : demande de modification d'une offre.

Pour l'agent BD :
Le plan d'action :
- Enr_mod() : est l'action d'enregistrement de la modification.

Évènement :
- offr_mod : est l'évènement indiquant la modification de l'offre.

5.6 Implémentation

Dans cette partie nous abordons l'aspect implémentation. Nous avons réalisé un site Web dynamique pour simuler le mécanisme d'enchères combinatoires. Nous donnons les technologies utilisées pour concevoir le site Web ainsi que la vue globale de notre application Web.

5.6.1 Technologies utilisées

Les différentes technologies existent pour développer notre prototype sont :
- Servlet : les servlets sont des petits serveurs écrits en langage Java et qui utilisent une API spécifique.
- JSP : Les JSP, ou Java Server Pages est du code Java que l'on incorpore dans la page HTML avec des techniques de scriptings. Le serveur compile automatiquement la page en un servlet et l'exécute ensuite.
- Collaboration des Servlet et JSP : Les servlets sont utilisées par les développeurs pour gérer les aspects programmations d'une application web et les JSP sont utilisés par les infographistes pour effectuer l'affichage.
- Tomcat : Tomcat est un conteneur d'application web basé sur Java, et au code source ouvert. Il exécute des applications web JSP et servlets. Tomcat Server est l'implémentation de référence des spécifications servlet et JSP.
- SQL Server : Microsoft SQL Server est un SGBD d'une grande qualité.
- Communication via les RMI : RMI (Remote Method Invocation) est une API Java permettant de manipuler des objets distants de manière transparente pour l'utilisateur, c'est-à-dire de la même façon que si l'objet était sur la machine virtuelle (JVM) de la machine locale.

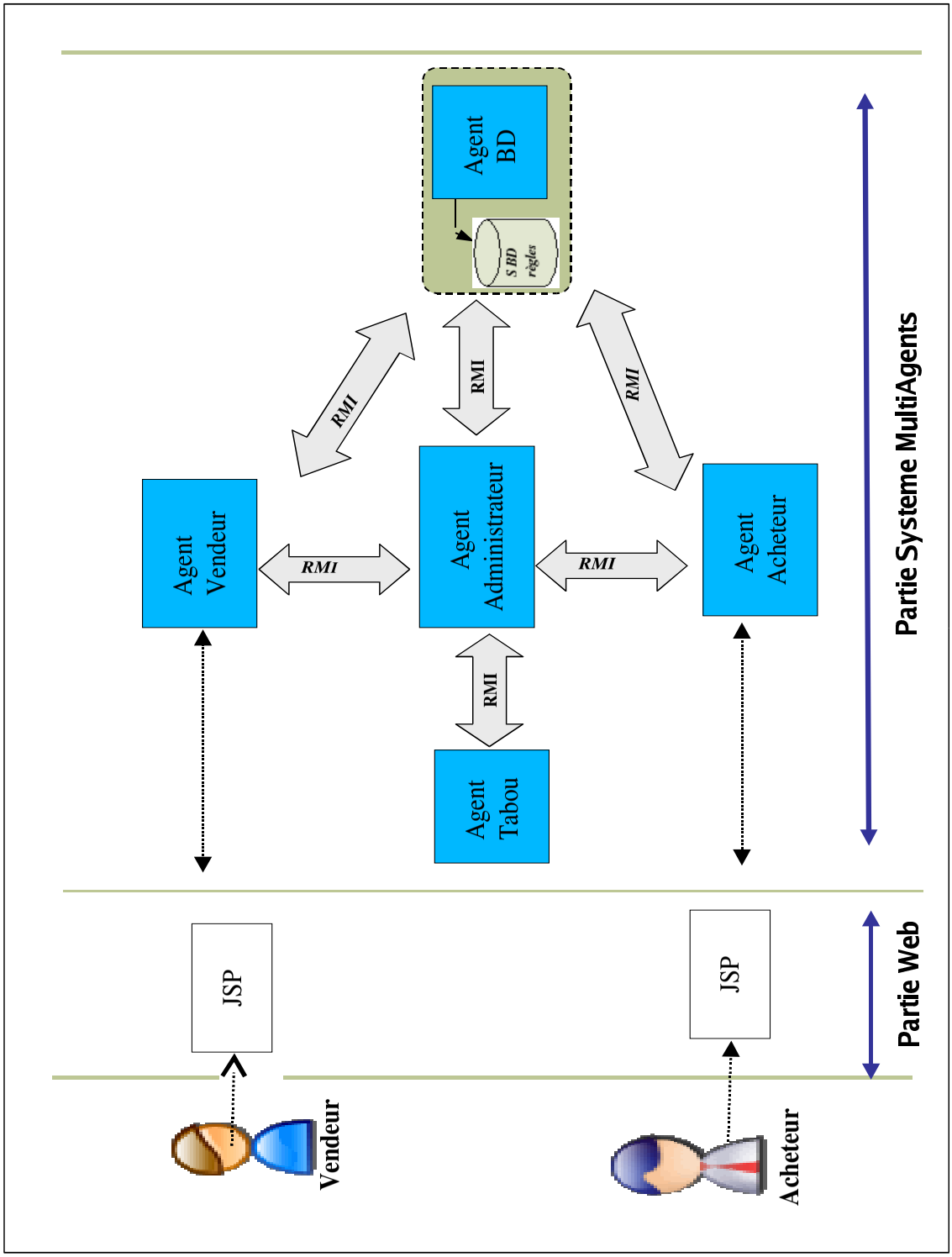

FIG. 5.9 – Application Web du NetCombinatoire

5.6.2 Application Web du NetCombinatoire

Comme le montre la figure 5.9, notre système est composé de deux parties :
- La partie SMA : qui est la partie invisible de notre application Web. C'est la partie traitement qui utilise des agents communicants via les RMI. Cette partie est déjà déatillée en *paragraphes 5.4 et 5.5.*
- Partie Web : qui représente le modèle de vue de l'application c'est-à-dire les différentes interfaces de l'application, sous formes de pages JSP.

5.6.2.1 Benchmarks utilisés

Pour valider notre système, nous avons généré quelques benchmarks aléatoires représentant des problèmes du détermination de gagnant. Les bechmarks utilisés pour la validation du système de Netcombinatoire sont des problèmes de petite taille où la solution optimale est possible à obtenir en un temps reasonable. Notre objectif principal est de voir comment nos agents collaborent et intéragissent au sein de la plateforme proposée, et ce pour trouver une solution optimale aux problèmes considérés.

5.6.2.2 Interfaces Web du NetCombinatoire

Les principales interfaces graphiques de notre site Web sont : la page d'autentification, la page accueil, la page vente, la page achat et la page historique. Nous avons utilisé les servlets et les JSP pour créer nos pages Web. Les servlets sont utilisées pour gérer les aspects programmations et les JSP sont utilisés pour effectuer l'affichage.
- La page d'authentification contient deux zones, une pour le login et l'autre pour le mot de passe. Cette page permet aux utilisateurs déjà inscrits l'accès aux fonctionalités du site Web, et ce dans le cas où leur login est correct. Pour avoir un login l'utilisateur doit s'incrire tout d'abord sur le site en appuyany sur un lien hypertext qui activera le formulaire d'inscption.
- La page accueil contient toutes les options du site à savoir : la vente, l'achat, la proposition d'offres, le résultat d'enchères ainsi que l'historique du site.
- La page vente est une page dynamique qui permet aux vendeurs de proposer une offre de vente.
- La page achat est une page dynamique similaire à la page vente. Elle permet aux acheteurs de formuler leur offres d'achat.
- La page historique est une page dynamique permet d'afficher l'historique des utilisateurs du site, toutes les offres effectuées ainsi que les résulats d'enchères passées.

5.7 Conclusion

Dans ce chapitre nous avons proposé un système à base d'agent pour simuler le mécanisme d'enchères combinatoires. Pour évaluer les offres combinées, nous avons utilisé un agent de recherche taboue. Nous avons choisi la technologie d'agents pour l'automatisation du processus d'enchères sur Internet car nous pensons que le paradigme agent est un outil idéal pour ce genre d'applications. Néanmoins, la sécurité des agents, la transaction et le payement sont des points importants qui doivent être approfondis ultérieurement.

Conclusion

Les problèmes d'optimisation combinatoire auxquels nous nous sommes intéressés sont réputés difficiles : consistant en la recherche d'une solution de meilleure qualité dans un ensemble fini mais de très grande cardinalité donc non énumérable. A leur appartenance à la classe des problèmes *NP-Complets*, s'ajoute également le fait que souvent n'ont été résolus que des problèmes de petite taille.

Dans cette thèse, nous avons étudié deux problèmes difficiles : le problème de satisfiabilité *MAX-SAT* et le problème de la détermination du gangnat dans les enchères combinatoires (*PDG*) qui sont deux problèmes *NP-Complets*.

Le problème de la satisfiabilité maximale *MAX-SAT* est le problème central de la NP-Complétude [Cook, 1971]. Il présente un double intérêt : d'une part, il apporte une réponse à de nombreux problèmes de décision se posant souvent, de façon cruciale, dans des domaines très divers. D'autre part, il nous a permis de montrer comment construire des algorithmes performants pour des problèmes très ardus.

Dans la première partie de cette thèse, nous avons proposé des approches évolutionnaires pour la résolution du problème *MAX-SAT*.

Nous avons proposé une nouvelle stratégie de sélection basée sur la diversité et la qualité pour choisir une collection de solutions qui participent à la phase de reproduction et donnent une descendance. Ensuite, nous avons utilisé un opérateur de combinaison spécifique au problème *MAX-SAT* pour générer de nouvelles solutions qui sont améliorées par une recherche locale stochastique (SLS).

La principale motivation est de realiser un compromis entre la diversification et l'intensification, ce qui permet d'éxplorer efficacement l'espace de recherche. En effet, le croisement est un mécanisme de diversification alors que la recherche locale est un mécanisme puissant d'intensification.

Les trois composantes proposées ont été incorporées, tout d'abord, dans un algorithme génétique. Ensuite, une variante de recherche dispersée, utilisant les trois composantes déjà citées, a été étudiée pour le problème *MAX-SAT*.

Les expérimentations reportées au chapitre 3, nous ont permis de clarifier ce que l'on peut attendre des approches évolutionnaires, et de confirmer l'existence des méthodes capables de produire d'excellents résultats.

Dans nos tests, nous avons utilisé des données *MAX-SAT* dont le degré de difficulté est assez important ceci est dû essentiellement à la nature de ces problèmes et le nombre important d'optima-locaux dans l'espace de solutions. Ces instances se caractérisent par un nombre de variables compris entre 60 et 3176 et un nombre de clauses compris entre 100. et 95904.

Ce qu'il faut noter aussi en ce qui concerne les solutions trouvées, c'est l'écart entre ces dernières et la solution optimale qui est négligeable dans certaines instances, ce qui prouve la puissance des approches proposées dans la résolution du problème *MAX-SAT*.

Les résultats obtenus par nos approches évolutionnaires sont très encourageants, montrant l'avantage de fusionner les composants principaux : la stratégie de sélection, l'opérateur de croisement spécifique au problème *MAX-SAT* et la recherche locale (SLS). Plusieurs expérimentations menées sur des benchmarks et des comparaisons avec des algorithmes génétiques, Walkasat, FlipGA et GASAT montrent l'efficacité de notre nouvelle approche. En perspective, nous souhaitons regarder l'hybridation d'algorithmes complets et d'algorithmes de recherche locale ou évolutionnaires.

Dans la deuxième partie de cette thèse, un algorithme de recherche taboue a été développé pour résoudre le problème de la détermination du gagnant (PDG) qui est un problème *NP-Complet* [Rothkopf *et al.*, 1998].

La méthode taboue que nous avons proposée démarre d'une solution créée aléatoirement et selon le codage de clef aléatoire, RK (Random Key Encoding RK, puis la meilleure allocation voisine est sélectionnée pour être la prochaine solution courante. Pour produire des solutions voisines, nous avons défini deux mouvements qui sont : "On-Bid" et "On-Item".

Afin d'échapper aux allocations déjà visitées, notre stratégie de recherche utilise une liste pour maintenir les numéros d'offres récemment choisies. Ces numéros sont dits "mouvements tabous". Quand un mouvement tabou appliqué à une allocation courante donne une meilleure solution ; nous acceptons ce mouvement malgré son statut tabou par critère d'aspiration. Une étape de diversification est lancée pour explorer de nouvelles régions, ce qui évitera la stagnation du processus de recherche. La recherche s'arrête après un certain nombre d'itérations fixé d'une manière empirique.

Des expérimentations numériques ont été reportées au chapitre 4 dans le but de tester et de prouver l'efficacité de notre approche. Une étude comparative avec quelques algorithmes de l'état de l'art concernant PDG à savoir $Casanova$ et $SAGII$ est établie dans ce chapitre.

Nous avons constaté que la recherche taboue a prouvé sa performance grâce à l'inclusion de plusieurs techniques de diversification et d'intensification. Les résultats respectifs sont acceptables et meilleurs que ceux de Casanova.

Dans le chapitre 5, nous avons simulé le mécanisme d'enchères combinatoires au moyen d'un système multiagents. Les agents offrent au processus commercial sur le Web l'automatisation et l'autonomie. Dans un tel environnement, les agents vendeurs et acheteurs ont la possibilité de lancer des offres sur un ensemble d'objets provenant d'une base finie. L'agent vendeur reçoit les demandes des agents acheteurs pour des paquets d'objets différents. Son but étant d'allouer ces objets de sorte à maximiser ses gains. Il s'agit d'un problème d'optimisation, c'est pourquoi nous avons proposé un agent de recherche taboue pour évaluer les offres afin de trouver une solution optimale au problème.

5.7.1 Perspectives

Nos recherches vont naturellement se prolonger, dans les trois voies suivantes :

- Concevoir un '*Solver*' efficace pour les problèmes *SAT*, en hybridant une méthode systématique et une recherche locale.
- Prolonger l'étude des approches évolutionnaires proposées. En général, il reste à élaborer des algorithmes performants pour résoudre le problème de la détermination du gagnant (PDG)
- Renforcer la sécurité du système d'agents proposé.

Bibliographie

[Alsinet *et al.* 2003] T. Alsinet, F. Manya, et J. Planes. "Improved branch and bound algorithms for MAX-SAT" *In Proceedings of the 6th International Conference on the Theory and Applications of satisfiability Testing. SAT2003*, pp : 408-415, 2003.

[Anderson *et al.* 2000] A. Anderson, M. Tenhunen et F.Ygge. "Integer Programming for Combinatorial Auction Winner Determination", *In Proceedings of 4th International Conference on Multi-Agent Systems,* IEEE Computer Society Press, July, pp : 39-46, 2000.

[Arpinar *et al.* 2000] S.Arpinar, A.Dogac et N.Tatbul. "An open electronic marketplace through agent-based workfolows : Moppet", *In International Journal on Digital Libraries,* vol 3, $n1$, pp : 36-59, 2000.

[Banks *et al.* 1989] J.S. Banks, J.O. Ledyad et D.P. Porter. "Allocating uncertain and unresponsive ressource : an experimental approach", *In the RAND Journal of Economics,* vol 20, $n1$, pp : 1-25, 1989.

[Bean 1994] J.C. Bean. "Genetics and random keys for sequencing and optimization", *In ORSA Journal of Computing*, Vol 6, $n2$, pp : 154-160, 1994.

[Benhamou et Sais 1992] B. Benhamou , L.Sais. 1992. "Theoretical study of symmetries in propositional calculus and applications". *In Proceedings of the CADE'92*, pp : 281-294, 1992.

[Boughaci *et al.*2008] D. Boughaci, B.Benhamou and H. Drias, "Genetic Algorithms and Scatter Search for the MAX-SAT Problems", *In : Journal of Mathematical Modelling and Algorithms,*, Vol 7, pp : 101-124. Février 2008. Published by Springer.

[Boughaci et Drias 2005 b] D. Boughaci, H. Drias, "Taboo search as an intelligent agent for bid evaluation", *In : International Journal of Internet and Enterprise Management,* vol 3, $n2$, pp : 170-186, 2005. Published by Inderscience Enterprises Ltd., Geneva, Switzerland.

[Boughaci *et al.*2005] D. Boughaci, H. Drias, B. Benhamou, "Two Hybrid Tabu Scatter Search Meta-heuristic for Solving MAX-SAT Problems", *In : Alexander Gelbukh, Raul Monroy (Eds.), a special issue on Advances in Artificial*

Intelligence Theory, Journal of Research on Computing Sciences (RCS), ISSN 1665 - 9899, n16, pp : 45 - 54, 2005.

[Boughaci et Drias2005 c] D. Boughaci, H. Drias, "Stochastic Search Methods for MAX-SAT Problems" , *In GESTS International Transaction on Computer Science and Engineering*. Editor : K.E. Bae, S.B. Kim and S.Y. Lee GESTS Society, ISSN : 1738-6438, Vol 11, n1, pp : 13-20, June 2005.

[Boughaci et Drias2005 a] D. Boughaci , H.Drias "Efficient and Experimental Meta-heuristics for MAX-SAT Problems". *In Lecture Note in Computer Sciences, Volume 3503 / 2005,* WEA 2005 : pp :501-512, 2005.

[Boughaci *et al.*2007 a] D. Boughaci, B. Benhamou, H. Drias, "IGA : an Improved Genetic Algorithm for MAX-SAT Problems", *In : Indian International Conference on Artificial Intelligence (IICAI)-07*, pp : 132- 150, Pune, India. Décembre 2007.

[Boughaci *et al.*2007 b] D. Boughaci, B. Benhamou, H. Drias, "Solving MAX-SAT Problems using a Scatter Search Metaheuristic", *In : Proceddings of the COSI'07* , pp :167-178, 11-13 Juin 2007.

[Boughaci *et al.*2007 c] D. Boughaci, B. Benhamou, H. Drias, "Une recherche disperse pour le problème MAX-SAT", *In Proceddings : RJCIA'07*, 2-6 Juillet 2007.

[Boughaci *et al.*2007 d] D. Boughaci, H. Drias, B. Benhamou, "An ebXML-based Multiagent System for B2B eCommerce", *In the proceedings of the 22nd ACM Symposium on Applied Computing*, pp : 659- 663, 2007.

[Boughaci et Drias 2004 a] D. Boughaci, H. Drias, "A Performance Comparison of Evolutionary Meta-heuristics and Solving MAX-SAT", *In : International Journal of Information Technology*, vol. 1, n1, pp :45-49, 2004.

[Boughaci *et al.*2006 a] D. Boughaci, B. Benhamou, H. Drias, "A Specific Genetic Algorithm for MAX-SAT problems", *In : META'06 workshop and school on Metaheuristics,* 2-3-4 Novembre 2006. Hamammat, Tunisie.

[Boughaci *et al.*2006 b] D. Boughaci, H. Drias, B.Benhamou, "Approches Méta-heuristiques pour les problèmes max-SAT difficiles", *In : 7ème congrés de la Société Française de Recherche Opérationnelle et d'Aide à la Décision "Roadef06"*, 6, 7, 8 Février 2006. LILLE.

[Boughaci *et al.*2005 a] D. Boughaci, H. Drias, B. Benhamou, "A Scatter Search Variant to solve max-SAT Problems", *In : Workshop, Combination of Methaeuristic and Local Search with Constraint Programming (MLS+CP)*, University of Nantes, France., novembre 2005.

[Boughaci *et al.*2005 b] D. Boughaci, H. Drias, B. Benhamou, "Combining a Unit Propagation with Genetic Algorithms to Solve max-SAT", *In : Workshop of*

Combination of Metaheuristic and Local Search with Constraint Programming (MLS+CP), University of Nantes, France, novembre 2005.

[Boughaci et Drias 2005 b] D. Boughaci, H. Drias, "An Agent-Based Approach using the ebXML Specifications for E-business", *In : International Conference on Information Technology : Coding and Computing* (ITCC'05), vol. 2, pp : 766-767, IEEE Computer Society, 4-6 April 2005 2005. Las Vegas, Nevada, USA.

[Boughaci et Drias 2004 d] D. Boughaci, H. Drias, "A Performance Comparison of Evolutionary Meta-heuristics and Solving MAX-SAT Problems", *In : International Conference on Computational Intelligence, ICCI 2004, Ali Okatan (Ed.)*, pp : 379-383, International Computational Intelligence Society 2004, December 17-19, 2004 2004. Istanbul, Turkey.

[Boughaci et Drias 2004 c] D. Boughaci, H. Drias, "PTS : A Population-based Tabu Search for the maximum satisfiability problems,", *In : The IEEE GCC Conference*, pp. 622-625, IEEE GCC, November 2004. Bahrain.

[Boughaci et Drias 2004 e] D. Boughaci, H. Drias, "Cooperative Agents and Taboo Search Meta-Heuristic to Support an ebXML-Based Platform", *In : the 11th (ISPE) International Conference on Concurrent Engineering (CE2004)*, pp : 229,234, springer and China press, 26-30 JULY 2004. BEIJING, CHINA.

[Boughaci et Drias 2004 b] D. Boughaci, H. Drias, "Solving weighted Max-Sat optimization problems using a Taboo Scatter Search metaheuristic", *In : the 2004 ACM Symposium on Applied Computing (SAC)*, Hisham Haddad, Andrea Omicini, Roger L. Wainwright, Lorie M. Liebrock (Eds.) :, pp : 35-36, ACM, March 14-17 2004. Nicosia, Cyprus.

[Boughaci *et al.*2004] D. Boughaci, H Drias et B. Benhamou, "Solving Max-SAT Problems Using a memetic evolutionary Meta-heuristic", *In Proceedings of 2004 IEEE CIS 2004,* pp : 480-484, 2004.

[Boughaci et Drias 2003] D. Boughaci, H. Drias, "Taboo Search as an Intelligent Agent for Bid Evaluation", *In : Concurrent Engineering (CE2003)*, pp : 43-47, 27-30 juillet 2003. Madeira Island, Portugal, 2003.

[Borchers et Furman 1999] B. Borchers , J. Furman. "A two-phase exact algorithm for Max-SAT and weighted Max-SAT problems". *In Journal of Combinatorial Optimization,* 2 (4), pp : 299-306, 1999.

[Chavez et Maes 1998] A.Chavez , P.Maes. "An Agent marketplace for buying and selling goods ". *In Procedding of the 1st International Conference on Electronic Commerce,IECE'98,* pp : Soeul, Korea, 1998.

[Cook 1971] S.A. Cook. "The complexity of theorem proving procedures". *In Procedding of the 3rd ACM symposium on Theory of Computing,* pp : 151-158, Ohio, 1971.

[Davis et Putnam 1960] M. Davis , H. Putnam. "A computing procedure dor quantification theory". *In Journal of ACM*, Vol 7, n3,1966.

[Davis *et al.*1962] M. Davis, G. Logemann, et D. Loveland. "A machine program for theorem proving". *In Communication of ACM*, 5, pp :394-397, 1962.

[Dorigo 1992] M. Dorigo. "Optimization, Learning and Natural Algorithms", *PhD thesis, Politecnico di Milano, Italie*, 1992.

[Dorigo 2006] M. Dorigo, M. Birattari, T. Stutzle. "Ant Colony OptimizationŽ : Artificial Ants as a Computational Intelligence Technique", *IEEE Computational Intelligence Magazine*, vol 1, n4, pp : 28-39, 2006.

[Drias 1993] H. Drias. "Approche probabiliste du dénombrement des solutions du problème MAX-SAT" *Thèse de doctorat d'état*, 1993.

[Drias 1998] H. Drias. "A Monte Carlo Algorithm for the Satisfiability Problem" *In Lecture Note in Computer Sciences, Volume 2070 IEA/AIE (Vol. 1)*, 1998, pp : 159-168.

[Drias 1999] H. Drias. "Randomness in Heuristics : An Experimental Investigation for the Maximum Satisfiability Problem" *In Lecture Note in Computer Sciences, Volume 1606 IEA/AIE (Vol. 1)*, pp : 700-708, 1999.

[Drias et Khabzaoui 2001] H. Drias , M.Khabzaoui, "Scatter Search with Random Walk Strategy for SAT and MAX-W-SAT Problems" *In Lecture Note in Computer Sciences, Volume 1415 IEA/AIE (Vol. 1)*, pp : 35-44, 2001.

[Drias et Ibri 2003] H.Drias, S.Ibri, "Parallel ACS for Weighted MAX-SAT" *In Lecture Note in Computer Sciences, Volume 1415 IWANN*, pp : 414-421, 2003.

[Drias *et al.*2003] H.Drias, A.Taibi et S.Zekour, "Cooperative Ant Colonies for Solving the Maximum Weighted Satisfiability Problem", *In Lecture Note in Computer Sciences, Volume 2686 IWANN*, pp : 446-453, 2003.

[Drias *et al.*2005] H. Drias, S.Sadeg et S.Yahi, "Cooperative Bees Swarm for Solving the Maximum Weighted Satisfiability Problem", *In Lecture Note in Computer Sciences, Volume 3512 IWANN* pp : 318-325, 2005.

[Frank 1994] J. Frank, "A study of genetic algorithms to find approximate solutions to hard 3CNF problems" *In Proceedings of Golden West international conference on artificial intelligence*,1994.

[Ferber 2002] J. Ferber, "Technologie multiagents" *Hermes*,2002.

[Fujishima *et al.*1999] Y. Fujishima, K. Leyton-Brown, Y. Shoham. " Taming the computational complexity of combinatorial auctions : optimal and approximate approaches". *In the Proceedings of the Sixteenth international joint conference on artificial intelligence*, pp : 548-553, 1999.

[Garey et Johnson 1979] M.R. Garey , D.S. Johnson. "Computers and Intractability, A Guide to the Theory of NP-Completeness". *In W.H. Freeman Company,* San Francisco, 1979.

[Givry *et al.*2003] S.D. Givry, J. Larrosa, P. Meseguer, et T. Schiex. "Solving Max-SAT as weighted CSP". *In Proceedings of the 9th International Conference on Principles and Practice of Constraint Programming (CP2003),* pp : 363-376, 2003.

[Glover 1994] F. Glover "Tabu search for nonlinear and parametric optimization (with links to genetic algorithms)". *In Discrete Appl. Math.* (49), pp : 231-255, 1994.

[Glover 1999] F. Glover. "Scatter search and path relinking". *In D. Corne, M.Dorigo, F. Glover, eds. New Ideas in Optimization. McGraw-Hill*, New York, pp :297-316, 1999.

[Glover *et al.*2003] F. Glover, M. Laguna et R. Marti. "Scatter Search : Advances in Evolutionary Computation : Theory and Applications", *In A. Ghosh and S. Tsutsui (Eds.), Springer-Verlag,* New York, pp : 519-537 (2003).

[Goldberg 1989] D. E Goldberg. "Genetic Algorithms in search, Optimization and Machine Learning". *Wokingham, Addison-Wesley.* 1989.

[Gootlieb *et al.*2002] J. Gottlieb, E. Marchiori, et C. Rossi. "Evolutionary algorithms for the satisfiability problem". *In Journal of Evolutionary Computation,* 10(1), pp :35-50, 2002.

[Guo *et al.*2004] Y. Guo, A. Lim, B. Rodrigues, Y. Zhu. ă"Heuristics for a brokering set packing problem". *In Proceedings of eighth international symposium on artificial intelligence and mathematics*, pp : 10-14, 2004.

[Guo *et al.*2006] Y. Guo, A. Lim, B. Rodrigues, Y. Zhu. "Heuristics for a bidding problem". *In Journal of Computers and Operations Research* Vol 33, Issue 8 , August 2006, pp : 2179-2188, 2006.

[Hao *et al.*2003] J.K. Hao, F.Lardeux et F.Saubion, "Evolutionary Computing for the Satisfiability Problem". *In Lecture Notes in Computer Sciences (EvoCOP03), pp :258-267,* UK, Springer, 2003.

[Hansen et Jaumard 1990] P. Hansen , B. Jaumard. "Algorithms for the Maximum Satisfiability" *In Journal of Computing* (44), pp : 279-303, 1990.

[Holland 1975] J.H. Holland, "Adaptation in natural and artificial systems", *University of Michigan Press*, Ann Arbor (1975).

[Holland et ăO'sullivan 2004] A. Holland,ăB. O'sullivan. "Towards Fast Vickrey Pricing using Constraint Programming", *In Artificial Intelligence Review*, Vol 21, $n3-4$ / June, pp : 335-352, 2004.

[Hoos 2002] H.H. Hoos. "An Adaptive Noise Mechanism for WalkSAT". *In Proceedings of the National Conference on Artificial Intelligence AAAI/IAAI 2002* : pp : 655-660,2002.

[Hoos et Boutilier 2000] H.H. Hoos, C. Boutilier, "Solving combinatorial auctions using stochastic local search". *In Proceedings of the 17th national conference on artificial intelligence*, pp : 22-29, 2000.

[Hoos et Stutzle 2005] H.H. Hoos , T. Stutzle, "Stochastic Local Search". *Cambridge, Massachusetts : Morgan Kaufman,* 2005.

[Laguna et Glover 1999] M. Laguna , F. Glover, "Scatter Search" *Graduate school of business, University of Colorado, Boulder*, 1999.

[Laguna *et al.*1999] M. Laguna, R. Marti et V. Campos, "Scatter Search for the linear ordering problem", *University of Colorado, Boulder*,1999.

[Lardeux *et al.*2006] F.Lardeux, F. Saubion et J.K. Hao. "GASAT : A genetic local search algorithm for the satisfiability problem". *In Journal of Evolutionary Computation,* Summer 2006, vol 14, $n2$, pp : 223-253, MIT Press, 2006.

[Lau et Goh 2002] H.C. Lau HC , Y.G. Goh. "An intelligent brokering system to support multi-agent web-based 4th-party logistics". *In Proceedings of the 14th international conference on tools with artificial intelligence*, pp : 154-161, 2002.

[Leyton-Brown *et al.*2000 a] K. Leyton-Brown, M. Pearson, Y. Shoham. "Towards a universal test suite for combinatorial auction algorithms". *In Proceeedings of the ACM conference on electronic commerce*, pp : 66-76, 2000.

[Leyton-Brown *et al.*2000 b] K. Leyton-Brown, M. Tennenholtz, Y. Shoham.ă "An Algorithm for Multi-Unit Combinatorial Auctions". *In Proceedings of the 17th national conference on artificial intelligence* , Austin, Games-2000, Bilbao, and ISMP-2000, Atlanta.

[Li *et al.*2006] C.M. Li, F. Manya et J. Planes. Detecting Disjoint Inconsistent Subformulas for Computing Lower Bounds for Max-SAT. *In Proceedings of Twenty-First National Conference on Artificial Intelligence (AAAI 2006).* July 2006.

[Li et Huang 2005] C.M. Li, W.Q. Huang. "Diversification and Determinism in Local Search for Satisfiability". *In Proceedings of SAT 2005* pp : 158-172, 2005.

[Li et Anbulagan 1997] C.M Li , A. Anbulagan. "Heuristics based on unit propagation for satisfiability problems". *In Proceeding. of the IJCAI'97*, pp : 366-371, 1997.

[Li 2000] C.M. Li, "Integrating equivalency reasoning into davis-putnam procedure". *In Proceedings. of the AAAI'00*, pp : 291-296, 2000.

[Marchiori et Rossi 1999] E. Marchiori , C. Rossi. "A flipping genetic algorithm for hard 3-SAT problems", *In Proceedings. of the Genetic and Evolutionary Computation Conference*, vol 1, pp : 393-400, 1999.

[Mazure *et al.*1997] B. Mazure, L. Sais et E. Greroire,. "A Tabu search for SAT" *in proceedings the In Proceedings of the National Conference on Artificial Intelligence AAAI-97/IAAI-97*, pp : 281-285, Providence, Rhode Island, 1997.

[McAllester *et al.*1997] D. McAllester, B. Selman, et H. Kautz. "Evidence for invariants in local search". *In Proceedings of In Proceedings of the National Conference on Artificial Intelligence AAAI-97,* pp : 321-326, 1997.

[Mills et Tsang 2000] P. Mills , E.P.K Tsang "Guided local search for solving SAT and weighted MAX-SAT problems", *In Journal of Automated Reasoning, Special Issue on Satisfiability Problems, Kluwer,* Vol.24, pp : 205-223, 2000.

[Moskewicz *et al.*2001] M.W. Moskewicz, C.F. Madigan, Y.Zhao, L.Zhang, et S.Malik, 2001. "Chaff : Engineering an efficient SAT solver". *In Proceedings of the 38th Design Automation Conference (DAC'01).*

[Moscato 1989] P. Moscato. , "On Evolution, Search, Optimization, Genetic Algorithms and Martial Arts : Towards Memetic Algorithms", *In Caltech Concurrent Computation Program*, C3P Report 826, 1989.

[McAfee et TMcMillan 1987] R. McAfee, P.J. McMillan. "Auctions and bidding", *In Journal of Economic Literature*, Vol. 25, pp : 699 ?738, 1987.

[Nisan 2000] N. Nisan. " Bidding and allocation in combinatorial auctions", *In Proceedings of ACM Conference on Electronic Commerce (EC'00)*, Minneapolis : ACM SIGecom, ACM Press, October, pp.1-12, 2000.

[Papadimitrion 1994] C.H. Papadimitrion. "Computational complexity", *Addision-Wesley*, New-york, 1994.

[Pardalos *et al.*1994] P.M. Pardalos, F. Rendl et H. Wolkowicz. "The quadratic assignment problem, a survey and recent developments", *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, Vol. 16, pp : 1-42, 1994.

[Rana et Whitley 1998] S. Rana , D. Whitley. "Genetic algorithm behavior in the maxsat domain". *In A. E.Eiben, T. Back, M. Schoenauer, and H.-P. Schwefel, editors, Proceedings of PPSN-V, Fifth International Conference on Parallel Problem Solving from Nature, volume 1498 of Lecture Notes in Computer Science,* pp : 785-794. Springer Verlag, Berlin, Germany,1998.

[Rothkopf *et al.*1998] M.H. Rothkopf, A. Pekee et M. Ronald. "Computationally manageable combinatorial auctions", *In Management Science*, Vol. 44, $n8$, pp : 1131-1147, 1998.

[Sandholm 1999] T. Sandholm, (1999). "Algorithms for Optimal Winner Determination in Combinatorial Auctions". *In Artificial Intelligence*, Vol 135, $n(1-2)$, pp : 1-54, 1999.

[Sandholm *et al.*2001] T. Sandholm T, S. Suri, A. Gilpin et D. Levine . "CABoB : a fast optimal algorithm for combinatorial auctions". *In Proceedings of the International joint conferences on artificial intelligence*, pp : 1102-1108, 2001.

[Sandholm et Suri 2000] T. Sandholm, S. Suri. "Improved Optimal Algorithm for Combinatorial Auctions and Generalizations". *In Proceedings of the 17th national conference on artificial intelligence AAAI2000*, pp : 90-97, 2000.

[Sandholm 2006] T. Sandholm. "Optimal Winner Determination Algorithms" . *In P. Cramton et al. (ed.), Combinatorial Auctions*, MIT Press, 2006.

[Selman *et al.*1992] B. Selman, H. Levesque, et D. Mitchell. "A new method for solving hard satisfiability problems". *In Proceedings of the National Conference on Artificial Intelligence,* pp : 440-446. AAAI Press/The MIT Press, Menlo Park, CA, USA, 1992.

[Selman *et al.*1994] B. Selman, H. Kautz, et B. Cohen. "Noise strategies for local search". *In Proceedings the national conference on artificial intelligence AAAI-94,* pp : 337-343, 1994.

[Shen et Zhang2002] H. Shen, H. Zhang. "Study of lower bound functions for max-2-sat". *In Proceedings of AAAI-02,* pp : 185-190, 2002.

[SIMA2007] 'Special Issue on Memetic Algorithms', (2007), *In IEEE Transactions on Systems, Man and Cybernetics - Part B*, **Vol. 37**, $n1$, Ong Y.S. ; Krasnogor, N. ; Ishibuchi H. (Eds.), Feb 2007.

[Siarry1995] P. Siarry . "La méthode du recuit simulé : théorie et applications". *In Automatique productique, informatique industrielle* Vol 29 $n4$/1995.

[Stutzle *et al.*2003] T. Stutzle, H.H. Hoos et A. Roli. "A review of the literature on local search algorithms for MAX-SAT", *Internet document*, 2003

[Talbi *et al.*1994] E. G. Talbi, T. Muntean , I. Samarandache . "Hybridation des algorithmes genetiques avec la recherche tabou" , *In Proceddings of the Evolution Artificielle EA'94*, Toulouse, France, Sept 1994.

[Vries et Vohra 2003] S. de Vries, R. Vohra, "Combinatorial auctions a survey", *In INFORMS Journal of Computing*, Vol 15 , pp : 284-309, 2003.

[Wurman *et al.*1998] P.R. Wurman, M.P. Wellman et W.E. Walsh. "The michigan Internet auctionbot : A configurable auction server for human and software agents", *In Proceddings of the 2nd International Conference on Autonomous Agent*, Minneapolis, 1998.

[Xing et Zhang 2005] Z. Xing, W. Zhang, "MaxSolver : An Efficient Exact Algorithm for (Weighted) Maximum Satisfiability", *In Artificial Intelligence* ,(vol 2), pp : 47-80, 2005.

[Zhang 1997] H.Zhang, "SATO : An efficient propositional prover". *In Proceedings of the 14th International Conference on Automated deduction*, vol 1249 of LNAI, pp : 272-275, 1997.

[Zhang *et al.*2003] W. Zhang, Rangan et M. Looks, "Backbone guided local search for maximum satisfiability", *in Proceedings of the 18th International Joint Conference on AI (IJCAI-03)*, Acapulco, Mexico, Aug. 9-15, 2003, pp : 1179-1184.

Glossaire

- GA : Genetic Algorithms
- SS : Scatter Search
- MA : Memetic Algorithms
- LS : Local search.
- TS : Tabu search.
- SA : Simulated Annealing.
- SSV : Scatter search variant.
- SLS : Stochastic local search.
- GAV : Genetic Algorithms variant.
- $SGAV$: Specific Genetic Algorithm Variant.
- SAT : Satisfiability Problem.
- $MAX - SAT$: Maximum Satisfiability Problem.
- $MAX - W - SAT$:Weighted Maximum Satisfiability Problem
- DP : Davis et Putnam Procedure.
- WDP : Winner determination problem.
- CA : Combinatorial auctions.
- BoI : Branch-on-Items.
- BoB :Branch-on-Bids.
- $CABoB$: Cambinatorial auctions Branch-on-Bids.
- $CASS$: Combinatorial Auction Structural Search.
- $CAMUS$: Combinatorial Auctions Multi-Unit Search.
- $Casanova$: Local search for the WDP.
- FCC : Federal Communication Commission.
- $AUML$ Agent Unified Modeling Language.
- $NetCombinatoire$ Plate forme based agents for combinatorial auctions.

Table des figures

Résumé

Notre thèse comporte deux grandes parties, dont le point commun est la résolution de problèmes NP-complets.

Nous nous intéressons, dans la première partie de la thèse au problème SAT. Le problème SAT de la logique booléenne est le premier problème démontré NP-Complet. C'est la satisfaisabilité d'une formule propositionnelle donnée sous forme clausale : conjonction de clauses, où chaque clause est une disjonction de littéraux, et un littéral est une variable ou sa négation. Le problème est de décider alors s'il y a une affectation de valeurs de vérité aux variables propositionnelles qui rend la formule vraie. Dans le cas où cette affectation n'existe pas, l'ensemble de clauses est dit non satisfaisable et une variante du problème appelée MAX-SAT est introduite. Cette variante consiste à chercher une affectation de valeurs de vérité qui maximise le nombre de clauses satisfaites.

Pour contribuer à la résolution du problème MAX-SAT par les approches évolutionnaires, nous proposons, tout d'abord, une nouvelle stratégie de sélection qui se base sur la diversité et la qualité pour choisir une collection d'individus qui vont participer à la phase de reproduction et donner une descendance. Ensuite, nous utilisons un opérateur de combinaison spécifique au problème MAX-SAT pour générer de nouveaux enfants qui sont améliorés par une recherche locale stochastique (SLS). Les trois composantes proposées sont incorporées, premièrement, dans un algorithme génétique pour améliorer sa performance. Ensuite, une variante de recherche dispersée, utilisant les trois composantes déjà citées, est étudiée pour le problème MAX-SAT. Dans le but de tester et de prouver l'efficacité de nos approches, une étude comparative avec quelques algorithmes de l'état de l'art concernant MAX-SAT à savoir Walksat , FlipGA et GASAT est établie dans la première partie de la thèse.

Nous nous intéressons, dans la deuxième partie de cette thèse, au problème de la détermination du gagnant *PDG* dans les enchères combinatoires : un ensemble d'objets est soumis à la vente face à plusieurs acheteurs, chaque acheteur, pour des raisons de complémentarité entre les objets, désire acheter un sous-ensemble d'objets qui lui est propre, et dont il fournit une estimation. Les conflits entre les acheteurs naissent des éventuelles intersections entre les sous-ensembles. Le vendeur doit alors résoudre un problème d'optimisation combinatoire NP-Complet pour réaliser la vente qui lui rapportera le plus.

Autour de ce thème, nous proposons, d'une part, une architecture à base d'agents pour simuler le mécanisme d'enchères combinatoires. D'autre part, nous proposons une métaheuristique de recherche taboue pour évaluer les offres et déterminer le gagnant de l'enchère.

Abstract

The satisfiability problem (*SAT*) is a core problem in both artificial intelligence discipline and computational complexity theory. Formally, given a set of clauses where a clause is a disjunction of literals and a literal is a variable or its negation. The SAT problem is to decide whether an assignment of values to variables exists such that all the clauses are simultaneously satisfied. The maximum satisfiability problem (*MAX-SAT*) is the optimization variant of SAT. Given a propositional formula in conjunctive normal form (CNF), the *MAX-SAT* problem consists in finding a variable assignment that maximizes the number of satisfied clauses.

In the first part of this thesis, we investigate some evolutionary algorithms for *MAX-SAT* problems. Our approaches are characterized by their selection strategy that uses both fitness and diversity to choose individuals for participating in the reproduction phase, by their powerful SLS used as a local improvement technique and their population management strategy.

Our aim is to associate to each generation a collection of both highest quality and diverse individuals. The resulting algorithm is enhanced by using an advanced crossover operator (to *MAX-SAT*) combined with an improved SLS as an improvement strategy.

In the second part of this thesis, we are interested in the optimal winner determination problem (*WDP*) in combinatorial auctions. Given a set of bundles bids, the winner determination problem is to decide which of the bids to accept. More precisely, the *WDP* is finding an allocation that maximizes the auctioneer's revenue, subject to the constraint that each item can be allocated at most once.

Our approaches are based on métaheuristics and Agents. On one hand, we propose an agent-based architecture to implement the combinatorial auctions mechanisms. On the other hand, we develop a new no-exact tabu search algorithm to evaluate bids and select the winning ones for the *WDP*.