7e Conférence Francophone de MOdélisation et SIMulation - MOSIM'08 - du 31 mars au 2 avril 2008 - Paris - France
«Modélisation, Optimisation et Simulation des Systèmes : Communication, Coopération et Coordination»

APPROCHES ÉVOLUTIONNAIRES POUR LE PROBLEME MAX-SAT

D.BOUGHACI[1,2] **, B.BENHAMOU**[1] **H. DRIAS**[2]

[1]INCA /LSIS,
CMI 39 rue Fredric Joliot-Curie
13453 Marseille cedex 13.
{boughaci, Belaid.benhamou}@cmi.univ-mrs.fr

[2]LRIA/USTHB
BP 32 El-Alia, Beb-Ezzoaur,
- 16111, Alger, Algérie.
{dboughaci, h_drias}@usthb.dz

RÉSUMÉ : *Dans ce papier, 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.*

MOTS-CLÉS : *Algorithme génétique, recherche locale, recherche dispersée, opérateur de croisement, approches évolutionnaires, MAX-SAT.*

1. INTRODUCTION

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 :

- La première classe englobe les algorithmes complets ou exacts dont le principe essentiel consiste généralement à énumérer, souvent de manière implicite, l'ensemble des solutions de l'espace de recherche. La plupart des algorithmes exacts sont basés principalement sur la procédure de Davis-Putnam-Loveland (Davis *et al.*, 1962). Parmi les algorithmes complets pour le problème MAX-SAT, nous citons (Alsinet *et al.*, 2003; Borchers et Furman, 1999; Li *et al.*, 2006; Xing et Zhang, 2005). Les méthodes exactes 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 approchées 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 Huang, 2005)

sont des exemples d'algorithmes de recherche locale. Parmi les approches évolutionnaires proposées pour le problème SAT, nous citons : les algorithmes génétiques (Frank, 1994; Hao *et al.*, 2003; Rana et Whitley, 1998), la recherche dispersée (Boughaci et Drias, 2005), et les algorithmes mémétiques (Boughaci *et al.*, 2004; Gottlieb *et al.*, 2002; Lardeux *et al.*, 2006; Marchiori et Rossi, 1999). 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.

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).

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 le 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 (Holland, 1975) pour améliorer sa performance. Ensuite, une variante de recherche dispersée, utilisant les trois composantes déjà citées, est étudiée pour le problème MAX-SAT.

Nos approches présentent des similitudes avec les algorithmes évolutionnaires déjà proposés (Lardeux *et al.*, 2006; Marchiori et Rossi, 1999) et qui remplacent la phase de mutation par une recherche locale. Nos méthodes se distinguent par leur stratégie de sélection qui considère le 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 le fitness et la diversité. Notre objectif est d'associer à chaque génération une collection composée de deux types d'individus: ceux ayant un fort 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.

Le reste de ce papier est organisé comme suit: La prochaine section (2) présente notre recherche locale utilisée par nos approches pour améliorer la qualité des solutions. Nous décrivons à la troisième section notre algorithme génétique spécifique au problème MAX-SAT. Nous présentons dans la quatrième section notre variante de recherche dispersée pour le problème MAX-SAT. Quelques résultats expérimentaux sont donnés dans la section 5. Enfin, nous terminons, ce papier, par une conclusion et quelques perspectives de ce travail.

2. RECHERCHE LOCALE STOCHASTIQUE(SLS)

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 un certain nombre d'itérations fixé d'une manière empirique.

PROCEDURE 1. La méthode d'amélioration de SLS
Entrée: la formule MAX-SAT , l'individu V
maxflips, dp, wp
Sortie: un individu amélioré V
Début
r = un nombre aléatoire entre 0 et 1
Pour flip=1 **à** maxflips
faire
Si ($r \prec wp$) **alors**
$flipvar$ = choisir aléatoirement une variable
sinon Si ($r \prec dp$) **alors**
cl= choisir aléatoirement une clause non satisfaite
$flipvar$ = choisir aléatoirement une variable de cl
sinon $flipvar$ = choisir la meilleure variable;
Finsi
Finsi
$V = V$ avec $flipvar$ renversée.
Finpour
retourner le meilleur individu trouvé.
Fin

La technique de recherche locale que nous proposons est basée sur les concepts bien connus de Walksat (Selman *et al.*, 1994) à chaque itération, la variable à inverser (changement d'une solution) 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 la Procédure 1.

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 dans l'espace de recherche, 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 le 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 :

3.1. 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.

3.2. Fonction d'évaluation

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

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).

3.4. 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 suivantes :

- Sélectionner les B_1 meilleures solutions de P pour pouvoir déduire les solutions de haute qualité choisies selon leur fonction d'évaluation. L'ensemble B reçoit initialement les B_1 meilleures solutions de P.
- Pour chaque solution V appartenant à $P-B$, calculer la diversité de V par rapport aux solutions dèjà dans B. Pour calculer la diversité d'une solution V par rapport à un ensemble de solutions B, on doit calculer $|B|$ distances de Hamming tel que $|B|$ est la cardinalité de l'ensemble B. C'est pourquoi la nouvelle stratégie de sélection est efficace sur un ensemble B de petite taille d'environ 10% de la population totale.
- Sélectionner un certain nombre B_2 de solutions de $P - B$, celles ayant une grande diversité. La solution qu'on doit rajouter à l'ensemble B doit être la plus éloignée de solutions déjà dans l'ensemble B.
- 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. La taille de B est fixée empiriquement.

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).

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 la Procédure 2.

PROCEDURE 2. L'opérateur de combinaison
Entrée: deux parents, $Parent1$ et $Parent2$
Sortie: un enfant, $Child$
Début
(*initialement toutes les variables dans $Child$ sont non valuées*)
Pour toute (clauses 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
Finpour
/* Le croisement uniforme*/
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
Finpour
Retourner l'enfant $Child$.
Fin

3.6. L'algorithme génétique spécifique au problème MAX-SAT

L'algorithme génétique spécifique au problème MAX-SAT est donné dans la Procédure 3:

PROCEDURE 3. Algorithme génétique spécifique au MAX-SAT
Entrée: une instance MAX-SAT.
Sortie: une affectation maximisant le nombre de clauses satisfaites.
Début
Générer aléatoirement une population initiale P;
Pour I=1 au nombre maximum d'essais (Maxtries)
Faire
Sélectionner une collection d'individus B de P
Tant que (le nombre maximum de générations n'est pas atteint)
Faire
Répéter
Sélectionner deux parents d'individus de B;
Générer aléatoirement un nombre r entre 0 et 1;
Si ($r \prec$ le taux de croisement) **alors**
Appliquer l'opérateur de combinaison aux parents
pour obtenir un nouvel enfant V;
Appliquer SLS sur V;
Si (V améliore la qualité de B) **alors**
Ajouter V aux B_1 meilleurs individus;
Enlever de B le mauvais individu;
SinonSi (V améliore la diversité de B) **alors**
Ajouter V aux B_2 individus dispersés;
Enlever de B l'individu le moins dispersé;
Finsi
Finsi
Finsi
Jusqu'au (toutes les combinaisons des parents sont examinées)
Fin Tant que
Reproduire une nouvelle population P;
Fin pour
Retourner le meilleur individu trouvé ainsi que son fitness.
Fin

4. UNE VARIANTE DE RECHERCHE DISPERSÉE POUR LE PROBLÈME MAX-SAT

La recherche dispersée (Glover, 1999) est une métaheuristique à stratégies d'évolution. Elle intègre les idées majeures des méthodes évolutionnistes (population de solutions, recombinaison de solutions). La variante de recherche dispersée que nous proposons (SSV) démarre d'une population initiale variée générée en uilisant une méthode de diversification. 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 *Section* 3 et qui utilise la nouvelle stratégie de sélection basée sur le 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.

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 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: $Y_1 = 1\text{-}\ X_1$ (1)

$Y_{1+k\times h}=1 - X_{1+k\times h}, \quad k = 1,\dots k^*,\ k^* \leq n/h,$
$1 \leq h < n$ (2)

Solutions de type 2: Z est le complément à 1 de Y (3)

	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	

Figure 1: Exemple d'une population de taille 8.

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 (1), (2) et (3). On obtient la population donnée dans la Figure 1.

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 la procédure 4:

PROCEDURE 4. Recherche dispersée pour MAX-SAT
Entrée: une instance MAX-SAT.
Sortie: une affectation maximisant le nombre de clauses satisfaites.
Début
  // maxgen est le nombre maximum de générations
  // maxcombin est le nombre maximum de combinaisons
  gen =1;
  Générer une population initiale P;
  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
  Fintant que
  Régénérer une nouvelle population P;
  Fintant que
  Retourner la meilleure solution trouvée.
Fin

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 des 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.

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 FlipGA et GASAT (Lardeux *et al.*, 2006) est établie dans ce papier.

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.

[1]http://www.nmt.edu/ borchers/maxsat.html

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

[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.

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 la Table 1 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 est fixé à vars $\times$ 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 et 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 ces instances, 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 2 et 3 donnent quelques résultats trouvés où les paramètres sont fixés comme suit :

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

- 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=100, 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.

Pour la classe ais, nous avons utilisé une population de 50 solutions tandis que pour les autres classes la taille de population est fixée à 100 solutions.

5.2. Résultats

Les résultats obtenus sont donnés dans les tables 1, 2 et 3 [6]. 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.

5.2.1. Comparaisons avec GA classique

- MAX-2-SAT and MAX-3-SAT Instances

[6] *vars* (respectivement *cls*) est le nombre de variables (respectivement de clauses) de l'instance MAX-SAT.

Table 1: Les résultats trouvés par SSV, GAV, SGAV et GA sur les instances *MAX-2-SAT et MAX-3-SAT.*

Instances.cnf (#Vars/#Cls)	U	**SSV**		**GAV**		**SGAV**		GA	
		sol	time	*sol*	time	*sol*	time	*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
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

La Table 1 montre 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.

- SATLIB Instances

A partir de la Table 2, 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.

Table 2: Les résultats trouvés par SSV, GAV, SGAV et GA sur les instances 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
dubois29(87/232)	1	**1**	0.61	**1**	**0.03**	**1**	0.09	21	0.56
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
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
aim50-2_no2(50/100)	1	**1**	0.32	**1**	**0.01**	**1**	0.04	6	0.42
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
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
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

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 la Table 3, 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: FlipGA et GASAT.

Table 3: Comparaison entre SSV, GAV et SGAV sur des instances de *SATLIB*.

Instances.cnf (#Vars/#Cls)	U	**SSV** *sol*	time	**GAV** *sol*	time	**SGAV** *sol*	time
glassy-a(399/1862)	**0**	6	18.57	8	3.15	**5**	78.31
glassy-b(399/1862)	**0**	9	18.76	12	3.21	**8**	78.44
glassy-c(399/1862)	**0**	7	18.68	8	3.19	**6**	78.48
glassy-d(399/1862)	**0**	6	18.66	10	3.26	**6**	78.35
glassy-e(399/1862)	**0**	6	18.66	9	3.19	**6**	78.36
glassy-f(450/2100)	**0**	9	23.25	12	4.08	**8**	98.50
glassy-g(450/2100)	**0**	11	23.24	13	4.02	**9**	98.66
glassy-h(450/2100)	**0**	9	23.15	12	4.10	**7**	98.17
glassy-i(450/2100)	**0**	7	23.20	11	4.09	**7**	98.28
glassy-j(450/2100)	**0**	11	23.20	12	4.05	**9**	98.44
color-10-3(300/6475)	**0**	**0**	21.28	4	8.24	**0**	17.95
color-18-4(1296/95904)	**0**	17	1232.44	45	544.61	**15**	1510.09
par32-1-c(1315/5254)	**0**	**11**	138.03	49	28.51	**11**	658.82
par32-1(3176/10277)	**0**	47	659.41	65	122.55	**35**	1186.74
par32-2-c(1303/5206)	**0**	11	135.26	51	27.65	**10**	655.57
par32-2(3176/10253)	**0**	42	657.68	64	122.09	**37**	1195.25
par32-3-c(1325/5294)	**0**	**10**	139.52	50	28.55	11	679.15
par32-3(3176/10297)	**0**	49	660.23	68	122.64	**37**	1181.29
par32-4-c(1333/5326)	**0**	11	141.16	41	28.68	**9**	636.54
par32-4(3176/10313)	**0**	48	3634.44	58	122.88	**34**	1178.16
par32-5-c(1339/5350)	**0**	11	142.72	53	29.15	**11**	671.41
par32-5(3176/10325)	**0**	51	661.21	61	122.97	**37**	1174.43
ais10(181/3151)	**0**	**0**	1.93	1	2.25	**0**	**1.25**
ais12(265/5666)	**0**	1	29.58	1	5.58	**0**	**51,84**
ais6(61/581)	**0**	**0**	0.04	**0**	0.09	**0**	0.09
ais8(113/1520)	**0**	**0**	0.80	1	0.71	**0**	**1.90**
f1000(1000/4250)	**0**	7	90.06	10	16.15	**1**	408.78
f2000(2000/8500)	**0**	11	337.76	9	60.10	**6**	2670.74
f600(600/2550)	**0**	2	34.07	3	6.17	**0**	102.69
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)	**0**	14	155.06	38	28.73	**11**	3532.01
difp_19_1(1201/6563)	**0**	14	155.52	44	29.22	**10**	764.02
difp_19_3(1201/6563)	**0**	15	156.87	38	28.94	**12**	766.35
difp_19_99(1201/6563)	**0**	16	156.71	44	29.14	**10**	766.11
logistics.a(828/6718)	**0**	2	98.92	6	17.88	**1**	484.04
logistics.b(843/7301)	**0**	2	107.73	4	19.75	**1**	508.05
logistics.c(1141/10719)	**0**	3	201.57	6	36.96	**2**	3350.43
logistics.d(4713/21991)	**0**	11	2397.31	13	355.36	**7**	639.63
hgen2-a(500/1750)	**0**	5	22.77	6	3.95	**2**	97.45
hgen2-b(500/1750)	**0**	5	22.72	5	3.89	**3**	97.47
hgen2-c(500/1750)	**0**	3	22.88	8	3.86	**2**	97.53
hgen2-d(500/1750)	**0**	4	22.83	5	3.96	**2**	97.57
hgen2-e(500/1750)	**0**	4	22.76	8	3.96	**3**	97.67

5.2.3. 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 mémétique proposé par (Marchiori et Rossi, 1999) et utilisé dans (Lardeux *et al.*, 2006) pour des études comparatives.

La Table 4 compare SSV, SGAV, GASAT et FlipGA où fc est le nombre moyen de clauses falsifiées par la solution trouvée par chaque algorithme.

- 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 4 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).

Table 4: 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	**5.0**	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	**5.0**	7.83
glassy-b (450/2100)	8.95	86	11.45	21	9	23.15	**7.0**	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

6.CONCLUSION

Dans ce papier, nous avons proposé quelques approches évolutionnaires pour le problème MAX-SAT. La méthode de base utilise une nouvelle stratégie de sélection basée sur la diversité et le 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, 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.

References

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

Boughaci. D, Drias. H, 2005. Efficient and Experimental Meta-heuristics for MAX-SAT Problems. *LNCS, Volume 3503 / 2005,* WEA 2005: p. 501-512, 2005.

Boughaci. D, Drias.H and Benhamou.B, 2004. Solving Max-SAT Problems Using a mimetic evolutionary Meta-heuristic, *in Proceedings of the IEEE-CIS 2004,* p. 480-484.

Borchers.B and Furman.J, 1999. A two-phase exact algorithm for Max-SAT and weighted Max-SAT problems. *J. Combinatorial Optimization,* 2 (4) p. 299-306.

Cook. S.A, 1971. The complexity of theorem proving procedures. *In Proceedings of the 3rd ACM symposium on Theory of Computing,* p. 151-158, Ohio.

Davis. M, Logemann.G, and Loveland.D, 1962 A machine program for theorem proving. *In Communication of CACM*, 5 p. 394-397.

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

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

Glover. F, 1994. Tabu search for nonlinear and parametric optimization (with links to genetic algorithms). *In Discrete Applied Mathematics.* (49) p. 231-255.

Gottlieb. J, Marchiori.E, and Rossi.C, 2002. Evolutionary algorithms for the satisfiability problem. *In Evolutionary Computation,* 10(1) p. 35-50.

Hao. J.K, Lardeux.F and Saubion.F, 2003. Evolutionary Computing for the Satisfiability Problem". *Lecture Notes in Computer Science (EvoCOP03),* p. 258-267, UK, Springer.

Hansen. P and Jaumard.B, 1990. Algorithms for the Maximum Satisfiability *journal of computing* (44) p. 279-303.

Holland. J.H, 1975. Adaptation in natural and artificial systems, *University of Michigan Press*, Ann Arbor.

Hoos.H.H, 2002. An Adaptive Noise Mechanism for WalkSAT. *AAAI/IAAI 2002* p. 655-660.

Hoos. H.H and Stutzle.T, 2005. Stochastic Local Search. *Cambridge, Massachusetts: Morgan Kaufman,.*

Lardeux. F, Saubion.F and Hao.J.K, 2006. GASAT : A genetic local search algorithm for the satisfiability problem. *Journal of Evolutionary Computation,* Summer 2006, Vol 14, N2, p. 223-253, MIT Press.

Marchiori.E and Rossi.C, 1999. A flipping genetic algorithm for hard 3-SAT problems, *In Proceedings. of the Genetic and Evolutionary Computation Conference*, Vol 1, p. 393-400.

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

Li. C.M, Huang.W.Q, 2005. Diversification and Determinism in Local Search for Satisfiability. *In Proceedings of SAT 2005* p. 158-172.

Mazure. B, Sais.L and Greroire.E, 1997. A Tabu search for SAT *in proceedings the AAAI-97/IAAI-97*, p. 281-285, Providence, Rhode Island.

McAllester.D, Selman.B, and Kautz.H, 1997. Evidence for invariants in local search. *In Proceedings of AAAI-97,* p. 321-326.

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

Rana.S and Whitley.D, 1998. 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,* p. 785-794. Springer Verlag, Berlin, Germany.

Selman.B , Levesque.H and Mitchell.D, 1992. A new method for solving hard satisfiability problems. *In Proceedings of the 10th National Conference on Artificial Intelligence,* p. 440-446. AAAI Press/The MIT Press, Menlo Park, CA, USA.

Selman.B, Kautz.H and Cohen.B, 1994. Noise strategies for local search. *In Proceedings of AAAI-94,* p. 337-343.

Xing.Z, Zhang.W, 2005. MaxSolver: An Efficient Exact Algorithm for (Weighted) Maximum Satisfiability, *Artificial Intelligence* ,(Vol 2), p. 47-80.