***Sciences &*** Technologie **B** – **N°22**, Décembre (2004), pp. 47-53.

# ADAPTATION DE LA DYNAMIQUE DES SYSTEMES AUTO-ORGANISES A LA RESOLUTION DU PROBLEME DE SATISFAISABILITE MAXIMALE DE FORMULES PROPOSITIONNELLES

*Reçu le 02/09/2002 –Accepté le 27/11/2004*

**Résumé**

Le problème de satisfaisabilité maximale de formules propositionnelles (Max-SAT) est un problème central en intelligence artificielle, logique mathématique et optimisation combinatoire. Les algorithmes de résolution utilisent différentes heuristiques pour parcourir l'espace de recherche dont les performances sont liées à leur complexité et au réglage de leurs paramètres. L'Optimisation Extrémale (EO) est une des méthodes les plus simples à mettre en oeuvre et n'utilise qu'un seul paramètre de contrôle. Elle est inspirée de la dynamique des systèmes physiques de complexité émergente et de leur capacité à s'auto-organiser pour atteindre un état d'adaptation optimal. Ce travail est motivé par les progrès récemment réalisés par l'application de cette méthode à des problèmes classiques d'optimisation tels que le partitionnement de graphe. Dans cet article, nous décrivons un nouvel algorithme de recherche locale stochastique pour la résolution du problème Max-SAT s'articulant autour de la méthode EO et présentons une analyse empirique de ses performances. Les résultats numériques obtenus améliorent de façon significative ceux produits par les algorithmes de recuit simulé et de recherche taboue.

***Mots clés:*** *Max-SAT, Heuristique, Optimisation extrémale, Optimisation combinatoire, Recherche locale stochastique.*

**Abstract**

The maximum satisfiability problem of propositional formula (Max-SAT) is a central problem in artificial intelligence, mathematical logic and combinatorial optimization. Algorithms to solve them use a variety of heuristics to explore the search space. Their performances depend both on their complexity and their tuning parameters. Extremal Optimization (EO) is a single free parameter method inspired by the dynamics of physical systems with emergent complexity and their ability to self-organize to reach an optimal adaptation state. This work is motivated by the recent progress achieved with this method on a number of classical optimization problems such as the graph partitioning problem. In this paper, we describe a new stochastic local search algorithm based on EO to solve Max-SAT problems and present an empirical evaluation of its performance. Numerical results show that this algorithm improves significantly results obtained with simulated annealing and tabu search algorithms.

***Keywords:*** *Combinatorial optimization, Extremal optimization, Heuristic, Max-SAT, Stochastic local search.*

**M. EL BACHIR MENAÏ**
Département d'Informatique
Centre universitaire Chikh Larbi Tébessi
12000 Tébessa (Algérie)

**M. BATOUCHE**
Laboratoire LIRE
Département d'Informatique
Université Mentouri
25000 Constantine (Algérie)

**ملخص**

إن مسألة التواؤم الأعظم لصيغ القضايا (Max-SAT) تعتبر مشكلا مركزيا في ميادين الذكاء الاصطناعي، المنطق الرياضي والإستمثال التوافقي. إن العديد من الاستكشافات المستعملة في خوارزميات الحل موجودة، لكن فعالياتها تبقى مربوطة بتعقيداتها، وبالضبط لوسائطها المراقبة تجريبيا. إن طريقة الإستمثال الأقصى (EO) من أسهل الطرق عند التطبيق ولا تقبل إلا وسيطا واحدا للمراقبة، وهي مشتقة من ديناميكية الأنظمة الفيزيائية ذات التعقيد البارز، ومن قدراتها للتنظيم الذاتي لكي تحقق حالة تطابق أمثل. يعتبر هذا العمل محفزا بالنتائج الحديثة المشجعة المتوصل لها بهذه الطريقة من أجل مسائل تجزيئية البيانات. نصف في هذه المقالة طريقة جديدة لحل مسألة Max-SAT ونعرض نتائج تجريبية متحصل عليها من أجل مجموعة حالات عشوائية. تبرهن التجارب أن هذه الطريقة تحسن بصفة ملموسة نتائج سابقة متحصل عليها بطرق "تظاهر حمى المعدن ثانية" و "البحث المحرم".

***الكلمات المفتاحية:*** *إستمثال أقصى، إستمثال توافقي، بحث محلي، الاستكشاف، Max-SAT.*

Le problème de satisfaisabilité d'une formule booléenne mise sous forme normale conjonctive (SAT) est un problème *NP*-complet [9] (*NP* désigne la classe de tous les problèmes de décision pouvant être résolus en un temps polynomial par un algorithme indéterministe): étant donné une formule booléenne, la question est de savoir si elle admet un modèle. Une extension de SAT est le problème Max-SAT qui consiste à satisfaire le plus grand nombre possible de clauses de la formule propositionnelle. Max-SAT est d'intérêt considérable aussi bien en théorie qu'en pratique : il joue un rôle important dans la spécification de nombreuses classes d'approximation telles que PTAA (Polynomial Time Approximation Algorithm) et PTAS (Polynomial Time Approximation Scheme) [26]. En outre, plusieurs problèmes de Recherche Opérationnelle (RO) ou d'Intelligence Artificielle (IA) peuvent s'exprimer sous forme de problèmes Max-SAT (l'ordonnancement d'actions, la gestion de ressources, le diagnostic des pannes, la recherche d'une clique maximum dans un graphe, …). Max-SAT est un problème *NP*-difficile (problème d'optimisation dont le problème de décision correspondant est *NP*-complet), la recherche de solutions exactes requière alors des temps exponentiels. De nombreuses méthodes de résolution ont été développées et sont classées sommairement en deux grandes catégories: Les méthodes *exactes* (*complètes* pour les problèmes de décision) qui garantissent la complétude de la résolution, et les méthodes *approchées* (resp. *incomplètes*) qui gagnent en efficacité au détriment de la complétude.

Les méthodes exactes parcourent généralement l'espace de recherche de l'instance du problème de manière systématique en disposant de techniques pour détecter le plus tôt possible les échecs et d'heuristiques pour orienter les différents choix. Elles utilisent également des méthodes de filtrage et de consistance. Les méthodes exactes demandent généralement un temps de calcul exponentiel en la taille du problème à résoudre et sont appliquées à des instances de petite taille. Les méthodes approchées représentent une alternative très intéressante lorsque les instances à traiter sont de très grande taille. Ces méthodes utilisent essentiellement le principe de la réparation locale qui consiste en une amélioration itérative d'une solution initiale générée aléatoirement. Une même configuration de l'espace de recherche peut être visitée plus d'une fois et l'existence de minima locaux impose l'utilisation d'heuristiques d'exploration efficace évitant le blocage aux voisinages de ces minima. Depuis une dizaine d'années, des progrès importants ont pu être réalisés avec l'apparition de nouvelles générations de méthodes approchées puissantes et générales, appelées méthaheuristiques, adaptables et applicables à une large classe de problèmes (le recuit simulé, la recherche taboue, les algorithmes génétiques, les stratégies d'évolution, …). Des résultats encourageants ont été obtenus récemment par Boettcher et Percus avec leur nouvelle méthaheuristique, appelée « *Optimisation Extrémale* » (EO) [6] sur des instances des problèmes de partitionnement de graphe et du voyageur de commerce en comparaison avec les algorithmes génétiques et le recuit simulé.

Dans ce travail, nous nous intéressons à l'examen de cette méthode pour les problèmes Max-SAT. Après une présentation du problème Max-SAT et de quelques algorithmes utilisés pour sa résolution comprenant GSAT, le recuit simulé et la recherche taboue (section 1), nous détaillons la méthode EO (section 2). Nous montrons ensuite comment l'adapter pour la résolution du problème Max-SAT (section 3) et présentons les jeux de tests, les résultats expérimentaux et une analyse de ces résultats (section 4). Nous terminons l'article par quelques conclusions en présentant quelques voies de recherche.

## 1- PROBLEME Max-SAT

Soit un ensemble de $n$ variables booléennes $X = \{x_1,...,x_n\}$. Un littéral est une variable $x$ (appelé littéral positif) ou sa négation $\neg x$ (appelé littéral négatif). Une clause $C$ est une disjonction finie de littéraux. Une formule propositionnelle (ou instance) $A$ est en Forme Normale Conjonctive (CNF) si c'est une conjonction de clauses. Une interprétation est une application $I$ de l'ensemble $X$ des variables vers $\{V,F\}$, qui a toute variable $x$ lui associe une valeur de vérité. $I$ est un modèle de $A$ si $I(A) = V$. Un littéral $l$ est satisfaisable si et seulement si $I(l) = V$. Une clause C est satisfaisable si au moins un de ses littéraux est satisfaisable sinon elle est insatisfaisable. Étant donnée une instance $A$, le problème de satisfaisabilité (SAT) consiste à tester si toutes les clauses de $A$ sont satisfaisables par une interprétation $I$ donnée. Nous considérons $CF = (C_i)_{i \le m}$ un ensemble de $m$ clauses $C_i$ et $W = (w_i)_{i \le m} \in \mathrm{N}^m$ un vecteur de $m$ entiers $w_i$. A chaque clause $C_i$ on associe un poids $w_i$. Une interprétation $I$ pour la formule $CF$ de poids $W$, détermine un poids d'une valeur $wc(CF,W,I) = \sum_{I(C_i)=V} w_i$. Le problème Max-SAT consiste à déterminer une interprétation $I$ pour la formule $WF = \{CF,W\}$ maximisant la somme des poids des clauses satisfaisables. Si tous les poids $w_i$ sont égaux à 1 alors il s'agit d'un problème Max-SAT non pondéré sinon il est appelé problème Max-SAT pondéré ou simplement Max-SAT. Max-$k$SAT est un problème Max-SAT dans lequel chaque clause contient exactement $k$ littéraux. Max-$k$SAT est un problème *NP*-difficile pour $k \ge 2$ [10].

D'un point de vue théorique, de nombreux algorithmes d'approximation ont été proposés dans le but de réduire les limites théoriques à la recherche de la solution optimale. Un algorithme d'approximation est un algorithme qui produit, en un temps polynomial, une solution $F$ telle que $F \ge \varepsilon F^*$ ($0 < \varepsilon < 1$) où $\varepsilon$ est appelée la garantie de performance et $F^*$ la solution optimale. Par convention, l'algorithme $F$ est appelé algorithme à *ε–approximation*. Il a été prouvé qu'il existe une constante $c < 1$ pour laquelle il n'existe pas d'algorithme à *c-approximation* pour Max-SAT et Max-2SAT à moins que P=NP [1]. Le premier algorithme d'approximation pour Max-SAT décrit par Johnson [17], est un algorithme à 0.5-*approximation*. Goemans et Williamson [13] présentent un algorithme à 0.75-*approximation* puis une version améliorée à 0.7584-*approximation* [14]. Asano *et al.* [2] présentent un algorithme à 0.765-*approximation* en utilisant une approche de programmation semi-définie pour Max-SAT. À notre connaissance, le meilleur résultat obtenu actuellement est un algorithme à 0.7846-*approximation* proposé par Asano et Williamson [3].

En pratique, les limites théoriques des algorithmes d'approximation sont loin d'être atteintes justifiant la recherche de nouvelles heuristiques pour l'approximation de solutions à Max-SAT. Plusieurs algorithmes de recherche locale stochastique ont été proposés et testés pour Max-SAT. L'algorithme le plus populaire est sans aucun doute GSAT [22] (pour Greedy SATisfiability) défini initialement pour le problème SAT, puis appliqué à Max-SAT [24]. GSAT est un algorithme d'amélioration itérative se basant sur le principe de minimisation du nombre de clauses insatisfaites. Une itération GSAT examine les variables impliquées dans une clause insatisfaite afin de choisir celle dont le basculement (mettre $x_i$ à 1 si $x_i$ est à 0 et $x_i$ à 0 si $x_i$ est à 1) minimise le nombre total de clauses insatisfaites. La figure 1 présente une version de l'algorithme GSAT appliquée à Max-SAT non pondéré. Plusieurs variantes ont été proposées pour permettre à GSAT de sortir des minima locaux. GSAT+ ou GSAT avec *random-walk* [24] est une amélioration de GSAT se basant sur l'introduction de mouvements aléatoires. A chaque nouvelle itération, un mouvement de descente GSAT est effectué avec une probabilité fixée de (1 - $p$) et un

```
ALGORITHME MaxGSAT( ∑ , MaxMoves )
{∑ : ensemble de clauses}
{MaxMoves : nombre maximum de déplacements}
{eval (∑ , I) : calcul du nombre de clauses de ∑
insatisfaites par l'interprétation I}
 I ← une interprétation aléatoire pour ∑
 Moves ← 0
 BestError ← eval (∑ , I)
 BestSol ← I
 Tantque (Moves < MaxMoves)
  Si I satisfait ∑ Alors Retourner ( I )
  Sinon
    I ← I dans laquelle la variable maximisant le
       nombre de clauses satisfaites est basculée.
    Moves ← Moves + 1
    Error ← eval (∑ , I)
    Si Error < BestError
         Alors
            Moves ← 0
            BestSol ← I
            BestError ← Error
    Finsi
  FinTantque
  Retourner ( BestSol)
FIN MaxGSAT
```

**Figure 1:** Algorithme GSAT appliqué à Max-SAT.

mouvement aléatoire avec une probabilité $p$, basculant une variable choisie aléatoirement parmi celles impliquées dans une clause insatisfaite. L'utilisation de la probabilité $p$ (appelée bruit) permet de sortir des minima locaux. L'algorithme WSAT (ou WalkSAT) [24] est un descendant de GSAT+ utilisant un sous-ensemble du voisinage de GSAT+. Il existe autant de variantes de WSAT que de manières de choisir une variable dans une clause insatisfaite par l'interprétation courante (Novelty et R-Novelty [20]). Les résultats expérimentaux de GSAT et ses variantes connaissent des succès remarquables. Ces algorithmes sont capables de traiter des instances de milliers de variables alors que les meilleurs algorithmes exacts ne dépassent pas en moyenne 500 variables [24]. L'algorithme WSAT a été adapté par Jiang *et al.* [16] pour la résolution d'instances de Max-SAT. Son application à des encodages du problème d'arbre de Steiner en instances de Max-SAT, montre que cette approche est compétitive avec les meilleurs algorithmes d'arbre de Steiner.

La méthode de recuit simulé [18] s'inspire du processus de recuit physique utilisé en métallurgie pour fabriquer des cristaux (dans lesquels la configuration d'énergie interne est minimale): Un matériau est porté à température élevée, puis refroidi de manière à ce que l'équilibre thermique ait le temps de s'instaurer entre deux décroissances de la température. Le processus se déroule selon la méthode de Metropolis *et al.* [21]: A chaque étape, le nouvel état est accepté si la différence d'énergie $\Delta E$ occasionnée par un déplacement aléatoire diminue. Sinon, il est accepté avec une probabilité définie par $p(\Delta E,T) = \exp(-\Delta E/C_b \times T)$ où $T$ est la température du système et $C_b$ la constante de Boltzmann (loi canonique de Gibbs-Boltzmann). En pratique, l'algorithme retourne la meilleure configuration trouvée lorsque aucune configuration voisine n'a été acceptée pendant un certain nombre d'itérations à une température donnée ou lorsque la température atteint la valeur zéro ou encore lorsqu'un nombre prédéfini de déplacements est réalisé. Son application à Max-SAT peut être vue comme une version étendue de la recherche locale stochastique. A chaque nouvelle itération, un voisin de l'interprétation courante est généré de manière aléatoire. Si ce voisin augmente la somme des poids des clauses satisfaites $wc(CF,W,I)$, il est systématiquement retenu. Sinon, il est accepté avec une probabilité $p(\Delta wc,T)$ qui dépend d'une part de l'importance de la dégradation $\Delta wc$ et d'autre part d'un paramètre de contrôle, la température $T$ définie par une fonction décroissante. La performance de cette méthode dépend largement du schéma de refroidissement utilisé.

La recherche taboue est une méthode proposée indépendamment par Glover [11,12] et Hansen et Jaumard [15]. Cette méthode se base sur la même ossature que GSAT en utilisant une *liste taboue* (mémoire à court terme) d'une longueur $l$ n'autorisant le basculement d'une variable une nouvelle fois qu'après au moins $l$ itérations. Rendre momentanément des variables *taboues* permet de couvrir et d'explorer au mieux l'espace de recherche. En pratique, les performances de l'algorithme dépendent essentiellement d'une taille optimale de la liste taboue. TSAT [19] est un algorithme tabou appliqué au problème SAT pouvant être très compétitif avec WSAT.

## 2- LA METHODE EO

La méthode EO a été introduite par Boettcher et Percus [6] dans le but de résoudre des problèmes d'optimisation combinatoire difficiles. Elle s'inspire du modèle de Bak-Sneppen [4] proposé comme représentation de co-évolution d'espèces pour expliquer le phénomène d'auto-organisation dans l'évolution biologique des espèces. Dans ce modèle une valeur qualitative comprise entre 0 et 1 appelée adaptabilité (ou fitness) est associée à chaque espèce. A chaque itération l'adaptabilité de l'espèce de moindre qualité est remplacée par une valeur aléatoire. Après un nombre suffisant d'itérations, le système atteint un état hautement corrélé appelé *état critique auto-organisé* dans lequel toutes les espèces ont atteint un état d'adaptation optimal. C'est un état globalement stable tout en étant localement instable.

Nous considérons un système décrit par un ensemble de variables $x_i$ auxquelles sont associées des valeurs d'adaptabilité $\lambda_t$. La forme générale de l'algorithme EO est décrite par [6, 7]:

(1) Choisir aléatoirement un état initial pour le système.

(2) Pour un nombre prédéfini d'itérations,

a/ Ranger les variables $x_i$ du système par rapport à leur adaptabilité $\lambda_t$.

b/ Effectuer un mouvement dans l'espace de recherche en modifiant la valeur de la variable ayant la plus mauvaise adaptabilité $\lambda_i \in [0,1]$.

Le rangement des variables permet, d'une part de conserver les portions les mieux adaptées de la solution, et d'autre part de mettre à jour les variables ayant les plus

mauvaises adaptabilités. Différentes configurations de l'espace de recherche sont explorées et la qualité globale de la solution est améliorée. Une extension de cette procédure, notée $\tau$-EO, [6, 7, 8] range les variables du système de 1 à $N$ selon leur adaptabilité croissante $\lambda_i$. Une loi de distribution de probabilité sur les rangs des variables est considérée : $P(n) \propto n^{-\tau}, 1 \le n \le N$ pour $\tau$ donné. A chaque itération, une variable de rang $k$ correspondant à $P(k)$, est sélectionnée et son état est modifié. Selon la loi de probabilité $P$, la variable ayant la plus mauvaise adaptabilité (d'indice 1) est choisie plus fréquemment. Un biais par rapport aux variables de mauvaise adaptabilité est maintenu et aucun rang n'est complètement exclu du choix. Les performances du processus de recherche dépendent du choix de la valeur du paramètre $\tau$. Pour $\tau = 0$, la recherche s'apparente à une marche aléatoire où toutes les configurations de l'espace de recherche sont équiprobablement atteignables. Pour de grandes valeurs de $\tau$, la recherche se focalise sur le choix de variables de moindre adaptabilité et est alors confinée dans des minima locaux. Pour estimer la valeur optimale de $\tau$, Boettcher et Percus [8] ont établi une relation entre $\tau$, le temps d'exécution $t$ et le nombre de variables $N$ du système ; indépendamment de la nature intrinsèque du problème. Pour $t = A \cdot N$ où $A$ est une constante, on a :

$$\tau \sim 1 + \frac{\ln\left(A/\ln(N)\right)}{\ln(N)} \quad \left(N \to \infty,\ 1 << A << N\right) \qquad (1)$$

Au voisinage de cette valeur optimale, les variables de meilleure adaptabilité ne sont pas complètement exclues du processus de sélection. De nombreuses configurations de l'espace de recherche deviennent accessibles permettant d'obtenir de meilleures performances.

L'algorithme EO s'articule autour d'un schéma de recherche locale comparable à GSAT. La différence essentielle entre ces deux algorithmes réside dans le calcul de leurs mouvements. EO se déplace dans l'espace de recherche en se basant sur l'adaptabilité intrinsèque des variables. GSAT parcourt l'espace en se basant sur un calcul anticipé de mouvements.

## 3- $\tau$-EO POUR Max-SAT

Un inconvénient majeur de la méthode EO réside dans la difficulté de définir de façon significative l'adaptabilité d'une variable du système. Si ces variables sont hautement interconnectées alors le processus de réévaluation de leur adaptabilité à chaque itération, ralentit considérablement le traitement.

Nous considérons une instance du problème Max-SAT à $n$ variables booléennes et $m$ clauses pondérées. $I$ étant la solution courante. L'adaptabilité $\lambda_i$ d'une variable $x_i$ est définie par le rapport de la somme des poids des clauses satisfaites par $I$ et contenant $x_i$ à la somme totale des poids des clauses :

$$\lambda_i = \frac{\sum_{x_i \in C_j \text{ and } I(C_j)=V} w_j}{\sum_{k=1}^{m} w_k} \qquad (2)$$

L'adaptabilité $\lambda_i$ d'une variable $x_i$ définit sont coût. La fonction de coût $C(I)$ de la solution courante $I$ est définie par la somme des coûts individuels des variables. La fonction objective (maximiser la somme des poids des clauses satisfaites par $I$ ) revient à minimiser $C(I)$, d'où

$$C(I) = -\sum_{i=1}^{n} \lambda_i \qquad (3)$$

La figure 2 présente brièvement l'algorithme $\tau$-EO_Max-SAT. La recherche commence à une interprétation aléatoire $I$ et se poursuit pendant un nombre *MaxSteps* d'itérations prédéfini. Chaque étape du processus correspond à un déplacement dans l'espace de recherche selon l'heuristique $\tau$-EO. La meilleure interprétation courante *BestSol* est associée au maximum courant de la somme des poids des clauses satisfaites. La procédure Update (*BestSol*, *Unsat*, *WUnsat*) met à jour l'ensemble des clauses insatisfaites *Unsat* par la meilleure solution courante *BestSol* et leur poids total *WUnsat*.

**ALGORITHME** EO _Max-SAT (∑, *MaxSteps*, $\tau$)
{∑ : ensemble de clauses}
{ *MaxSteps* : nombre maximum d'itérations}
{$\tau$ : paramètre libre}
{Update (*BestSol*, *Unsat*, *WUnsat*) : mise à jour des clauses insatisfaites par *BestSol* et de leurs poids}
$I \leftarrow$ une interprétation aléatoire pour les variables de ∑
*BestSol* $\leftarrow I$
*Unsat* $\leftarrow$ clauses insatisfaites par $I$
*WUnsat* $\leftarrow$ somme des poids des clauses de *Unsat*
**Pour** $k$=1 à *MaxSteps* **faire**
  **Si** $I$ satisfait ∑ **Alors** Retourner ($I$)
  Evaluer $\lambda_i$ pour chaque variable $x_i$ de ∑ (Eqn. 2)
  Ordonner les variables $x_i$ selon l'ordre croissant des valeurs de $\lambda_i$ .
  Choisir un indice $j$ avec une probabilité $P(j) \propto j^{-\tau}$
  $I' \leftarrow I$ dans laquelle la valeur de la variable $x_j$ est basculée.
  **Si** $C(I') < C(BestSol)$ **Alors** $BestSol \leftarrow I'$
  $I \leftarrow I'$
  Update (*BestSol*, *Unsat*, *WUnsat*)
**FinPour**
Retourner (*BestSol*, $C$(*BestSol*), *Unsat*, *WUnsat*)
**FIN** EO_Max-SAT

**Figure 2:** Algorithme $\tau$-EO_Max-SAT.

## 4- RESULTATS EXPERIMENTAUX

Les algorithmes considérés dans notre étude comparative sont GSAT+ [24], une version du recuit simulé (SA) [15] et une version de la recherche taboue appelée SAMD (Steepest Ascent Mildest Descent) [15]. Pour comparer les résultats expérimentaux obtenus par l'algorithme EO_Max-SAT avec ceux des algorithmes SA et SAMD, nous avons utilisé le même jeu de tests décrit dans [24, 5]. Il comporte des instances aléatoires de Max-3SAT et Max-4SAT produites par le générateur aléatoire disponible à l'URL http://www.cs.cornell.edu/home/selman/sat/sat-package.tar. Les instances aléatoires de formules CNF sont souvent utilisées pour l'évaluation de procédures de satisfaisabilité

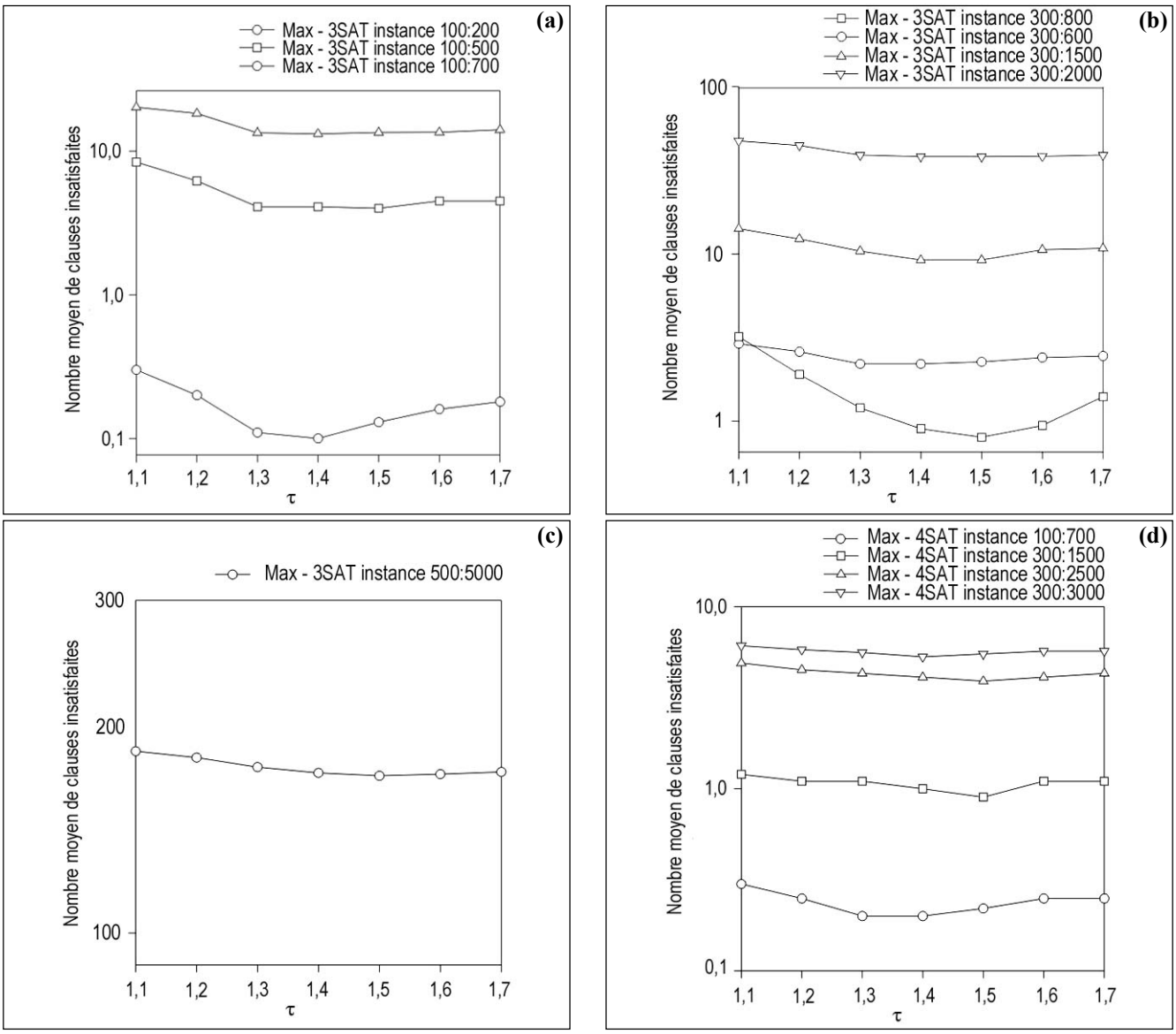


**Figure 3:** (a), (b), (c), (d) Courbes semi-logarithmiques des variations du nombre moyen de clauses insatisfaites en fonction du paramètre $\tau$.

car d'une part elles sont facilement générées, et d'autre part elles ne renferment aucune structure cachée inhérente à un problème particulier.

Les instances aléatoires Max-3SAT considérées sont (100,200), (100,500), (100,700), (300,600), (300,800), (300,1500), (300,2000) et (500,5000) où ($n$, $m$) désigne $n$ variables et $m$ clauses. Pour Max-4SAT, nous avons testé les instances aléatoires suivantes : (100,700), (300,1500), (300,2500) et (300,3000).

$\tau$ est le seul paramètre libre contrôlant les performances de $\tau$-EO-Max-SAT. Pour estimer sa valeur moyenne optimale, nous avons évalué pour chaque problème du jeu de tests, le nombre moyen de clauses insatisfaites sur un ensemble de 50 instances aléatoires. Pour chaque instance, le nombre d'itérations *MaxSteps* est fixé à 1000$n$. Les courbes de la figure 3 (a, b, c, d) représentent les variations du nombre moyen de clauses insatisfaites en fonction du paramètre $\tau$. Sur l'ensemble des instances, les meilleurs résultats sont obtenus pour des valeurs de $\tau$ comprises entre 1.3 et 1.5. Ceci correspond aux valeurs optimales obtenues par Boettcher et Percus [8] dans le cas du problème de partitionnement de graphe.

Les tableaux 1 et 2 représentent, respectivement, le nombre moyen de clauses insatisfaites obtenu pour chaque instance de Max-3SAT et de Max-4SAT. Les deux premières lignes de chaque tableau sont tirées de [15] et les troisièmes lignes sont tirées de [24]. Elles représentent des résultats obtenus respectivement par les méthodes SA, SAMD et GSAT+. Les deux lignes ajoutées aux tableaux, représentent les résultats que nous avons obtenus, respectivement, avec EO_Max-SAT (choix systématique de la variable ayant la plus mauvaise adaptabilité) et $\tau$-EO_Max-SAT ($\tau$ = 1.4). Pour chaque couple ($n$, $m$), 50 instances sont générées aléatoirement et chaque algorithme est exécuté 10 fois pour chaque instance pendant un temps d'exécution suffisamment long pour ne plus avoir d'amélioration possible. Pour l'ensemble des instances de Max-3SAT et Max-4SAT les nombres moyens de clauses

**Tableau 1:** Nombre moyen de clauses insatisfaites obtenu pour des instances aléatoires Max-3SAT.

| Variables<br>Clauses | 100<br>200 | 100<br>500 | 100<br>700 | 300<br>600 | 300<br>800 | 300<br>1500 | 300<br>2000 | 500<br>5000 |
|---|---|---|---|---|---|---|---|---|
| SA | 0.2 | 8.2 | 18.1 | 2.7 | 6.1 | 30.0 | 58.0 | 226.4 |
| SAMD | 0.3 | 5.1 | 14.7 | 2.4 | 4.3 | 15.3 | 39.0 | 182.8 |
| GSAT+($p$=0.5) | 0 | 2.9 | 12.9 | 0 | 0 | 8.1 | 34.9 | 163.6 |
| EO | 0.2 | 6.4 | 15.3 | 2.8 | 3.2 | 12.1 | 46.2 | 175.1 |
| $\tau$-EO ($\tau$=1.4) | 0.1 | 4.1 | 13.2 | 2.2 | 0.9 | 9.2 | 38.2 | 169.7 |

**Tableau 2:** Nombre moyen de clauses insatisfaites obtenu pour des instances aléatoires Max-4SAT.

| Variables<br>Clauses | | 100<br>700 | 300<br>1500 | 300<br>2500 | 300<br>3000 |
|---|---|---|---|---|---|
| SA | (meilleur) | 0.3 | 1.0 | 11.9 | 14.3 |
| SAMD | (meilleur) | 0 | 0 | 0.4 | 5.9 |
| | (moyen) | 0.1 | 1.3 | 3.8 | 9.4 |
| GSAT+ ($p$=0.5) | (meilleur) | 0 | 0 | 0 | 0.9 |
| | (moyen) | 0 | 0 | 0 | 2.2 |
| EO | (meilleur) | 0.2 | 1.0 | 4.8 | 6.1 |
| | (moyen) | 0.3 | 1.2 | 5.6 | 7.2 |
| $\tau$-EO ($\tau$=1.4) | (meilleur) | 0 | 0.2 | 0.3 | 3.2 |
| | (moyen) | 0.2 | 1.0 | 4.1 | 5.3 |

insatisfaites obtenus par $\tau$-EO_Max-SAT ($\tau$ = 1.4) sont, en moyenne, meilleurs que ceux obtenus par SA et SAMD. Cependant, ils sont légèrement moins performants que ceux obtenus par GSAT+. Ceci nous a amené à étudier la variation du nombre moyen de clauses insatisfaites en fonction du nombre d'itérations obtenue par $\tau$-EO_Max-SAT et GSAT+. La figure 4 (a, b) montre les résultats obtenus sur les instances 300:2000 et 500:5000 de Max-3SAT. Pour un nombre d'itérations réduit, $\tau$-EO_Max-SAT produit de meilleures solutions que GSAT+. À long terme, les performances de $\tau$-EO_Max-SAT stagnent et de meilleures solutions sont produites par GSAT+. Nous supposons que cette contre-performance soit associée à la définition de l'adaptabilité d'une variable dépendant naturellement de la fonction objective du problème. Il est, cependant, nécessaire de continuer les tests expérimentaux sur des instances plus larges pour mieux identifier les facteurs supposés influencer les performances.

## CONCLUSION

Dans cet article, nous avons présenté un nouvel algorithme $\tau$-EO_Max-SAT se basant sur la dynamique des systèmes auto-organisés pour l'approximation de solutions au problème Max-SAT. Les tests expérimentaux montrent une amélioration significative des résultats produits par $\tau$-EO_Max-SAT en comparaison avec ceux des algorithmes de recuit simulé et de recherche taboue sur des instances aléatoires de Max-SAT. Notre algorithme produit de meilleures solutions que GSAT+ (une variante de la procédure GSAT) pour un même nombre d'itérations réduit. En revanche, ses performances stagnent pour un nombre élevé d'itérations. Nous envisageons d'étudier son comportement en utilisant une définition différente pour l'adaptabilité d'une variable et de comparer ses performances avec d'autres algorithmes de recherche locale stochastique de la même famille que GSAT+ tels que WSAT et R-Novelty.

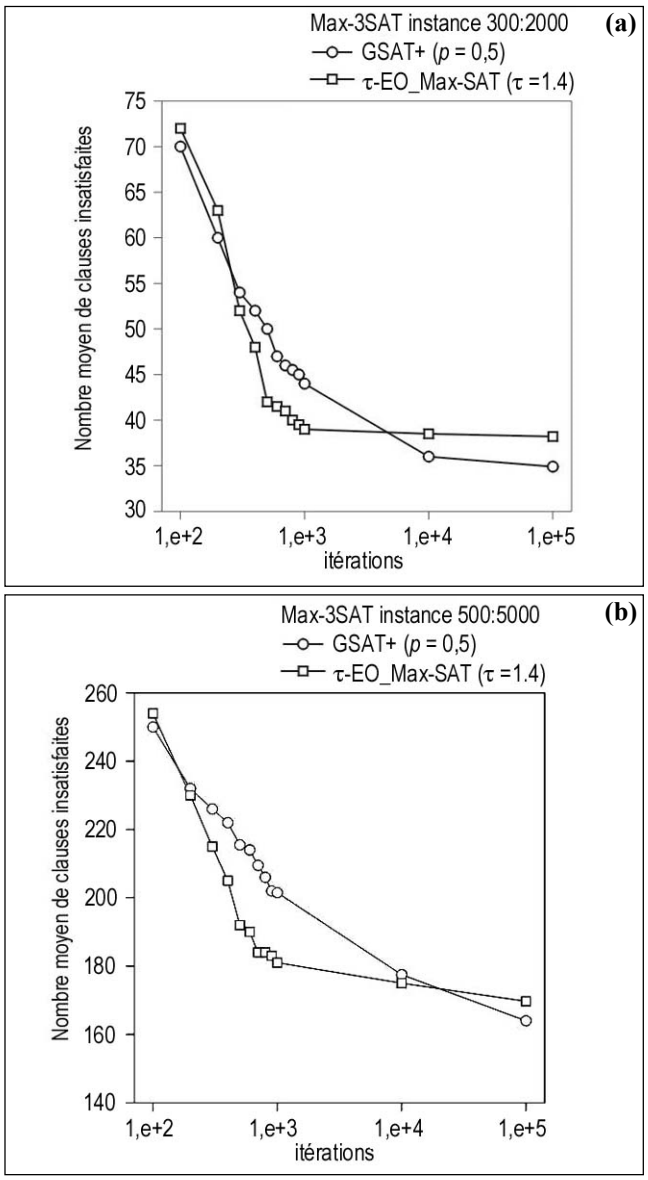


**Figure 4:** (a), (b) Courbes des variations du nombre moyen de clauses insatisfaites en fonction du nombre d'itérations.

***Remerciements:***

*Nous souhaitons remercier les rapporteurs de cet article pour leurs critiques et leurs remarques. Nous remercions également N. Jouandeau du laboratoire d'IA de l'université de Paris 8, pour sa relecture de cet article qui a permis d'améliorer sa présentation et sa lisibilité.*

## REFERENCES

**[1]-** Arora S., Lund C., Motwani R., Sudan M., Szegedy M., "Proof verification and hardness of approximation problems", in Proceedings of the 33rd Annual Symposium on Foundations of Computer Science, (1992), pp. 14-23.

**[2]-** Asano T., Ono T., Hirata T., "Approximation algorithms for the maximum satisfiability problem", *Nordic Journal of*

*Computing*, 3, (1996), pp. 388-404.

**[3]-** Asano T., Williamson D.P., "Improved approximation algorithms for MAX SAT", *Journal of Algorithms*, 42(1), (2002), pp. 173-202.

**[4]-** Bak P., Sneppen K., "Punctuated equilibrium and criticality in a simple model of evolution", *Physical Review Letters*, 71, (1993), pp. 4083-4086.

**[5]-** Battiti R., Protasi M., "Reactive search, a history-sensitive heuristic for Max-SAT", *ACM Journal of Experimental Algorithmics* 2, (1997), p.2.

**[6]-** Boettcher S., Percus A.G., "Nature's way of optimising", *Artificial Intelligence*, 119, (2000), pp. 275-286.

**[7]-** Boettcher S., Percus A.G., "Optimization with extremal dynamics", *Physical Review Letters*, Vol.86, N°23, (2001), pp. 5211-5214.

**[8]-** Boettcher S., Percus A.G., "Extremal optimization for graph partitioning", *Physical Review E*, Vol.64, 026114, (2001), pp. 1-13.

**[9]-** Cook S.A., "The complexity of theorem proving procedures", in Proceedings of the 3rd Annual ACM Symposium of the Theory of Computation, (1971), pp. 151-158.

**[10]-** Garey M., Johnson D., "Computers and intractability: A guide to the theory of NP-completeness", W.H. Freeman and Company, New York, (1979).

**[11]-** Glover F., "Tabu search : Part I", *ORSA Journal on Computing*, 1 (3), (1989), pp. 190-206.

**[12]-** Glover F., "Tabu search : Part II", *ORSA Journal on Computing,* 2 (1), (1989), pp. +32.

**[13]-** Goemans M., Williamson D.P., "A new $\frac{3}{4}$-approximation algorithm for the maximum satisfiability problem", *SIAM Journal on Discrete Mathematics*, 7, (1994), pp. 656-666.

**[14]-** Goemans M., Williamson D.P., "Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming", *Journal of the ACM*, 42, (1995), pp. 1115-1145.

**[15]-** Hansen P., Jaumard B., "Algorithms for the maximum satisfiability problems", *Computing*, 44, (1990), pp. 279-303.

**[16]-** Jiang Y., Kautz H., Selman B., "Solving problems with hard and soft constraints using a stochastic algorithm for Max-SAT", in Proceedings of the 1st International Joint Workshop on Artificial Intelligence and Operational Research, (1995).

**[17]-** Johnson D., "Approximation algorithms for combinatorial problems", *Journal of Computer and System Sciences*, 9, (1974), pp. 256-278.

**[18]-** Kirkpatrick S., Gelatt C.D., Vecchi P.M., "Optimization by simulated annealing", *Science*, 220, (1983), pp. 671-680.

**[19]-** Mazure B., Sais L., Gregoire E., "Tabu search for SAT", in Proceedings of the 14th National Conference on AI, AAAI-1997, (1997), pp. 281-285.

**[20]-** McAllester D., Selman B., Kautz H., "Evidence for invariants in local search", in Proceedings of the 14th National Conference on AI, AAAI-1997, (1997), pp. 321-326.

**[21]-** Metropolis N., Rosenbluth A.W., Rosenbluth M.N., Teller H., Teller E., "Equation of state calculations by fast computing machines", *Journal of Chemical Physics*, 21, (1953), pp. 1087-1092.

**[22]-** Selman B., Kautz H., "An empirical study of greedy local search for satisfiability testing", in Proceedings of the 11th National Conference on AI, AAAI-1993, (1993), pp. 46-51.

**[23]-** Selman B., Kautz H., "Domain independent extensions to GSAT : Solving large structured satisfiability problems", in Proceedings of the IJCAI-93, (1993), pp. 290-295.

**[24]-** Selman B., Kautz H., Cohen B., "Noise strategies for improving local search", in Proceedings of the 12th National Conference on Artificial Intelligence, AAAI-1994, (1994), pp. 337-343.

**[25]-** Spears W.M., "Simulated annealing for hard satisfiability problems", in D.S. Johnson and M.A. Trick editors, In Cliques, Coloring and Satisfiability : Second DIMACS Implementation Challenge, (1996), pp. 553-558.

**[26]-** Yannakakis M., "On the approximation of maximum satisfiability", *Journal of Algorithms*, 17, (1994), pp. 475-502. □