**Université Saad Dahleb Blida**

Faculté : des Sciences

Département : Informatique

## Diplôme préparé : Licence

Spécialité : **SIQ**

Thème :

Problème de SAT

*Encadreur :* Dr. Chemchem M A

*Binôme :*

\_Touil Abdelwahab

\_Tidafi Omar Abdelaziz

*Groupe :* 3

Année en cours : 2016/2017

1. Résumé

L’objectif de cette étude est de faire une recherche comparative entre les méta-heuristiques avec des approches originales pour la résolution du problème MAX SAT. Et de proposer des méthodes pour la résolution de ce problème de satisfiabilité booléenne ou problème de décision. Tout en se focalisant sur la conception d'une approche hybridée avec une méta-heuristique afin de booster les performances en matière de temps de réponse. Et en contournant la complexité du problème. Et en fin élaborer une étude expérimentale.

Au but de la réalisation de notre objectif, nous avons proposé et étudié plusieurs travaux déjà existant qui nous ont inspiré à proposer deux approches essentielles. Notre première approche consiste à implémenter un algorithme basé sur la procédure DPLL. Qui teste des benchmarks et détermine si ces derniers sont satisfiable ou non, en testant toutes leurs closes et en prenant compte du temps d’exécution de cette opération afin de le comparé sa performance avec d’autre algorithmes, Plus de délaies sont proposées dans la littérature du rapport. Notre deuxième approche consiste à proposer un algorithme hybridé Algorithme Génétique, basé sur quelques notions d’optimisations en appliquons l’opération de croisement de l’optimisation génétique, le principe de changement de gènes dans l’algorithme de mutation et tout en utilisant le maximum d’itération. L’objectif de cette hybridation est double: premièrement, proposer un algorithme original permettant la résolution de problèmes d’optimisation de satisfiabilité des clauses, avec des résultats optimaux dans une courte durée, contrairement à l’algorithme de satisfaction DPLL conçu pour la résolution de problèmes de satisfiabilité. Deuxièmement, donnée à l’utilisateur plus de perception et d’interaction avec un nombre de paramètres riche, précis et essentiels pour le fonctionnement de l’algorithme (taille de la population initiale, taux de mutation, taux de croisement et le max itération). Les deux approches sont fêtent Pour faire face à une grande gamme de problèmes pouvant être modélisés par des structures binaires.

Afin de tester de manière optimale les performances des algorithmes (les approches) que nous avons proposés, nous avons boosté pendant les tests le processus javaw.exe pour plus d’accès au processeur et on augmente la quantité de mémoire allouée à Java pour de meilleur résultat. Et on a fais en sorte que les résultats de chaque performance soit enregistré automatiquement dans des fichiers de manière à facilité le calcule des statistique de leur performances. Et pour que notre travaille soit expérimenté en tout simplicité on a fais en sorte qu’il donne des graphes des défirent résultat pour les comparer avec d’autres travaux.

**Mots clés:** méta-heuristiques, Optimisation, Hybridation, Benchmarks, clauses.

1. Remerciements

On tient à remercier toutes les personnes qui ont contribué de près ou de loin au succès de notre projet. Et qui nous ont aidés lors de ca réalisation et lors de la rédaction de ce rapport.

Tout d'abord, On’ adresse nos remerciements a **nos chers parents** qui sans eux rien de cela ne serait possible, Ils nous ont soutenue tout au long de notre parcours et nous ont offert un environnement de travail idéale.

En suite, On tient à remercier vivement notre encadreur **Dr. Chemchem M A,** qui nous a beaucoup aidés dans nos recherches, pour le temps passé ensemble et pour le partage de son expertise. **Dr. Chemchem,** qui nous a orientés, cancellé et suivie tout au long de notre projet. Ce qui nous a permis d’acquérir beaucoup de connaissances indispensables dans la réalisation des différentes étapes du projet SAT.

On tient aussi à remercier nos camarades, **les autres binômes**, qui nous ont éclairés sur certains points, pour leur esprit d’équipe. Ils furent d'une aide précieuse dans les moments les plus délicats.

Et à remercier cordialement **Mme. Oukid L**, pour son accueil, et son module ‘Rédaction Scientifique’, grâce à quoi on a pue rédigé un bon rapport.

Enfin, On tient à remercier toutes les personnes qui nous ont soutenues, aidés et ont contribué même de loin, à la création de notre projet SAT.

1. Table des matières
2. Résumé…………………………………………………….....3.
3. Remerciements…………………………………………...…..4.
4. Table des matières……………………………………………5.
5. Liste des figures……………………………………………...6.
6. Liste des tableaux…...………………………………………..6.
7. Introduction…………………………………………………..7.
8. Etat de l’art…………………………………………………...8.

* Introduction……………………………………………8.
* Finition et situation du travail………………………….8.
* Travaux déjà publiée…………………..……………….9.
* FlipGA…………………………………………..9.
* MiniSAT……………………………………….11.
* Avantage……………………………………………....12.

1. Les approches proposées……………………..…………...…13.

1er Approches ‘ DPLL ’…………………………………...13.

2eme Approche ‘ l’Algorithme Génétique ’...…...…….…15.

1. Evaluation et expérimentation………………………………..19.
2. Outils et méthodes…………………………………………....5.
3. Conclusion…………………………………………………....8.
4. Références….…………………………………………………9.
5. Annexes……………………………………………………...10.
6. Liste des figures

**FIGURE 1 :** " Algorithme du solveur MiniSAT "

**FIGURE 2 :** " Optimiser du texte le texte du benchmark "

**FIGURE 3 :** " exemple de satisfiabilité de deux close "

**FIGURE 4 :** " Séparation des clause d’un individu dans un tableau "

**FIGURE 5 :** " un tableau d’un individu avec ses variable en binaire "

**FIGURE 6 :** " Tableau de résultat d’un individu "

**FIGURE 7 :** " Croisement sur un point "

**FIGURE 8 :** " Mutation avec un point ".

**FIGURE 9 :** " schéma qui explique le déroulement de l’algorithme génétique ".

**FIGURE 10 :** " résultats des tests de l’algorithme DPLL sur des Benchmarks ".

1. Liste des tableaux

**Tableau 1 :** " Comparaison des différents algorithmes évolutionnaires "

**Tableau 2 :** " résultats en teste DPLL en milliseconde et en minutes "

1. Introduction

     Depuis les débuts du 21em siècle, les chercheurs informatiques n'ont cessé d'essayer de représenter le plus fidèlement possible ce qui nous entourait, allons jusqu’a crée une science rien que pour ca nommée ‘l'intelligence artificielle’. La ils ont eu beaucoup de difficultés, Et l’un des problèmes les plus standards et très connu dans la communauté de l'intelligence artificielle, c’est Le problème de satisfiabilité SAT.

En informatique théorique, le problème SAT ou problème de satisfiabilité booléenne est le problème de décision, qui, étant donné une formule de logique propositionnelle, détermine s'il existe une assignation des variables propositionnelles qui rend la formule vraie.

Ce problème est aussi très important en théorie de la complexité. Il a été mis en lumière par le théorème de Cook, qui est à la base de la théorie de la NP-complétude.

Le problème SAT a aussi de nombreuses applications notamment en satisfaction de contraintes, planification classique, model chocking, diagnostic, et jusqu'au configurateur d'un PC ou de son système d'exploitation: on se ramène à des formules propositionnelles et on utilise un solveur SAT.

Grasse à cette étude, On pourra étudier, implémenter et booster des algorithmes de résolution de ce problème tel que DPLL et ‘l’algorithme génétique’ et d’autres à la grande échelle.

Dans le cadre de ce projet, On vat développer des approches originales basées sur les méta-heuristiques pour la résolution de la variante de ce problème MAX SAT.

La première originalité du travail consiste à se focaliser sur la conception d'une approche hybridée avec une méta-heuristique afin de booster les performances en matière de temps de réponse.

La deuxième est de contourner la complexité du problème et d'aller vers plus de stabilité et plus de simplécité. Une étude expérimentale doit être élaborée pour étayer la performance des approches proposées.

1. Etat de l’art

* Introduction :

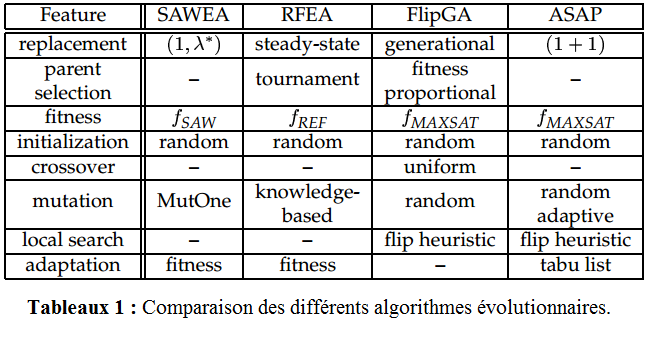
Afin de réaliser une étude comparative entre les travaux réalisé et ceux déjà existant, Cette partie est faite. C’est que le choix des travaux au moment de réaliser les spécifications n’était pas assez diversifié. Les travaux ne sont pas tous complet, et certains utilisent la même manière fondamentale de résoudre les problèmes. Difficile de comparer de manière précise si les adversaires ont beaucoup de similarités. On a jugé bon de prendre **FlipGA** qui est un des travaux SAT qui utilise des algorithmes génétiques. Et **MiniSAT** basés sur la procédure DPLL.

* Finition et situation du travail :

Aujourd'hui, de nombreux chercheur on pu en venir à bout du problème SAT mais leurs solution proposé diffère en matière de temps de résolution, plus c’est efficace plus ca coute en temps de résolution. Donc ce n’est plus une question de résoudre mais de proposer des approches meilleures, en prenons compte du développement de matérielle qui optimise la qualité du travail.

Et grasse des nombreuses recherches on a pu réaliser un travaille qui soit assez performent pour réaliser des résolutions sur des benchmarks de grande envergure et cela en un temps plutôt raisonnable. Et afin que notre travaille soit expérimenté en tout simplicité on a fais en sorte qu’il donne des graphes des défirent résultat pour les comparer avec d’autres travaux.

* Travaux déjà publiée :

 Comparaison des différents algorithmes évolutionnaires :

* **FlipGA (Marchiori, Rossi, 1999):**

FlipGA a été l’inspiration principale pour SAT durant nos recherche, il explique en quoi chacune de ses fonctionnalités permet d’améliorer les résultats, c’est donc en ça que SAT en est une inspiration. En voici l’article de présentation : <http://www.cs.ru.nl/~elenam/fsat.pdf>.

**Les Fonctionnalités de FlipGA :**

**.Replacement : Generational** Le principe étant de se servir des meilleurs (selon la fitness) de la population précédente pour générer les enfants de la population suivante.

**.Parent Selection : Fitness proportional** La sélection des parents dans FlipGA fonctionne selon le principe de roulette wheel 1

**.Fitness :** **FmaxSAT** La fitness de chaque individu se calcule de la manière suivante : () ainsi un individu ayant une fitness de 1.00 sera un individu solution du problème.

**.Initialisation : Random** très simplement, pour initialiser chaque individu au sein de la population, on leur affecte une assignation choisie aléatoirement. La sélection naturelle va faire en sorte que les mauvais disparaissent au profit des bons.

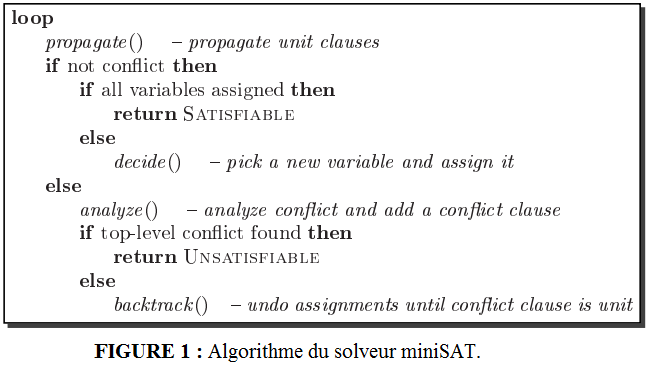
.**Crossover : uniform** FlipGA, tout comme SATyr, utilise un croisement d’individus uniforme. Cela signifie que lorsque l’on prend 2 parents, l’enfant va posséder en moyenne 50% du parent 1 et 50% du parent 2.

**.Mutation : random** Pour réaliser une mutation sur un individu, on va sélectionner une variable au hasard, et on va inverser sa valeur.

.**Local search : flip heuristic** fonctionne comme la mutation, sauf que les changements de valeur de variables sont réalisés dans l’optique d’améliorer la fitness de l’individu.

* **MiniSAT (Eén et Sörensson, 2005) :**

Ses fonctionnalités étaient les plus performantes de la compétition des algorithmes révolutionnaires qui a eu lieu en 2005 comme le prouve ces différentes coupes et trophées gagnés par MiniSAT, quatre trophées dans la résolution du problème MaxSAT en 2005. Pour apprendre beaucoup plus en détails comment fonctionne ce dernier, en voici l’article de recherche : <http://minisat.se/downloads/MiniSat.pdf>.

 MiniSAT est basée sur la procédure ou l'algorithme de Davis\_Putnam\_Logemann\_Loveland (DPLL). Il a la particularité, en comparaison avec les autres travaux sur DPLL, de générer beaucoup de conflits au début de la résolution, et ainsi être très performant sur les problèmes ‘évidemment UNSAT’ (là où les conflits sont rapidement identifiables).

* Avantage :

On peut constater que grasse aux algorithmes évolutionnaires talque FllipGA, MiniSAT et autres... On a pu faire un énorme progrès concernons la résolution du problème SAT ou problème de satisfiabilité booléenne ou en générale le problème de décision. Les deux algorithmes sont fêtent Pour faire face à une grande gamme de problèmes pouvant être modélisés par des structures binaires.

FlipGA nous offre des Fonctionnalités qui permettent d’améliorer les résultats. Car FlipGA a concentré son travaille non pas que sur la résolution du problème mais aussi en améliorant les méthodes de résolution. Il c’est basé sur les notions d’optimisations talque l’opération de croisement de l’optimisation génétique, le principe de changement de gènes dans l’algorithme de mutation la notion de Fitness proportionnel, le Local search (flip heuristic) et tout en utilisant le maximum d’itération.

Et grasse a ses performance, on peu obtenir des résultats optimal.

MiniSAT, conte a lui, contrairement aux autres travaux de la procédure de DPLL, génère beaucoup de conflits au début de la résolution conduisons a des résultats meilleure sur les problèmes ‘évidemment UNSAT’.

1. Les approches proposées

Pour la résolution du problème Max SAT, on s’est focalisé lord de notre projet sur deux approches distinctes :

* Une approche classique, l’implémentation de l'algorithme de **Davis\_Putnam\_Logemann\_Loveland (DPLL).**
* Et approche plus ou moins perfectionné, l’implémentation **de l’algorithme génétique**.

**1er Approches ‘ DPLL ’ :**

L’algorithme de Davis\_Putnam\_Logemann\_Loveland (DPLL) est un algorithme de Backtracking, complet, de résolution du problème SAT.

**.Le principe :**

On a implémenté un algorithme permettant de prendre en comme seul paramètre un Benchmark de Clauses. Afin de procédé aux tests sur ce dernier, on Donne des normes binaires aux variables du benchmark. Et en suit, on fait en sorte de satisfaire toutes ses clauses du benchmark. Et après une période de temps raisonnable (qui dépond de la talle de notre benchmark) notre algorithme nous affiche comme résultats si ce benchmark est satisfiable ou non, avec la durée de toms qu’a prie cette opération.

**.Pour cela on a suivie les étapes suivantes :**

1. **traitement de texte :**

On a procédé au le traitement de texte du Benchemark, de tel sorte qu’on a extrait le nombre des variable, le nombre des clauses, le nombre de variable dans chaque clause et en fin on a prie les clauses et on les a écrie de la manière suivante :

|  |
| --- |
| Une close  (X\_n1 + -X\_n2 +...+ X\_nN) \* (-X\_n1 + X\_n2 +…+ X\_nN) \* …..….. \* (X\_n1 + X\_n2 +...+ X\_nN) /n /n Fin /n |

**FIGURE 2 :** " Optimiser du texte le texte du benchmark ".

Pour que notre procédure DPLL les reconnaisse.

1. **Satisfaire les closes :**

Rendre les variables binaires :

On prend chaque close du tableau et on initialise ses variable d’une façon binaire (une variable prend 1 ou 0), On donne a chaque variable déjà initialiser dans le Benchmark la même initialisation, par exemple :

X\_n’ ’ initialisé a ‘ 1 ’ dans une close précédente, alors on initialise à ‘ 1 ’ chaque valeur similaire retrouvé dans le Benchmark, Et ‘ 0 ‘ si on la retrouve en négatif (‘ –X\_n’ ‘).

En même temps, Notre procédure DPLL teste les déférant closes du benchmark, de manière que la ou on trouve ‘ + ‘ exécute le ‘ OU Logique‘ entre ses variables binaires, et la ou en trouve ‘ \* ’on exécute le ‘ ET Logique ’ avec priorité aux ‘ ( ‘ et aux ‘ OU Logique‘.par exemple :

|  |
| --- |
| Deux closes :  (1+1+0+0) \* (0+1+0+0) 🡪 1 U 1 U 0 U 0  ∧ 0 U 1 U 0 U 0    🡪 1  ∧ 1    🡪 1 |

**FIGURE 3 :** " exemple de satisfiabilité de deux close ".

Et il essaie touts les approches possibles (à chaque fois il change l’initialisation des variable binaires) jusqu'à ce qu’il trouve un cas ou les close sont toutes satisfiables (après traitement toutes = 1), alors il donne comme résultat que le benchmark est satisfiable et il indique le temps du traitement.

Si il teste touts les approches possibles sans trouve un cas ou les close sont toutes satisfiables (après traitement toutes = 1), alors il donne comme résultat que le benchmark est satisfiable et il indique le temps du traitement.

**2ème Approche ‘ l’Algorithme Génétique ’ :**

Ses fonctionnalités permettent d’améliorer les résultats de satisfiabilité dans le problème SAT.

**.Le principe :**

On a implémenté un algorithme permettant de prendre en paramètre un taille de population initiale, taux de mutation, taux de croisement, le max itération et un Benchmarks de Clauses, Donné des normes binaire aux variables de ses clause et en suit de satisfaire le maximum de clause tout en prônant en complet de la population initiale, le taux de mutation, le taux de croisement, et le max itération.

Et à la fin, on offre comme résultats le temps d’exécution, le pourcentage de taux réussite et un tableau qui classe les solutions en considérant leur fitness.

**.Pour cela on a suivie les étapes suivantes :**

1. **traitement de texte :**

On a procédé le traitement de texte du Benchmark, de tel sorte qu’on a extrait le nombre des variable, le nombre des clauses, le nombre de variable dans chaque clause et en fin on a prie les clauses et on les a écrie mit dans un tableau de manière suivante :

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| séparations  Une close   |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | N1 | N2 | …. | Nn | 0 | N1 | N2 | …. | Nn | 0 | ……… | 0 | N1 | N2 | …. | Nn |   .On a séparé les clauses par des 0.  .On a placé chaque numéro selon sa pelasse dans sa close  .Et on a placé chaque close selon sa pelasse dans le Benchmark. |

**FIGURE 4 :** " Séparation des clause d’un individu dans un tableau ".

Rendre les variables binaires :

On prend chaque close du tableau et on initialise ses variable d’une façon binaire (une variable prend 1 ou 0), On donne a chaque variable déjà initialiser dans le Benchmark la même initialisation, par exemple :

‘ N’ ’ initialisé a ‘ 1 ’ dans une close précédente, alors on initialise à ‘ 1 ’ chaque valeur similaire retrouvé dans le Benchmark, Et ‘ 0 ‘ si on la retrouve en négatif (‘ –N’ ‘).

Et on a séparé les closes par des ‘ ; ’.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| séparations  Une close   |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | b | b | …. | b | ; | b | b | …. | b | ; | ……… | ; | b | b | …. | b |   Avec b : 1 ou 0. |

**FIGURE 5 :** " un tableau d’un individu avec ses variable en binaire ".

1. **Résultat de l’individu :**

On pond l’individu, et pour chaque close, on exécute le ‘ OU Logique‘ entre ses variables binaires. par exemple :

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| trois closes :     |  |  |  |  | | --- | --- | --- | --- | | 0 | 1 | 0 | 0 |  |  |  |  |  | | --- | --- | --- | --- | | 1 | 1 | 0 | 0 |  |  |  |  |  | | --- | --- | --- | --- | | 0 | 0 | 0 | 0 |   🡪 1 U 1 U 0 U 0   0 U 1 U 0 U 0 0 U 0 U 0 U 0    🡪 1 1 0     |  |  |  | | --- | --- | --- | | 1 | 1 | 0 |   Tableau de résultat 🡪 |

**FIGURE 6 :** " Tableau de résultat d’un individu ".

On obtient à la fin un tableau de résultat de taille égale au nombre des closes.

Et on le classe avec les autres résultats selon son fitness (**L’étape ‘ c ’**).

1. **Evaluer les solutions (fitness) et les trier :**

On calcule La fitness de l’individu de la manière suivante : () ainsi un individu ayant une fitness de 1.00 sera un individu solution du problème.

On obtient à la fin un tableau de résultat de taille égale au nombre des closes.

Et on le classe avec les autres résultats tous classés selon leurs fitness dans un grand tableau, s’il a sa place, et on supprime celui qui a le plu petit fitness.

1. **Initialiser population :**

On a crée une population de taille spécifier en paramètre d’entré.

Et la taille de chaque individu est égale au nombre de variable de benchmark.  
 Chaque variable d’un individu prend une norme binaire de façon aléatoire (avec la fonction Radom).

1. **Evaluer les solutions (fitness) la population et les trier dans le grand tableau de résultats (étape ‘ c ’).**
2. **Appliquer le croisement :**

Lors du croisement on a utilisé manières de croiser les individus appelé "un point de croisement" :

On prend au hasard deux tableaux de solution parents (de notre grand tableau des solutions).

On sélectionne un point au hasard qui soit autre que le 1er ou le N-ème point de nos deux tableaux parents.

Et on les croise à ce point, comme cela :

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Point de croisement  **Parent 1 :**   |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 |   **Résultat (fils) :**     |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 |   **Parent 2 :**   |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | |

**FIGURE 7 :** " Croisement sur un point ".

Le nombre de fois qu’on exécute cette opération (le croisement) varie selon le tau de croisement et la talle de population entré en paramètre. On calcule ce nombre comme ca :

*NbCroisement* = ( ).

1. **Evaluer les solutions (fitness) des résultats de la mutation et les trier dans le grand tableau de résultats (étape ‘ c ’).**
2. **Appliquer la mutation :**

Lors de la mutation on a suivie le principe de changement de gènes :

On prend au hasard un tableau de solution parent (de notre grand tableau des solutions).

On sélectionne un point au hasard de notre tableau parents.

Et on les croise à ce point, comme cela :

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Point de mutation  **Parent :**   |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 1 |   **Résultat (fils) :**     |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | |

**FIGURE 8 :** " Mutation avec un point ".

Le nombre de fois qu’on exécute cette opération (la mutation) varie selon le tau de mutation et la talle de population entré en paramètre. On calcule ce nombre comme ca :

*NbMutation* = ( ).

1. **Evaluer les solutions (fitness) des résultats du croisement et les trier dans le grand tableau de résultats (étape ‘ c’).**

**.Explication du déroulement de l’AG:**

|  |
| --- |
| **0000000000000000000.png** |

**FIGURE 9 :** " schéma qui explique le déroulement de l’algorithme génétique ".

1. Evaluation et expérimentation

Les tests sont faits en deux parties selon les résultats obtenues de chaque algorithme :

* **Tests avec l’algorithme implémentant la procédure DPLL sur différant benchmarks qu’il les a satisfaites:**

|  |  |
| --- | --- |
| sstttttttt.png  **Tableau 2 :** " résultats en teste DPLL en ms et en minutes ". | TEST.png |

**FIGURE 10 :** " résultats des tests de l’algorithme DPLL sur des Benchmarks ".

**Constatation :**

.Plus le nombre de close et de variable augmente plus le DPLL prend plus de temps à satisfaire les closes du benchmark.

.après avoir teste toutes les possibilités et ne pas pouvoir trouvé de solution le DPLL indique que le benchmark est non satisfiable.

**Résultats :**

.Le temps de résolution varie avec la talle du benchmark (nombre de ses variables et de ses close).

.Le DPLL peut trouver de benchmark qui est non satisfiable.

.le DPLL donne un résultat stricte soit il ya une solution soit la solution n’existe pas.

* **Tests avec l’Algorithme Génétique sur différant benchmarks qu’il les a satisfaites:**

ON a effectué 5 tests :

1. Tester le taux de mutation :

Talle du benchmark :

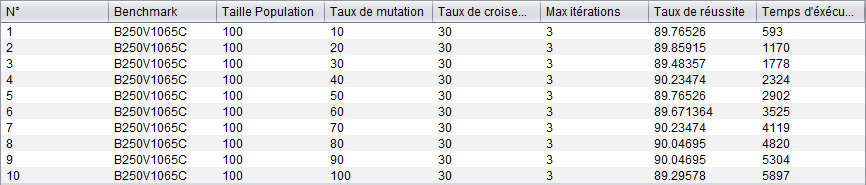
Nombre de variables : 250

Nombre de closes : 1065

Paramètre fixé :

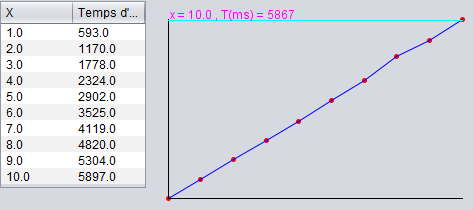
Talle population : 100

Taux de croisement : 30

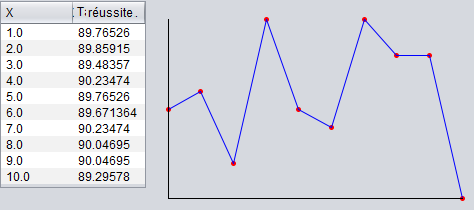
Max itération : 3

On teste le taux de mutation de 10 à 100 par pas de 10 :

**Tableau 3 :** " teste Taux de mutation ".



**FIGURE 11 :** " Taux de mutation en mesure de temps d’exécution ".



**FIGURE 12 :** " Taux de mutation en mesure de taux de fitness ".

**Constatation :**

.plus le taux de mutation augmente plus le temps d’exécution augmente.

.le pourcentage de la fitness (taux de réussite) augmente plus au mois.

1. Tester le taux de croisement :

Talle du benchmark :

Nombre de variables : 250

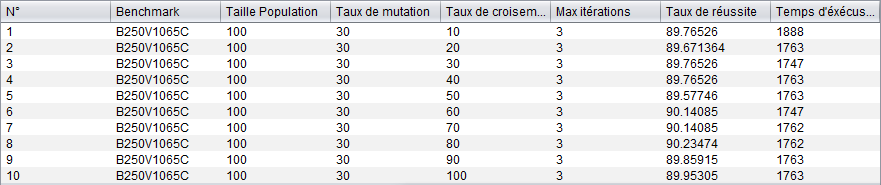
Nombre de closes : 1065

Paramètre fixé :

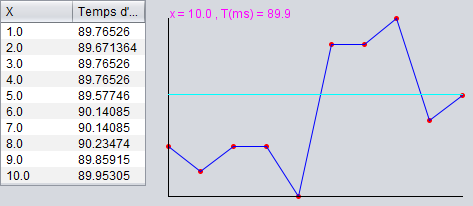
Talle population : 100

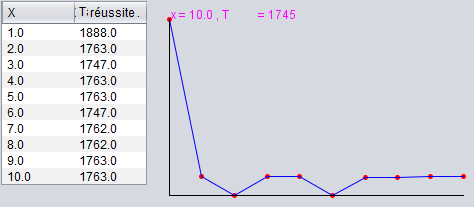
Taux de mutation : 30

Max itération : 3

 On teste le taux de croisement de 10 à 100 par pas de 10 :

**Tableau 4 :** " teste Taux de croisement ".

****

**FIGURE 13 :** " Taux de croisement en mesure de temps d’exécutions ".

**FIGURE 14 :** " Taux de croisement en mesure de taux de fitness ".

**Constatation :**

.plus le taux de croisement augmente plus le temps d’exécution augmente.

.le pourcentage de la fitness (taux de réussite) augmente plus au mois.

1. Tester le taux de croisement et la mutassions en même temps :

Talle du benchmark :

Nombre de variables : 250

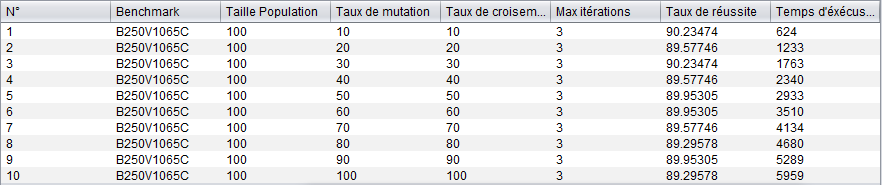
Nombre de closes : 1065

Paramètre fixé :

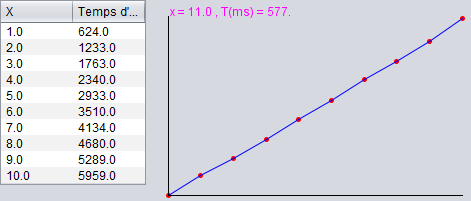
Talle population : 100

Max itération : 3

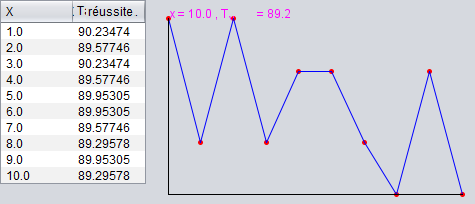
On teste le taux de croisement et le taux de mutation de 10 à 100 par pas de 10 :



**Tableau 5 :** " teste Taux de croisement et Taux de croisement ".

****

**FIGURE 15 :** " Taux de croisement et taux de mutation en mesure de temps d’exécutions ".

****

**FIGURE 16 :** " Taux de croisement et taux de mutation en mesure de taux de fitness ".

**.conclusions des 3 tests :**

.Le temps d’exécution augmente avec l’augmentation du taux de croisement et/ou de mutation.

. Les taux de croisement et celui de mutation améliore les résultats et augmente la fitness.

1. Tester de max itération :

Talle du benchmark :

Nombre de variables : 250

Nombre de closes : 1065

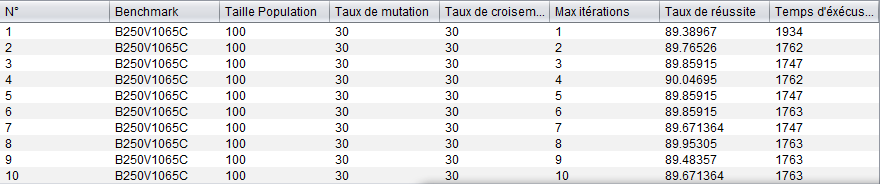
Paramètre fixé :

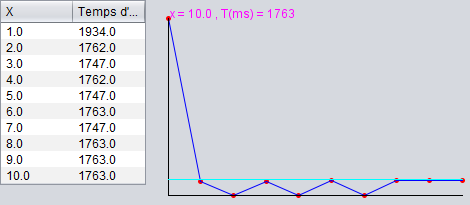
Talle population : 100

Taux de mutation : 30

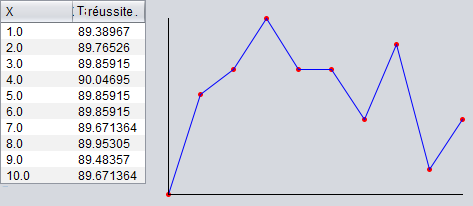
Taux de croisement : 30

On teste la max itération de 1 à 10 par pas de 1 :



** Tableau 6 :** " teste Taux de croisement et Taux de croisement ".

**FIGURE 17 :** " Max Itération mesure de temps d’exécutions ".



**FIGURE 18 :** " Max Itération en mesure de taux de fitness ".

**.conclusions :**

. Plus elle augmente plus la max itération est couteuse en matière de temps d’exécution.

.l’augmentation de la max itération améliore le taux de fitness.

1. Tester différents benchmark :

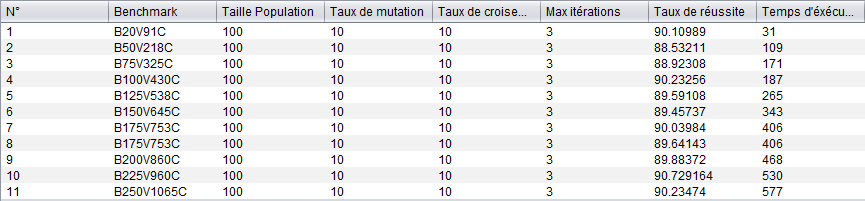
Paramètre fixé :

Talle population : 100

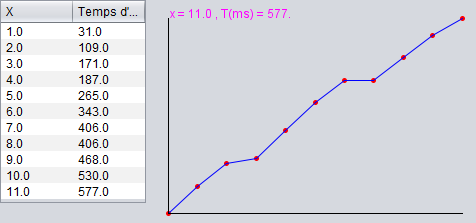
Taux de mutation : 30

Taux de croisement : 30

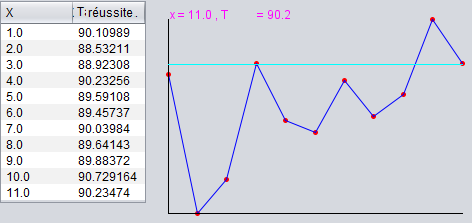
Mas itération : 3

 On teste différent benchmark de talle entre 20 et 250 variable :

**Tableau 7 :** " résultat de test sur défirent benchmarks ".



**FIGURE 19 :** " Max Itération mesure de temps d’exécutions ".

****

**FIGURE 20 :** " résultat tests de différent benchmarks en mesure de taux de fitness ".

**.conclussions :**

.plus la talle du benchmarks augmente (nombre des variables et nombre des closes), plus sa coute en temps d’exécution.

.on a un taux de résolution qui est pluton stable avec une moyenne de 89 % de taux de satisfaction des benchmarks.

**. Conclusion générale des tests :**

.plus on augmente les paramètres d’entré que ca soit le taux de croisement ou de mutation ou de la population initiale ou de la max itération plus c’est couteux en terme de temps d’exécution.

.Et plus on les augmente les paramètres d’entré (que ca soit le taux de croisement ou de mutation ou de la population initiale ou de la max itération) plus c’est rentable en terme de résolution et plus on a un meilleur fitness.

1. Outils et méthodes

**Le matériel utilisé pour ce projet est :**

* Processeur :  Intel(R) Core(TM) i3 CPU 540 @ 3.07GHz 3.06GHz
* Mémoire installée (RAM) : 4.00 Go (3.24 Go utilisable)
* Windows 7 Professionnel Système d’exploitation 64 bits
* Evaluation : 3.2 Indice de performance Windows

**La technique utilisée et logicielles :**

* Cette application est implémentée avec le langage JAVA sous l’environnement NetBeans8.0.2. + l’IDE.
* Lors de l’implémentation on a respecté l’ergonomie de l’IHM.
* Touts en utilisant des compétences en POO (Programmation Orienté Objet).
* Afin de Booster l’environnement Java, on a procédé comme tel :
* Il faut télécharger et installer la dernière version de Java Runtime Environment, en faisant attention qu'elle corresponde à notre configuration.
* Augmenter la priorité du processus:

- Ctrl+Alt+Suppr

- Gestionnaire des tâches 🡪 Onglet Processus

- Clic droit sur Javaw.exe 🡪 Définir Priorité 🡪 Haute

* Augmenter la quantité de mémoire allouée à Java:

- Panneau de configuration 🡪 Programmes 🡪 Java

- Le panneau de configuration de java s'ouvre alors

- Cliquez sur le 2e onglet en partant de la gauche ("Java") puis sur "Afficher"

- On ce retrouvez avec cette une fenêtre dans la colonne "Paramètres d'exécution", rajoutez la ligne suivante dans la boite de dialogue:

-Xms256m -Xmx1024m

- Cochez la case dans la colonne "Activé".

- On a alloué 2048 Mo max à java.

* Utilisation de tous les corps de notre processeur:

- Ctrl+Alt+Suppr

- Gestionnaire des tâches 🡪 Onglet Processus

- Clic droit sur Javaw.exe 🡪Définir l'affinité... 🡪 Vérifier que tous vos corps sont cochées.

1. Conclusion

Nous nous sommes intéressés dans cette étude aux problèmes de satisfiabilité booléenne ou encore plus connu par le problème de SAT, Et aux méthodes de résolution et d’optimisation de décisions. Notre objectif est en partie de nous focaliser sur la conception d'une approche hybridée avec une méta-heuristique afin de booster les performances en matière de temps de réponse. En autre partie, contourner la complexité du problème et aller vers plus de stabilité et plus de simplicité. Tout en faisans Une étude expérimentale pour étayer la performance des approches proposées.

Afin de réaliser notre but, nous avons organisé nos recherches en deux phases. Dans la première phase, nous nous sommes concentrés sur l’établissement d’un état de l’art sur les problèmes d’optimisation de satisfiabilité et les différentes méthodes de résolution proposées dans la littérature. Nous avons essayé de récolter des notions de bases en optimisation, comprendre le rôle des notions d’optimisations en appliquons l’opération de croisement en optimisation génétique, le principe de changement de gènes dans l’algorithme de mutation et tout en utilisant le maximum d’itération. Et d’étudié les principes de l’algorithme DPLL.

En deuxième phase Nous avons implémenté deux algorithmes et expérimenter des tests sur eux. L’Algorithme Génétique hybridé, basé sur le croisement, la mutation et l’étatisation de maximum d’itération. Et implémenté comme tout le bon résolvons de problème de SAT de nos jours, un autre algorithme basé sur la procédure DPLL.

Et grasse nos deux algorithme implémenté on a pue obtenir des résulta sur les tests de satisfiabilité sur des grand benchmarks de closes. Avec notre algorithme DPLL on peu préciser si un benchmark peut être satisfiable ou non. Cela avec un temps de traitement assez raisonnable. Et avec notre autre algorithme on a pue réaliser un double objectif: premièrement, proposer un algorithme original permettant la résolution de problèmes d’optimisation de satisfiabilité des clauses, avec des résultats optimaux dans une durée optimal. Deuxièmement, donnée à l’utilisateur plus de perception et d’interaction avec un nombre de paramètres riche, précis et essentiels pour le fonctionnement de l’algorithme (taille de la population initiale, taux de mutation, taux de croisement et le max itération).

Pour que notre travaille soit expérimenté en tout simplicité on a fais en sorte qu’il donne des graphes des défirent résultat pour les comparer avec d’autres travaux.

En final, Les deux approches sont fêtent Pour faire face à une grande gamme de problèmes pouvant être modélisés par des structures binaires.

1. Références

  Pour un aperçu de ce qui se fait dans le monde de la recherche en (2015), précise différents types d’algorithmes pour résoudre des problèmes SAT :

**« Le livre 1 :** Problème SAT : progrès et défis »

Analyse et explication pour comment implémenter diverse méthodes pour résoudre SAT

« **Le livre 2 :** Handbook of Satisfiability »

Amélioration des codages, automatisation de la traduction et étude comparative

**« Planification SAT** »

1. Annexes

<http://www.satlive.org/>

<http://www.cs.ru.nl/~elenam/fsat.pdf>

<http://minisat.se/downloads/MiniSat.pdf>

<http://minisat.se/downloads/MiniSat.pdf>.