République Algérienne Démocratique et Populaire
Ministère de L‟enseignement Supérieur et de la Recherche Scientifique
Université Mentouri de Constantine
Faculté des Sciences de l‟Ingénieur
Département d‟Informatique

N° d'ordre :

Série :

Utilisation des Approches d'Optimisation Combinatoire pour la Vérification des Applications Temps Réel

par

Abdesslem LAYEB

Thèse présentée à la faculté des Sciences de l‟Ingénieur
en vue de l‟obtention du grade de Docteur en science
en Informatique

a été évaluée par un jury composé des personnes suivantes :

Dr. Mohamed Kheireddine KHOLLADI. MC à l'université Mentouri de Constantine	**Président**
Dr. Djamel Eddine SAIDOUNI. MC à l'université Mentouri de Constantine	**Directeur de recherche**
Dr. Mohamed Tahar KIMOUR. MC à l'université Badji Mokhtar d'Annaba	**Examinateur**
Dr. Mohamed Tarek KHADIR. MC à l'université Badji Mokhtar d'Annaba	**Examinateur**
Dr. Salim CHIKHI. MC à l'université Mentouri de Constantine	**Examinateur**

27/05/2010

A la mémoire de mon père

A la mémoire de mon frère

Remerciements

D'abord, Je remercie mon Dieu le tout puissant de m'avoir aidé, et de m'avoir donné le courage et l'honneur d'être un Docteur en Informatique. J'espère bien être à la hauteur de ce titre.

Je tiens, avant tout, à exprimer ma profonde gratitude à Monsieur Djamel Eddine Saidouni pour m'avoir accueilli, encadré et soutenu tout au long de ces quatre années de thèse et pour ses précieux conseils et ses encouragements continuels. Je lui suis particulièrement reconnaissant de m'avoir laissé une grande liberté scientifique, ce qui m'a offert la possibilité d'explorer des pistes de recherche en complet accord avec mes centres d'intérêt.

Je remercie également tous les Professeurs membres du jury. Qu'ils veuillent trouver ici toute ma reconnaissance pour la peine qu'ils ont prise afin d'examiner et d'évaluer ce travail.

Je remercie tout le personnel des laboratoires LIRE et MISC sans oublier tous le personnel du département d'informatique de l'université Mentouri de Constantine.

Merci à tous les membres de ma famille la grande et la petite qui, par leur support et encouragement, m'ont permis de m'investir entièrement dans mes études.

Je tiens à remercier toutes les personnes qui, directement ou indirectement ont contribué à la réalisation de ce mémoire.

Abdesslem LAYEB

Abstract

In this doctoral thesis, we are interested in two big combinatory optimization problems encountered within the formal methods when we check applications.

The first problem is the variable reordering for Binary Decision Diagram (BDD). The BDD is used to represent in symbolic manner a set of states. It‟s largely used in the field of formal checking. The variable ordering is a very important step in the BDD optimization process. A good order of variables will reduce considerably the size of a BDD. Unfortunately, the search for the best variables ordering has been showed NP-difficult.

The second kind of problems handled in this thesis is the satisfiability problems (SAT and MAX-SAT). The Boolean satisfiability problem is defined as the task of determining the satisfiability of a given Boolean formula by looking for the variable assignment that makes this formula evaluating to true. The Maximum Satisfiability (Max-SAT) problem is a variant of SAT problem that aims to find the variable assignment maximizing the number of satisfied clauses. It has been demonstrated that the Max-SAT problem is NP-complete. This complex problem has several applications in different domains such as model checking, graph coloring and task planning, etc.

The use of heuristics like genetic algorithms and local search is essential to find acceptable solutions within reasonable times. Within this perspective, we are interested in applying hybrid evolutionary algorithms to solve these problems.

The purpose of this work is multiple. In a first place, it is a question of exploring the field of combinatorial optimization and its basis in order to well manage the various concepts and algorithms used in this thesis. In a second place, it is a question of prospecting the field of binary decision diagrams underlining the reordering problem involved in the combinatorial explosion of the BDD representation. In a third place, it is a question of undertaking an investigatory study aiming at synthesizing work relating to the combinatorial explosion in the field of boolean satisfiability problems. Finally, in fourth place, it is question of proposing new approaches to apprehend these problems while being based on the combinatorial optimization methods like the genetic algorithms, the algorithms inspired by quantum, the local search, etc.

Key words: binary decision diagram, SAT, MAX-SAT, optimization, quantum evolutionary algorithms, local research, exact approaches, hybrid approaches.

ملخص

لقد اهتممنا في هذه المذكرة بمسألتين مهمتين في مجال التحقق الشكلي للبرامج المعقدة. المسألة الأولى هي ترتيب المتغيرات في بيانات التحكم الثنائية. تستعمل هذه الأخيرة بكثرة في مجال التحقق الشكلي و خاصة في مجال التحقق باستعمال تقنية تفقد النماذج. غير أن حجم البيانات الثنائية يتعلق بصفة كبيرة بترتيب المتغيرات للدالة المراد تمثيلها. لسوء الحظ، إيجاد الترتيب الملائم هو مسألة في غاية الصعوبة. المسألة الثانية التي تطرقنا إليها في هذه المذكرة هي إيجاد حلول لإشباعية المتغيرات المنطقية في قيود منطقية تعتبر هذه المسألة بمثابة العمود الفقري لحل العديد من المشاكل في مختلف الميادين كمجال التحقق الشكلي للبرامج و الدارات الكهربائية. على غرار المسألة السابقة فإن هذه الأخيرة تتميز هذه المسألة بصعوبة قوية وضخمة. من أجل هذه المسائل، استعمال طرق الحل الجديدة تعتبر مخرجا لحل هذه المسائل المستعصية. إن الغاية من هذا البحث تكمن في حصر أهداف أربعة أهداف. أولاها القيام ببحث مكتبي لاكتشاف طرائق تحسين الحلول. الهدف الثاني يكمن في البحث عن مسائل التحسين في مجال التحقق الشكلي. أما الهدف الثالث فهو دراسة الطرق المستعملة حاليا في حل هذه المسائل. رابعا اقتراح جملة من الحلول لهذه المسائل باستعمال طرق جديدة لتحسين الحلول.

كلمات مفتاحية: التحقق الشكلي، بيانات التحكم الثنائية، حلول لإشباعية المتغيرات المنطقية في قيود منطقية، طرق ذكية متطورة.

Résumé

Nous nous intéressons dans cette thèse de doctorat à deux importants problèmes d‟optimisation combinatoire rencontrés dans le cadre de la vérification formelle des applications complexes.

Le premier problème est celui de l‟ordonnancement des variables dans un diagramme de décision binaire (BDD). En effet, les BDDs sont largement utilisés dans le domaine de la vérification formelle et notamment dans la vérification par évaluation de modèle (model checking). Cependant, la taille des BDDs dépend étroitement de l‟ordre de variables choisi lors de la construction du BDD représentant une fonction. De ce fait, il est important de trouver un bon ordre de variables qui minimise le nombre de nœuds dans un BDD. Malheureusement, cette tâche n‟est pas facile vu qu‟on a un nombre exponentiel d‟ordres possibles.

Le deuxième problème concerne la résolution des problèmes de satisfiabilité SAT et MAXSAT. Le problème de satisfiabilité de formules propositionnelles (SAT) est depuis un grand nombre d‟années un problème central dans bon nombre de domaines tel que la vérification et la spécification de circuits électroniques et les techniques de type Model Checking. Malheureusement le problème SAT se caractérise par une grande complexité temporelle.

Pour cela, l‟utilisation des heuristiques est indispensable pour trouver des solutions acceptables et dans des délais raisonnables. L‟exploitation des caractéristiques des méthodes d‟optimisation combinatoire constitue une bonne alternative pour traiter les problèmes sous-jacents.

L'objectif de ce travail est multiple. Dans un premier lieu, il s'agit d'explorer le domaine de l‟optimisation combinatoire et ses fondements afin de bien maîtriser les différents concepts et algorithmes utilisés. Dans un second lieu, il s'agit de prospecter le domaine de la vérification formelle en soulignant les problèmes liés à l‟explosion combinatoire du graphe d‟état et la vérification des formules logiques. Dans un troisième lieu, il s'agit de mener une étude investigatrice visant à synthétiser les travaux relatifs au problème d‟ordonnancement des variables d‟un BDD et les problèmes de satisfiabilité booléenne. En fin, en quatrième lieu, il est question de proposer de nouvelles approches pour appréhender ces problèmes en se basant sur les méthodes d‟optimisation combinatoire comme les algorithmes génétiques, les algorithmes inspirés du quantique, la recherche locale, etc.

Mots-clés : diagramme de décision binaire, SAT, MAX-SAT, optimisation, algorithmes évolutionnaires quantiques, recherche locale, approches exactes, approches hybrides.

Table des matières

Partie 2: Contributions

Listes des figures

Listes des tables

Liste des algorithmes

Introduction générale

Les systèmes informatiques sont de plus en plus répandus dans plusieurs systèmes en allons des systèmes simples comme les téléphones mobiles, les consoles de jeux, etc., aux systèmes plus complexes et plus critiques comme les avions, les centrales nucléaires, etc. Pour ces raisons, la vérification de ce genre de systèmes s'est largement développée ces dernières années. En effet, La vérification commence à occuper une place importante dans le processus de conception du matériel et des logiciels, plus spécialement dans la fabrication des systèmes informatiques. Les entreprises veulent éviter de construire des systèmes ne respectant pas leur spécification et surtout d'avoir à faire des maintenances et adaptations qui sont souvent très couteuses et qui entachent sa réputation. Par ailleurs, s'assurer qu'un circuit logique ou qu'un programme fonctionne correctement est devenu de plus en plus pénible au fur et à mesure que les systèmes se sont complexifiés. D'une manière simple, La validation fonctionnelle d'un système matériel consiste à vérifier le système vis-à-vis de son fonctionnement attendu. La vérification se fait soit directement sur le système ou bien sur un modèle représentatif de celui-ci.

Il existe plusieurs approches de vérification qu'on peut classer en deux classes : les approches informelles qui consistent à construire puis tester le système, est souvent un passage obligé (comme en attestent les nombreux essais de l'Airbus A380), mais elle n'est généralement pas exhaustive. Cependant, les approches formelles basées sur des raisonnements mathématiques, sont donc souvent préférées au moment de la conception des systèmes. Parmi les techniques les plus répandues dans le domaine de la vérification formelle on distingue la vérification par évaluation de modèle (model-checking en anglais) qui est une technique entièrement automatique basée sur les modèles. Le model-checking consiste à modéliser le comportement du système à vérifier, à énoncer formellement les propriétés de correction attendue puis à appliquer des algorithmes pour décider si le modèle satisfait la propriété.

L'un des problèmes rencontrés dans le domaine de la vérification formelle est l'explosion combinatoire. Par exemple dans le modele-checking, le nombre d'états dans les graphes des transitions peut atteindre des niveaux prohibitifs ce qui rend leur manipulation difficile, voire impossible. Pour cela, on utilise des méthodes de compression afin de réduire la taille des graphes d'états. La compression se fait en utilisant des structures de données afin de représenter de façon concise des ensembles d'états. Dans ce cas, Les opérations se font alors sur des ensembles d'états plutôt que sur des états explicites. Cette technique est souvent qualifiée de symbolique.

Parmi les représentations symboliques les plus connues, on trouve la représentation par les diagrammes de décision binaire (BDD, de Binary Decision Diagrams). Les BDDs sont des

structures de données utilisées pour représenter des fonctions booléennes. Les BDDs sont largement utilisés dans plusieurs domaines parce qu‟ils offrent une représentation canonique et une manipulation facile. Cette méthode de représentation a été utilisée avec succès dans la vérification de systèmes déterministes et probabilistes à espace d‟états discret et même dans la vérification des systèmes temps réel. En effet, les approches de vérification basées sur les BDDs permettent d'un autre côté, de minimiser l'impact de l'explosion du nombre d'états du système grâce à une représentation des états et des comportements du système particulièrement concise. Malheureusement, la taille d‟un BDD dépend de l‟ordre des variables utilisées lors de la génération, c'est pourquoi on utilise souvent le terme OBDD (pour ordered). De ce fait, il est important de trouver un bon ordre de variables qui minimise le nombre de nœuds dans un BDD. Malheureusement, cette tâche n‟est pas facile vu qu‟on a un nombre exponentiel d‟ordres possibles. En effet, Le problème de l‟ordre de variables a été démontré NP-difficile. Pour cela, plusieurs méthodes ont été proposées pour trouver le bon ordre de variables d‟un BDD et qu‟on peut classer en deux classes. La première classe tente d‟extraire le bon ordre en inspectant les circuits logiques. Par ailleurs, la deuxième classe est basée sur l‟optimisation dynamique d‟un ordre donné.

Nous montrons également dans la première partie de cette thèse, que les problèmes de diagnostic de certains systèmes peuvent être traduits en problèmes de satisfiabilité propositionnelle (SAT) de formules booléennes sous forme normale conjonctive (CNF). Les résultats montrent que les algorithmes SAT permettent de résoudre des problèmes que les algorithmes traditionnels de diagnostic ne peuvent pas traiter. Ce type de méthode de vérification est appelée *Bounded Model Checking* (BMC), qui utilise un solveur propositionnel SAT plutôt que des techniques de manipulation des BDDs.

Le problème de satisfiabilité de formules propositionnelles sous forme normale conjonctive (SAT) est depuis un grand nombre d‟années un problème central dans un bon nombre de domaines dont la vérification de circuits électroniques, les techniques de type Model Checking, la planification de tâches et la cryptographie. Le problème de satisfiabilité (SAT) consiste à trouver une affectation booléenne validant une formule en logique propositionnelle. Ce problème possède plusieurs variantes dont le problème MAX-SAT, auquel nous nous intéressons plus particulièrement dans cette thèse. En effet, ce problème peut être vu comme une formulation alternative du problème SAT qui consiste à trouver une affectation maximisant le nombre de clauses satisfaites dans une formule booléenne.

La résolution de ce problème est généralement considérée comme un problème de décision. Malheureusement, le problème de satisfiabilité booléenne a été démontré NP-complet. Etant donné l'importance de ce problème, de nombreuses méthodes de résolution ont été proposées et qui fournissent des résultats satisfaisants dans la pratique. Elles peuvent se départager en deux grandes classes, les méthodes complètes dites exactes et les méthodes incomplètes dites approchées. Les méthodes complètes permettent d‟effectuer une recherche

exhaustive exploitant complètement l‟espace de recherche. Malheureusement, ce type de méthode est impraticable dans le cas de grandes instances du fait de la complexité exponentielle de ces méthodes, c‟est le cas notamment des méthodes DP, DPLL et de variantes de méthodes plus génériques comme Branch & Bound. Cependant, les méthodes incomplètes n‟explorent que certaines zones de l‟espace de recherche et s‟avèrent particulièrement efficaces pour des instances de grandes tailles bien qu‟elles ne donnent pas des solutions exactes. La plupart des méthodes incomplètes sont basées sur la recherche locale.

Comme on peut remarquer que tous les problèmes traités dans cette thèse sont difficiles à résoudre et se caractérisent par une grande complexité aussi bien temporelle que spatiale. Ce type de problèmes appartient à ce qu‟on appelle les problèmes d‟optimisation combinatoire. Cette dernière est une branche de l'optimisation en mathématiques appliquées et en informatique, également liée à la recherche opérationnelle, l'algorithmique et la théorie de la complexité. Résoudre un problème d'optimisation consiste à trouver la ou les meilleures solutions vérifiant un ensemble de contraintes et d'objectifs définis par l'utilisateur. Pour déterminer si une solution est meilleure qu'une autre, il est nécessaire que le problème introduise un critère de comparaison. Ainsi, la meilleure solution, appelée aussi solution optimale, est la solution ayant obtenu la meilleure valuation au regard du critère défini. Il existe plusieurs méthodes de résolution pour chaque problème d‟optimisation combinatoire qu‟on peut classer en deux grandes classes : les méthodes de résolution exactes et les méthodes de résolution approchées.

Le principe essentiel d'une méthode exacte consiste généralement à énumérer, souvent de manière implicite l'ensemble des solutions de l'espace de recherche. Parmi les méthodes exactes, on trouve la plupart des méthodes traditionnelles telles que la programmation dynamique, la programmation linéaire et les techniques de séparation et évaluation progressive ou les algorithmes avec retour arrière. Les méthodes exactes ont permis de trouver des solutions optimales pour des problèmes de taille raisonnable. Malgré les progrès réalisés (notamment en matière de la programmation linéaire en nombres entiers), comme le temps de calcul nécessaire pour trouver une solution risque d'augmenter exponentiellement avec la taille du problème, les méthodes exactes rencontrent généralement des difficultés face aux applications de taille importante. D‟un autre côté, les méthodes approchées constituent une alternative très intéressante pour traiter les problèmes d'optimisation plus complexes si l'optimalité n'est pas primordiale. Depuis une dizaine d'années, des progrès importants ont été réalisés avec l‟apparition d‟une nouvelle génération de méthodes approchées puissantes et générales dédiées aux problèmes d‟optimisation difficile, souvent appelées métaheuristiques. Les métaheuristiques sont, on générale présentées sous la forme de concept d‟inspiration, et elles reprennent des idées que l‟on retrouve parfois dans la vie courante. Ces méthodes approchées peuvent se classer en différentes catégories :

– Constructives (algorithmes gloutons),

– Recherche locale (algorithmes de descente, recherche à grand voisinage, . . .),

– Métaheuristiques (recuit simulé, recherche Tabou,. . .),

– Évolutionnaires (algorithmes génétiques, algorithmes d‟optimisation par colonies de fourmis, algorithmes mémétiques,. . .).

Parmi les métaheuristiques les plus récentes, on peut citer les Algorithmes Quantiques Evolutionnaires (AQEs). Les AQEs constituent un nouveau champ de recherche qui combine les algorithmes évolutionnaires classiques et les principes de l‟informatique quantique. Ce dernier est un domaine récent en informatique qui essaie de donner des solutions à des problèmes non encore résolus par l‟informatique classique. En effet, l‟informatique quantique offre des capacités de traitement et de stockage exponentiels grâce à des principes de la mécanique quantique telles que la superposition, l‟enchevêtrement, l‟interférence. Cependant, l‟utilisation de ce nouveau paradigme est dépendante de la réalisation pratique des machines quantiques, mais cela n‟a pas empêché les chercheurs de combiner les algorithmes classiques tels que les algorithmes génétiques et les principes de l‟informatique quantique afin de tirer profit des capacités de ce dernier. Contrairement aux algorithmes quantiques purs, les algorithmes inspirés du quantique ne nécessitent pas la présence des machines quantiques. Les algorithmes inspirés du quantique ont démontré leur efficacité dans plusieurs problèmes d‟optimisation combinatoire.

Les différents travaux réalisés au cours de cette thèse ont donné lieu à la conception de plusieurs algorithmes hybrides pour résoudre les problèmes sous-jacents. Nous nous sommes intéressés aux possibilités d‟hybridation entre les méthodes d‟optimisation globales et locales, ainsi qu‟entre les méthodes de résolution exactes et approchées afin de pouvoir tirer avantage de chacune de ces approches. Le travail de cette thèse est composé en deux parties essentielles. La première partie présente tout d‟abord une introduction du contexte de cette thèse : les méthodes d‟optimisation combinatoire, le problème d‟ordonnancement de variables dans les diagrammes de décision binaire et le problème de satisfiabilité booléenne maximale dans la logique propositionnelle. D‟un autre côté, la deuxième partie renferme les travaux réalisés au cours de cette thèse pour la résolution des problèmes sous-jacents. Pour contribution, une validation expérimentale a été réalisée sur plusieurs benchmarks. Les résultats obtenus montrent l‟intérêt des méthodes proposées et laissent entrevoir les nombreuses perspectives ouvertes par ce type d‟hybridation.

Organisation

Cette thèse est composée de six chapitres divisés en deux parties :

La première partie présente le contexte de ce travail, elle est décomposée en quatre chapitres. Le premier chapitre présente les notions de base de l'optimisation combinatoire ainsi que les différentes méthodes utilisées pour la résolution des problèmes d'optimisation combinatoire. Dans un premier temps, on décrit le fonctionnement des principales méthodes exactes ainsi que les métaheuristiques les plus connues, tels que les algorithmes génétiques, la recherche locale, le recuit simulé, la recherche tabou, etc. Le deuxième chapitre est dédié à l'informatique quantique. Ce chapitre présente les fondements de l'informatique quantique, ainsi que les algorithmes inspirés du quantique. Nous présentons également un algorithme inspiré du quantique basé sur un algorithme de recherche dispersé. Le troisième chapitre est consacré aux diagrammes de décision binaire BDD (Binary Decision Diagrams), tout en commençant par les notions de base des BDDs, la réduction d'un BDD, l'utilité des BDDs dans la vérification des systèmes, etc. Ensuite, le problème d'ordre de variables d'un BDD est présenté ainsi que les différentes méthodes utilisées pour sa résolution. Dans le quatrième chapitre, nous présentons le problème de satisfiabilité booléenne SAT, qui va servir, notamment dans le chapitre 6, comme exemple d'application pour mettre en évidence l'utilité pratique des contributions proposées dans cette thèse. Après un rappel des éléments de base de la logique propositionnelle ainsi que certains problèmes qui lui sont associés tels que le problème MAX-SAT, nous présentons les différentes méthodes de résolutions existantes (méthodes exactes et méthodes approchées, hybrides).

La deuxième partie présente nos contributions, elle est décomposée en deux derniers chapitres de la thèse. Dans le cinquième chapitre, nous commençons d'abord par une introduction motivant le travail proposé. Puis, nous présentons nos contributions pour résoudre le problème d'ordonnancement dans les BDDs en commençant par la formalisation du problème, puis on présente les différentes approches développées. Ensuite, nous présentons le design expérimental relatif aux essais numériques. Enfin, nous présentons et discutons les résultats numériques obtenus. Quant au dernier chapitre, il est consacré à la présentation des approches proposées pour résoudre le problème de satisfiabilité maximale MAX SAT. On présente dans ce chapitre les principes, les méthodologies de résolution sous-jacentes ainsi que les résultats de l'évaluation de leurs performances.

A la fin de ce mémoire, nous donnerons une conclusion qui fait le bilan des contributions proposées et qui fait également l'analyse de ces contributions pour faire une ouverture sur de nouvelles directions de recherche à court terme ainsi qu'à moyen et long terme.

CHAPITRE 1

Introduction à l'optimisation combinatoire

"Imagination is more important than knowledge".

—Albert Einstein—

L'optimisation combinatoire est un outil indispensable combinant diverses techniques des mathématiques discrètes et de l'informatique afin de résoudre des problèmes souvent complexes de la vie réelle.

Dans ce premier chapitre, nous présentons d'abord de manière concise les notions de base de cette discipline et sa relation avec la théorie de la complexité. Ensuite, nous décrivons sommairement certaines méthodes, parmi les plus représentatives, développées pour résoudre les problèmes d'optimisation combinatoire. Ce chapitre a pour but de situer le contexte de ce travail de thèse et non de faire une synthèse exhaustive sur le sujet.

Sommaire

1.1. Introduction

L'optimisation combinatoire est un outil indispensable combinant diverses techniques des mathématiques discrètes et de l'informatique afin de résoudre des problèmes de la vie réelle [Baeck et al., 2000]. En effet, l'optimisation combinatoire, couvre aussi bien des domaines de l'ingénieur que les domaines de la recherche, et englobe un large éventail de techniques et fait toujours l'objet de recherches intensives. D'une manière simple, résoudre un problème d'optimisation combinatoire consiste à trouver l'optimum d'une fonction, parmi un nombre fini de choix, souvent très grand [Baeck et al., 1997]. Il s'agit, en général, de maximiser (problème de maximisation) ou de minimiser (problème de minimisation) une fonction objectif sous certaines contraintes. Le but est de trouver une solution optimale dans un temps d'exécution raisonnable. Néanmoins, ce but est loin d'être concrétisé pour plusieurs problèmes vu leurs complexités grandissantes. La théorie de la NP-complétude [Garey et al., 1979] a permis de classer les problèmes d'optimisation selon leurs complexités et elle fournit des informations pertinentes sur le genre de méthodes à choisir en fonction de la difficulté intrinsèque des problèmes. Les problèmes d'optimisation peuvent être départagés en deux classes, lorsqu'une solution est associée à une seule valeur, on parle de problèmes mono-objectifs, et lorsqu'elle est associée à plusieurs valeurs, on parle de problèmes multi-objectifs (ou multi-critères) [Fonseca et al., 1993]. Il faut noter que, l'optimisation d'un problème multi-obectif est souvent plus difficile que l'optimisation des problèmes mono-obectifs. En effet, l'optimisation multi-objectif permet de modéliser des problèmes réels faisant concourir de nombreux critères (souvent conflictuels) et contraintes. Dans ce contexte, la solution optimale recherchée n'est plus un simple point, mais un ensemble de bons compromis satisfaisant toutes les contraintes. Bien que les problèmes d'optimisation combinatoire soient souvent faciles à définir, ils sont généralement pénibles à résoudre [Papadimitriou, 1982]. En effet, la plupart de ces problèmes appartiennent à la classe des problèmes NP-difficiles et ne possèdent pas encore de solutions algorithmiques efficaces et acceptables pour toutes les données [Dréo et al. 2006]. Outre la classification selon le nombre de critère à optimiser, les méthodes de l'optimisation combinatoire peuvent être classées aussi en méthodes heuristiques et méthodes exactes (figure 1.1).

Les algorithmes exacts sont utilisés pour trouver au moins une solution optimale d'un problème [Palpant, 2005]. Les algorithmes exacts les plus réussis dans la littérature appartiennent aux paradigmes de la programmation dynamique, de la programmation linéaire en nombres entiers, ou des méthodes de recherche arborescente (Branch & Bound). Généralement, pour un problème donné d'optimisation discrète, ces algorithmes de résolution procèdent par énumération de toutes les solutions possibles afin de trouver la meilleure solution. Lorsque les ampleurs du problème deviennent importantes, une énumération explicite s'avère irréalisable, pour cela, on a recours aux procédures classiques par séparation et évaluation ("Branch and Bound") [Brucker 1998]. Ces algorithmes réalisent une

construction partielle d'un arbre de recherche des solutions en essayant de trouver une solution exacte pour un problème donné. Bien que les méthodes exactes trouvent des solutions optimales, leur grand inconvénient est l'explosion combinatoire, ce qui rend leur utilisation pratique difficile.

D'un autre côté, une méthode heuristique ou approchée est une méthode d'optimisation qui a pour but de trouver une solution réalisable de la fonction objectif en un temps raisonnable, mais sans garantie d'optimalité. L'avantage principale de ces méthodes est qu'elles peuvent s'appliquer à n'importe quelle classe de problèmes, faciles ou très difficiles, bien ou mal formulés, avec ou sans contrainte. En plus, elles ne nécessitent pas une spécification mathématique du problème. D'un autre côté, les algorithmes d'optimisation tels que les algorithmes de recuit simulé, les algorithmes tabous et les algorithmes génétiques ont démontré leurs robustesses et efficacités face à plusieurs problèmes d'optimisation combinatoires [Reeves, 1995].

Il faut noter qu'il n'existe pas une méthode sésame qui peut résoudre tous les problèmes, parfois c'est intéressant de faire hybrider plusieurs algorithmes de résolution au sein d'un seul algorithme afin de résoudre des problèmes très compliqués.

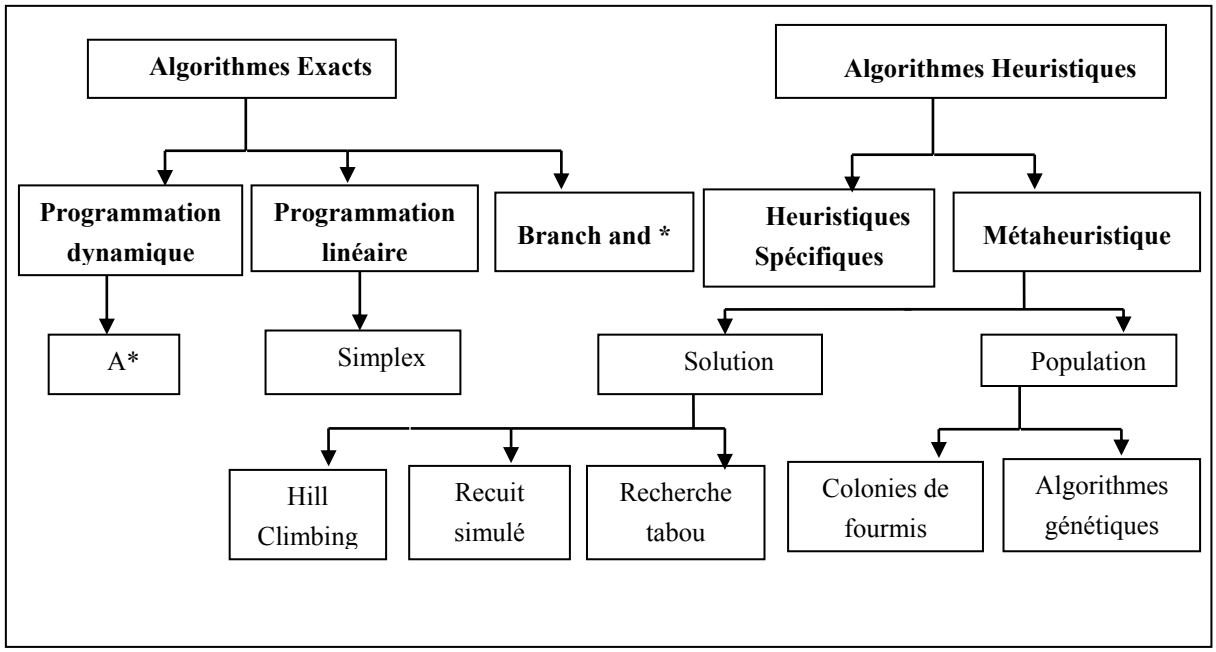


Figure 1.1 – Taxonomie des méthodes d'optimisation combinatoire.

1.2. La théorie de la complexité

La théorie de la complexité consiste à estimer la difficulté ou la complexité d'une solution algorithmique d'un problème posé de façon mathématique. Elle se concentre sur les problèmes de décision qui posent la question de l'existence d'une solution comme le problème de satisfiabilité booléenne [Cook, 1971].

La théorie de la complexité repose sur la notion de classes de complexité qui permettent de classer les problèmes en fonction de la complexité des algorithmes utilisés pour les résoudre. Parmi les classes les plus courantes, on distingue : la *classe P* (Polynomial time) qui englobe les problèmes pour lesquels il existe un algorithme déterministe de résolution en temps polynomial, et la *classe NP* (Nondeterministic Polynomial time) qui contient des problèmes de décision pour lesquels la réponse *oui* peut être décidée par un algorithme non-déterministe en un temps polynomial par rapport à la taille de l'instance. Les problèmes NP-complets sont définis comme suit [Garey et al., 1979] :

Définition 1.1 (Problème NP-complet) Un problème de décision π est NP-complet s'il satisfait les deux conditions suivantes : $\pi \in$ NP, et tout problème NP se réduit à π en temps polynomial.

L'une des questions ouvertes les plus fondamentales en informatique théorique est vraisemblablement la question si « P=NP ? » (Figure 1.2). Ceci revient à trouver un algorithme polynomial pouvant résoudre un problème NP-complet. Trouver un tel algorithme, pour un seul problème appartenant à la classe NP-complet, signifierait que tous les problèmes de cette classe pourraient être résolus en temps polynomial (voir Définition 1.1) et en conséquence, que P=NP. Cependant, il est commun de penser que P≠NP, mais aucune preuve n'a encore été trouvée jusqu'à aujourd'hui.

Il est important de préciser que tous les problèmes d'optimisation ne peuvent pas être classés comme des problèmes NP-complets, puisqu'ils ne sont pas tous des problèmes de décision, même si pour chaque problème d'optimisation on peut définir un problème de décision qui a une complexité équivalente.

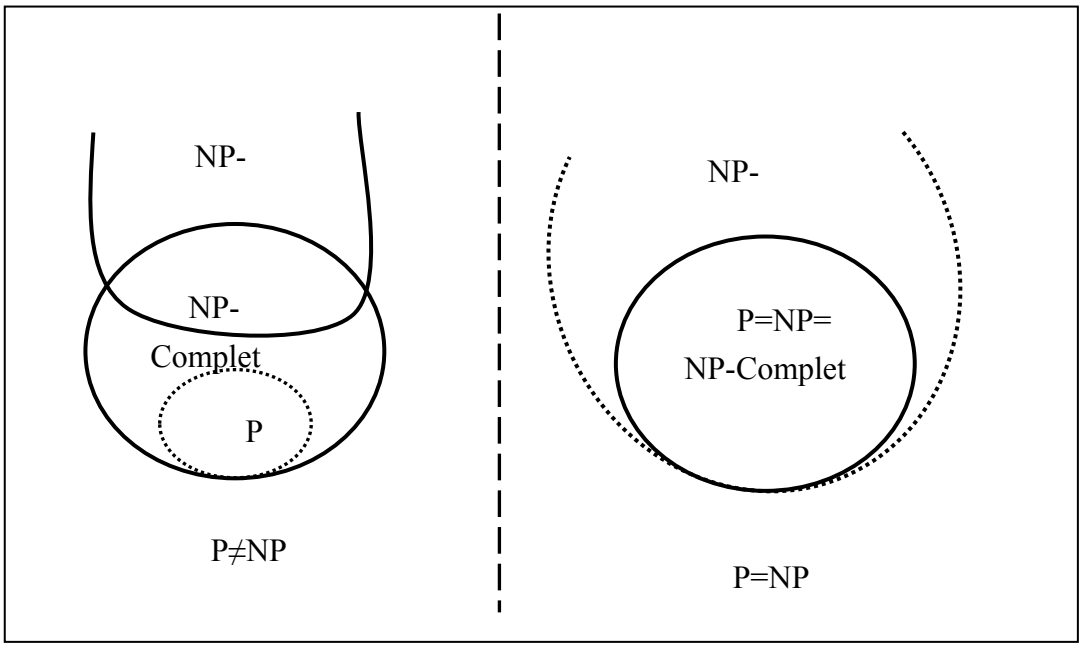


Figure.1.2 – classe P, NP, NP-complet, NP-difficile.

Définition 1.2 (Problème NP-difficile) Un problème π quelconque (de décision ou non) est NP-difficile si et seulement si il existe un problème NP-complet π′ qui est réductible à lui polynomialement.

La définition d'un problème NP-difficile est donc moins étroite que celle de la NPcomplétude [Papadimitriou et al., 1982]. De cette définition on peut observer que pour montrer qu'un problème d'optimisation est NP-difficile, il suffit de montrer que le problème de décision associé à lui est NP-complet. De cette façon un grand nombre de problèmes d'optimisation ont été prouvés NP-difficiles. C'est notamment le cas des problèmes du Voyageur de Commerce, de Partitionnement de Graphes et d'Affectation Quadratique.

1.3. Formulation mathématique des problèmes d'optimisation

Les problèmes d'optimisation combinatoire peuvent être formulés comme suit :

Définition 1.3 (Problèmes mono-objectifs) : Un problème d'optimisation est généralement formulé comme un problème de minimisation ou de maximisation, et écrit sous la forme suivant [Papadimitriou et al., 1982] :

$$\begin{cases} min_x\, f(x), \text{ Tel que,} \\ g_i(x) \leq 0\,,\ i = 1, \ldots, m\,, \\ h_j(x) = 0\,,\ j = 1, \ldots, p\,, \\ x \in S \subset R^n, \end{cases} \qquad (1.1)$$

Où f est la fonction (scalaire) à minimiser, appelée fonction coût ou fonction objectif, x représente le vecteur des variables d'optimisation, g_i sont les contraintes d'inégalité et h_j les contraintes d'égalité, et S est l'espace des variables (appelé aussi espace de recherche). S indique quel type de variables sont considérées : réelles, entières, mixtes (réelles et entières dans un même problème), discrètes, continues, bornées, etc. Un point x_A est appelé un point admissible si $x_A \in S$ et si les contraintes d'optimisation sont satisfaites : $gi(xA) \leq 0\,,\ i = 1, \ldots, m$ et $hj(xA) = 0\,,\ j = 1, \ldots, p$. La solution de (1.1) est l'ensemble des optima $\{x^*\}$.

Définition 1.4 (Problèmes multi-objectifs) : D'un point de vue mathématique, un problème d'optimisation multi-objectif, se présente, dans le cas où le vecteur $\vec{f}$ regroupe k fonctions objectif, de la façon suivante [Fonseca and Fleming, 1993] :

$$\begin{cases} min_x\ \vec{f}(x)\,, \text{ Tel que,} \\ \vec{g_i}(x) \leq 0\,,\ i = 1, \ldots, m\,, \\ \vec{h_i}(x) = 0\,,\ j = 1, \ldots, p\,, \quad x \in S \subset R^n, \end{cases} \qquad (1.2)$$

De ce fait, résoudre un problème d'optimisation combinatoire nécessite l'étude de trois points suivants :

– la définition de l'ensemble des solutions réalisables,
– l'expression de l'objectif à optimiser,
– le choix de la méthode d'optimisation (exacte ou approchée) à utiliser.

Les deux premiers points relèvent de la modélisation du problème, le troisième de sa résolution.

Définition 1.5 (Voisinage et mouvement) : Soit une solution s, on dit que s* est une solution voisine de s, si on peut obtenir s* en modifiant légèrement s. On dit aussi qu'on peut passer de s à s* en effectuant un mouvement. Le voisinage V(s) de s est l'ensemble des solutions voisines de s. La fonction $V : S \rightarrow S^2, \forall s \in S, V(s) \subset S$ est associe à chaque point de S un sous-ensemble de S [Devarenne, 2007].

Définition 1.6 (Optimum local) : Une solution $s \in S$ est un optimum local si et seulement si il n'existe pas de solution $s_0 \in v(s)$, dont l'évaluation est de meilleure qualité que s, soit :

$$\forall\ s_0 \in v(s) \begin{cases} f(s) \leq f(s_0) & \text{Dans le cas d'un problème de minimisation} \\ f(s) \geq f(s_0) & \text{Dans le cas d'un problème de maximisation} \end{cases}$$

Avec V (s) l'ensemble des solutions voisines de s.

Définition 1.7 (Optimum global) : Une solution est un optimum global à un problème d'optimisation s'il n'existe pas d'autres solutions de meilleure qualité. La solution $s^* \in S$ est un optimum global si et seulement si:

$$\forall\ s \in S \quad \begin{cases} f(s^*) \leq f(s) & \text{Dans le cas d'un problème de minimisation} \\ f(s^*) \geq f(s) & \text{Dans le cas d'un problème de maximisation} \end{cases}$$

Donc, l'optimum global est la solution s* qui vérifie la propriété précédente pour toutes les structures de voisinage du problème. La figure 1.3 schématise la courbe d'une fonction d'évaluation en faisant apparaître l'optimum global et local [Devarenne, 2007].

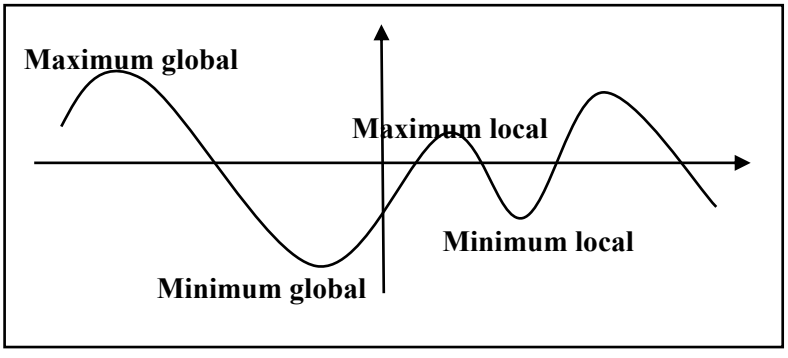


Figure 1.3 – Optima locaux et optima globaux d'une fonction à une variable.

1.4. Méthodes exactes

Le principe des méthodes exactes consiste à rechercher, souvent de manière implicite, une solution, la meilleure solution ou l'ensemble des solutions d'un problème. L'optimisation exacte concerne toutes les méthodes permettant d'obtenir un résultat dont on sait qu'il est optimal à un problème précis. Cela va des méthodes du simplex aux méthodes de Lagrangien en passant par la programmation dynamique. On peut classer les méthodes exactes en quatre grandes classes :

- La programmation dynamique,
- La programmation linéaire continue ou en nombres entiers,
- La programmation non linéaire avec ou sans contraintes,
- Les méthodes de recherche arborescente (Branch & Bound).

Des recherches plus récentes ont permis de développer la programmation dynamique et la programmation linéaire en nombres entiers [Schrijver, 1998], afin de trouver des solutions exactes à des problèmes réputés difficiles. Mais la résolution de problèmes par ces méthodes n'est pas toujours facile, notamment à cause de la difficulté à caractériser les problèmes susceptibles d'être traités. En effet, comme le temps de calcul nécessaire pour trouver une solution risque d'augmenter exponentiellement avec les dimensions du problème, les méthodes exactes sont néanmoins limitées par la taille des problèmes et les temps de calcul qui peuvent être prohibitifs avant que l'algorithme atteint la solution exacte. Nous allons présenter ci-après quelques méthodes exactes parmi les plus connues.

1.4.1. La programmation dynamique

La programmation dynamique est une des méthodes exactes les plus importantes dans la littérature pour la résolution des problèmes d'optimisation. La programmation dynamique permet de résoudre une catégorie particulière de problèmes d'optimisation sous contrainte. Elle s'applique à des problèmes d'optimisation dont la fonction objectif se décrit comme la somme de fonctions monotones non-décroissantes des ressources. Elle est basée sur le principe d'optimalité selon lequel " toute sous politique, d'une politique optimale, est optimale", et peut être appliquée à de nombreux problèmes complètement différents tels que des problèmes statistiques, continus, discrets, [Rustichini, 1998]. Cette solution s'avère intéressante notamment quand les sous-problèmes sont extraits d'un problème plus grand comme le problème du sac à dos. De ce fait, la résolution d'un problème en programmation dynamique est basée sur une décomposition du problème en sous-problèmes plus simples. A chaque sous-problème correspond un ensemble d'options, représentant chacune un coût en terme de fonction objectif. Un ensemble de choix doit donc être effectué pour les différents sous-problèmes dans le but d'arriver à une solution optimale [Dumas et al. 1986]. D'autre

part, certaines méthodes de la programmation dynamique permettent de résoudre un problème en combinant les solutions de ses sous-problèmes [Pearl 84]. Elles s'adaptent généralement aux problèmes pouvant être découpés en phases ou sous-problèmes i.e., le processus de prise de décision peut être organisé de manière séquentielle.

Les algorithmes basés sur la programmation dynamique sont généralement faciles à implémenter et très efficaces pour résoudre les problèmes de petites et moyennes tailles. Néanmoins, la programmation dynamique ne s'applique pas pour tous les types de problèmes d'optimisation combinatoire et son application sur des instances de grandes tailles est généralement très coûteuse voire impossible pour des raisons de complexité spatiale et/ou temporelle comme le problème d'alignement multiple de séquences biologiques.

1.4.2. La programmation linéaire

La programmation linéaire (PL) est une branche de l'optimisation permettant de résoudre de nombreux problèmes économiques et industriels. La programmation linéaire désigne la manière de résoudre les problèmes dont la fonction objectif et les contraintes sont toutes linéaires [Guéret, 2000]. Si l'ensemble de solutions possibles S est formulé comme un ensemble de variables à valeurs dans l'ensemble des réels R et si on a des contraintes à satisfaire des inégalités linéaires et si f est une fonction linéaire en ces variables, on parle alors d'un problème de programmation linéaire (PL). Plusieurs problèmes réels de recherche opérationnelle peuvent être exprimés comme un problème de PL. Pour cette raison un grand nombre d'algorithmes pour la résolution d'autres problèmes d'optimisation sont fondés sur la résolution de problèmes linéaires. Un programme linéaire peut être défini comme suit [Chandru et al, 1999] :

$$\begin{cases} min\ c^T x \\ Ax \geq b \\ avec\ c;\ x \in R^n;\ b \in R^m;\ A \in R^{m*n}. \end{cases} \qquad (1.3)$$

x est un vecteur n-dimensionnel représentant la solution qui doit être optimisée. c'est un vecteur de la même taille représentant la fonction objectif $c^T x$. De même, la matrice A représente avec le vecteur b les contraintes du problème linéaire. Une solution d'un programme linéaire est une affectation de valeurs aux variables du problème. Une solution est réalisable si elle satisfait toutes les contraintes du problème. L'algorithme d'optimisation le plus couramment utilisé en programmation linaire est l'algorithme du simplex [Hamdy, 2007]. Ainsi, étant donné un ensemble d'inégalités linéaires sur n variables réelles, l'algorithme peut trouver la solution optimale pour un problème linéaire d'une manière itérative en démarrant avec une solution initiale.

Malheureusement, une grande partie des problèmes réels ne se modélise pas sous la forme d'un programme linéaire pur. Dans plusieurs cas, les variables doivent prendre des valeurs entières et les problèmes qui en résultent sont souvent NP-difficiles. Nous présenterons, dans la suite, des méthodes de résolution exacte de ces types de problèmes.

1.4.3. La méthode par évaluation et séparation (Branch-and-Bound)

Un moyen naturel pour résoudre un programme linaire en nombre entier (PLNE) de très petite taille consiste à calculer l'ensemble des solutions entières du problème et à en retenir la meilleure. Or, au delà de quelques variables, cette énumération explicite devient impossible du fait de la combinatoire du problème. L'idée de la recherche arborescente par séparation et évaluation est d'effectuer une énumération implicite des solutions du PLNE. La méthode par évaluation et séparation progressive (en anglais "Branch and Bound") connue aussi sous le pseudo B&B est inspirée du principe latin "Divide ut Regnes" qui préconise de diviser ses ennemis pour régner plus aisément [Shih, 1978]. Cette méthode consiste à dénombrer les solutions d'un problème d'une manière intelligente utilisant certaines propriétés du problème en question. Contrairement, à la méthode purement énumérative, Cette technique utilise des heuristiques pour s'écarter des solutions incomplètes qui ne conduisent pas à la solution que l'on souhaite. De ce fait, on arrive généralement à obtenir la solution recherchée en des temps raisonnables. Bien entendu, dans le pire cas, on retombe toujours sur l'élimination explicite de toutes les solutions du problème [Ratschek et al, 1995] [Corrêa, 1997]. Pour appliquer la méthode de branch-and-bound, nous devons avoir :

- Le calcul d'une borne inférieure: la borne inférieure permet d'éliminer certaines solutions de l'arbre de recherche afin de faciliter l'exploration du reste de l'arbre et de faire converger la recherche vers la solution optimale.
- Le calcul d'une borne supérieure: la borne supérieure peut être une heuristique qui permet d'élaguer certaines branches de l'arbre de recherche.
- Application des règles de dominance: dans certains cas et pour certains problèmes, il est possible d'établir des règles qui permettent d'élaguer des branches de l'arbre de recherche.
- Stratégie de recherche: Il existe plusieurs techniques pour l'exploration de l'arbre de recherche. On peut utiliser soit une exploration en profondeur d'abord (Depth First Search -DFS), soit une exploration en largeur d'abord (Breadth First Search-BFS), soit encore une recherche qui utilise une file de priorité.
- Choix du critère d'évaluation: Lors d'une application de branch and bound, on a besoin d'avoir une fonction qui permet d'évaluer chaque nœud traité, et éventuellement élaguer ceux qui sont inutiles. De même, un nœud peut être élagué dans trois cas possibles: dans le premier cas, on arrive à un stade où la valeur de la borne inférieure d'un nœud courant est plus grande ou égale à la valeur de la borne supérieure qu'on avait établie auparavant.

Dans le deuxième cas, la solution n'est pas réalisable, l'un des critères d'évaluation n'a pas été respecté. Le troisième cas est le cas où on a obtenu une solution réalisable, tous les critères sont valides, mais la solution obtenue est supérieure à la borne inférieure.

Par convenance, l'exécution de la méthode de branch-and-bound se déroule à travers une arborescence, qu'elle construit et parcourt progressivement. Généralement, l'arbre construit est binaire. Nous allons faire cette hypothèse pour simplifier la présentation de la méthode. La généralisation à un arbre quelconque se fait sans difficulté. Les sommets de degrés extérieurs égaux à 0, en cours de construction de l'arbre, sont appelés *feuilles* ou *sommets terminaux* de l'arbre en cours de construction.

1.4.3.1 Procédure de séparation

A chaque sommet x_i de l'arbre correspond un domaine de solutions $ds_i \subset DS$ de façon qu'à tout moment de la construction du graphe, l'arbre possède la propriété suivante: Les ds_i associés aux feuilles du graphe forment une partition de DS (figure 1.4).

Pour ce faire, l'arbre est construit au départ d'un sommet (racine) dont le domaine associé est *DS* tout entier. Ensuite, à chaque étape un sommet x_i de niveau $N \geq 0$ et de domaine ds_i est analysé. Le B&B lui associe 2 descendants *y* et *y'* de niveau *N+1* dont les domaines respectifs forment une partition de ds_i. Les descendants du sommet x_i s'obtiennent en fixant une caractéristique binaire de la solution. Pour ce faire, on choisit une variable de séparation et on coupe son domaine de valeurs possibles en deux sous domaines. Une fois le sommet x_i analysé et les sommets *y* et *y'* définis, x_i n'est plus candidat à une étude ultérieure.

L'exemple ci-dessous montre un exemple de construction progressive de l'arborescence. Dans les cadres on trouve les choix de problèmes décomposés et la formulation de la décomposition du problème initial P0 en sous-problèmes Pi(j)(k) dont les domaines de solutions forment une partition de DS .

1.4.3.2 Procédure d'évaluation

Le procédé de construction de l'arbre étant acquis, il faut décider comment on choisit la feuille de l'arbre qui va être étudiée. Ce choix entre *y* et *y'*, ou leur abandon provisoire pour celui d'une autre feuille *z* non encore explorée est réglé par la procédure d'évaluation. A chaque sommet *x*, de domaine *DS* on associe une valeur *V(x)* qui constitue une borne inférieure de *F* sur *DS*. Parmi toutes les feuilles, le sommet choisi pour poursuivre la construction de l'arbre sera celui pour lequel *V(x)* est minimum. Le parcours nécessite donc parfois de "remonter" certains arcs en sens inverse pour revenir à des sommets de niveaux supérieurs avant de poursuivre l'exploration-construction du graphe. Cette façon de faire est appelée "backtracking". Le backtracking peut faire remonter à des niveaux proches de la racine pour lesquels peu de variables sont fixées et donc les domaines de solution sont encore vastes. Le backtracking doit donc être évité autant que possible grâce à un choix judicieux de la technique de séparation et la qualité de la technique d'évaluation.

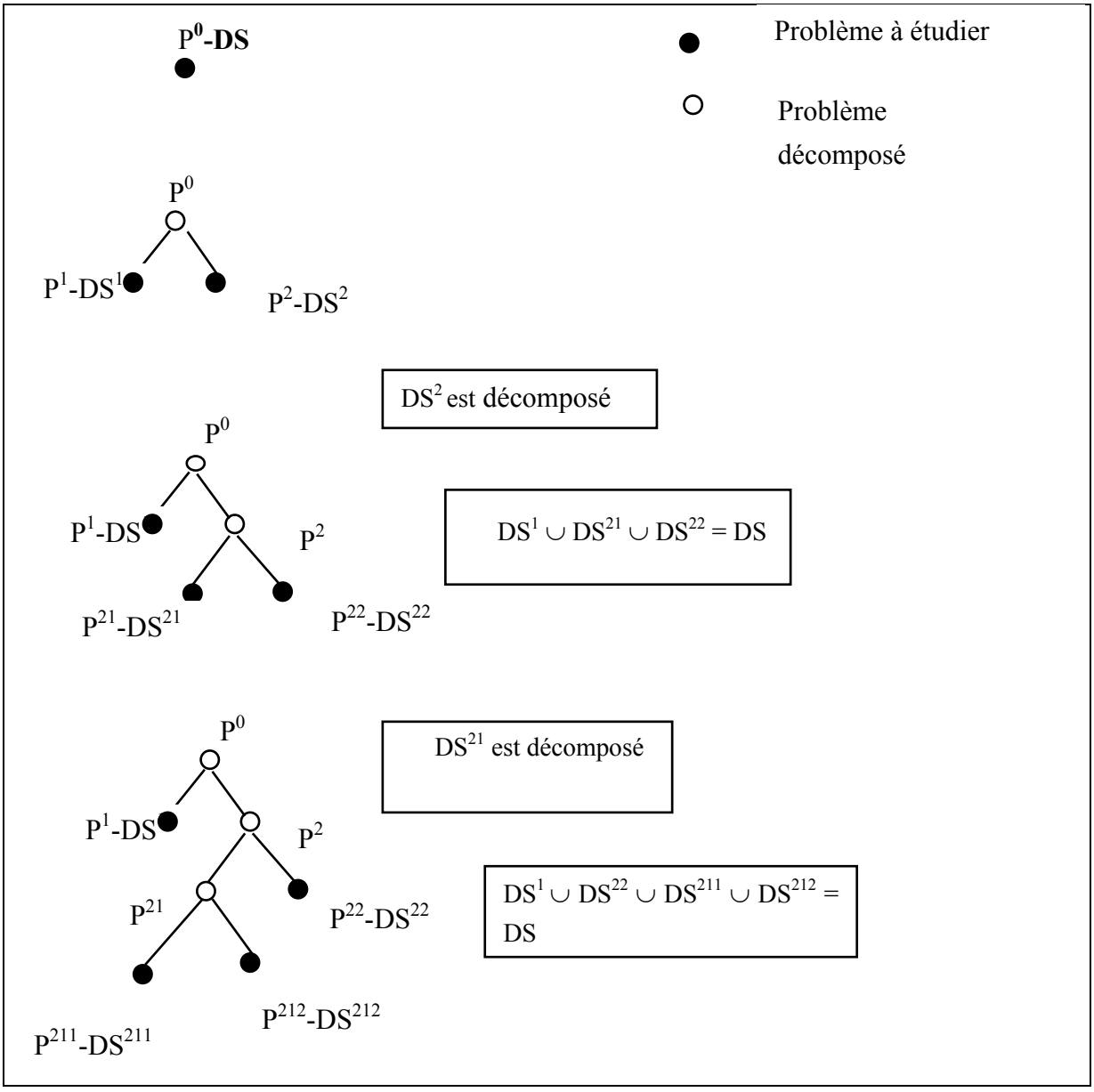


Figure 1.4 – Exemple de construction progressive de l'arborescence.

La procédure B&B est un algorithme et non une heuristique. Elle fournit bien la solution optimale du problème qu'elle résout. Bien que généralement, elle donne de bons résultats en terme de vitesse de calcul, il peut arriver qu'elle doive "backtracker" un grand nombre de fois et qu'elle finisse par construire et explorer un arbre gigantesque. Plusieurs amélioration de l'algorithme branch-and-bound ont été proposées, comme branch-and-cut, branch-and-price qu'on les note généralement par l'expression « *branch and * ».

1.5. Méthodes Métaheuristiques

Les métaheuristiques constituent une classe de méthodes qui fournissent des solutions de bonne qualité en temps raisonnable à des problèmes combinatoires réputés difficiles pour lesquels on ne connaît pas de méthode classique plus efficace. On appelle métaheuristiques (du grec, meta = qui englobe) des méthodes conçues pour échapper aux minima locaux. Le terme meta s''explique aussi par le fait que ces méthodes sont des structures générales dont il faut instancier les composants en fonction du problème par exemple, le voisinage, les solutions de départ ou les critères d''arrêt. Les métaheuristiques sont généralement des algorithmes stochastiques itératifs qui progressent vers un optimum global, c'est-à-dire l'extremum global d'une fonction en évaluant une certaine fonction objectif. Elles se comportent comme des algorithmes de recherche, tentant d''apprendre les caractéristiques d''un problème afin d''en trouver une approximation de la meilleure solution d'une manière proche des algorithmes d'approximation [Baeck et al., 1997]. L'intérêt croissant apporté aux métaheuristiques est tout à fait justifié par le développement des machines avec des capacités calculatoires énormes, ce qui a permis de concevoir des métaheuristiques de plus en plus complexes qui ont fait preuve d'une certaine efficacité lors de la résolution de plusieurs problèmes à caractère NP-difficile.

Il existe de nombreuses métaheuristiques allant de la simple recherche locale à des algorithmes plus complexes de recherche globale. Ces méthodes utilisent cependant un haut niveau d''abstraction leur permettant d''être adaptées à un large éventail de problèmes d''optimisation combinatoire. Nous pouvons partager les méthodes heuristiques en deux catégories. Celles qui permettent de déterminer un minimum local, et celles qui s'efforcent de déterminer un optimum global.

On appelle méthode (ou algorithme ou recherche) locale celle qui converge vers un minimum local. Les méthodes de recherche locale, appelées aussi méthodes de recherche par voisinages, partent d''une solution initiale et, par raffinements successives, construisent des suites de solutions de coûts décroissants pour un problème de minimisation. Le processus s''arrête lorsqu''on ne peut plus améliorer la solution courante ou parce que le nombre maximal d''itérations (fixé au départ) est atteint [Aarts and al., 1997]. Quoique, ces méthodes ne soient pas complètes (rien n''assure qu''elles trouvent toutes les solutions existantes), ni n''assurent la preuve d''optimalité, de telles méthodes parviennent très souvent à trouver des solutions de bonne qualité dans des temps de calcul raisonnables. En effet, elles sont souvent les premières méthodes testées sur les nouveaux problèmes combinatoires émergeant des applications réelles et académiques. On trouve dans la littérature de nombreuses méthodes locales. Les plus anciennes et les plus utilisées sont : la méthode de la descente [Papadimitriou, 1982], le recuit simulé [Kirkpatrick et al., 1983], la recherche tabou [Glover and al., 1997], etc.

Contrairement aux méthodes de recherche locale, les méthodes globales ont pour objectif d'atteindre un ou plusieurs optima globaux. Ces méthodes sont appelées également des méthodes à population. Celles-ci sont d'une grande diversité : parmi elles on retrouve notamment les algorithmes génétiques [Goldberg, 1994], les algorithmes à évolution différentielle [Storn et al., 1997], la recherche dispersée [Cordeau et al., 2004]. Dans cette thèse on s'intéresse plus particulièrement aux méthodes évolutionnaires.

D'autre part, on peut partager les métaheuristiques en deux grandes classes: les métaheuristiques à solution unique (i.e.; évoluant avec une seule solution) et celles à solution multiple ou population de solutions (Figure 1.5). Les méthodes d'optimisation à population de solutions améliorent, au fur et à mesure des itérations, une population de solutions. L'intérêt de ces méthodes est d'utiliser la population comme facteur de diversité.

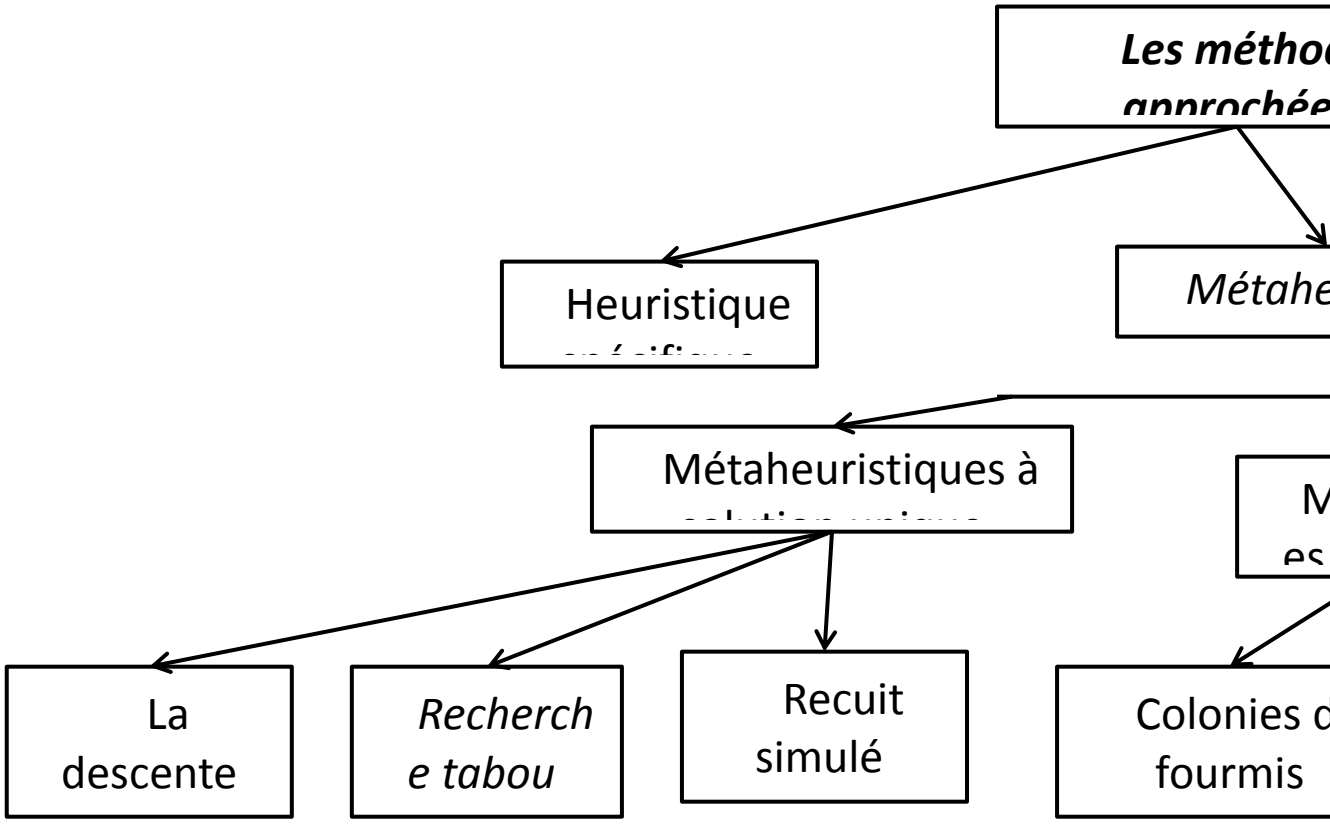


Figure 1.5 – Classification des métaheuristiques.

Finalement une « pseudo-classe » d'algorithmes a émergé ces dernières années pour résoudre des problèmes d'optimisation, qui contient des méthodes hybrides. Le principe consiste à combiner des algorithmes exacts et/ou des algorithmes approchés pour essayer de tirer profit des points forts de chaque approche et améliorer le comportement global de l'algorithme. Néanmoins, ces méthodes appartiennent forcément à l'une des classes de méthodes de résolution vues précédemment. En fait, une méthode hybride est soit exacte (i.e. donne une solution optimale) ou bien approchée (i.e. donne une solution approchée).

1.6. Algorithmes évolutionnaires

Parmi l'ensemble de techniques de recherche et d'optimisation, le développement des algorithmes évolutionnaires (AE) a été très important dans la dernière décennie [Dréo et al. 2006] [Eiben et al, 2003]. Les algorithmes évolutionnaires font partie du champ de l'Intelligence Artificielle (IA), il s'agit d'IA de bas niveau, inspirée par " l'intelligence " de la nature. Les algorithmes évolutionnaires sont basés sur le concept de la sélection naturelle élaborée par Charles Darwin [Koza, 1999]. En effet, ces algorithmes adoptent une sorte d'évolution artificielle analogue à l'évolution naturelle. Le vocabulaire utilisé dans les AE est directement inspiré de celui de la théorie de l'évolution et de la génétique. En effet, nous trouvons des mots d'individus (solutions potentielles), de population, de gènes (variables), de chromosomes, de parents, de croisement, de mutations, etc., et on s'appuie constamment sur des analogies avec les phénomènes biologiques. Il s'agit de simuler l'évolution d'une population d'individus divers à laquelle on applique différents opérateurs d'évolution comme les recombinaisons, les mutations, la sélection, etc (figure 1.6). Si la sélection s'opère en se basant sur une fonction d'adaptation, alors la population tend à s'améliorer. Contrairement aux techniques classiques de résolution de problèmes d'optimisation, un algorithme évolutionnaire est plus rapide, adaptable aux changements de l'environnement, et ne nécessite aucune connaissance du problème. En effet, on peut représenter celui-ci par une boîte noire comportant des entrées (les variables) et des sorties (les fonctions objectifs). L'algorithme ne fait que manipuler les entrées, lire les sorties, manipuler à nouveau les entrées de façon à améliorer les sorties, etc., [De Jong, 2006].

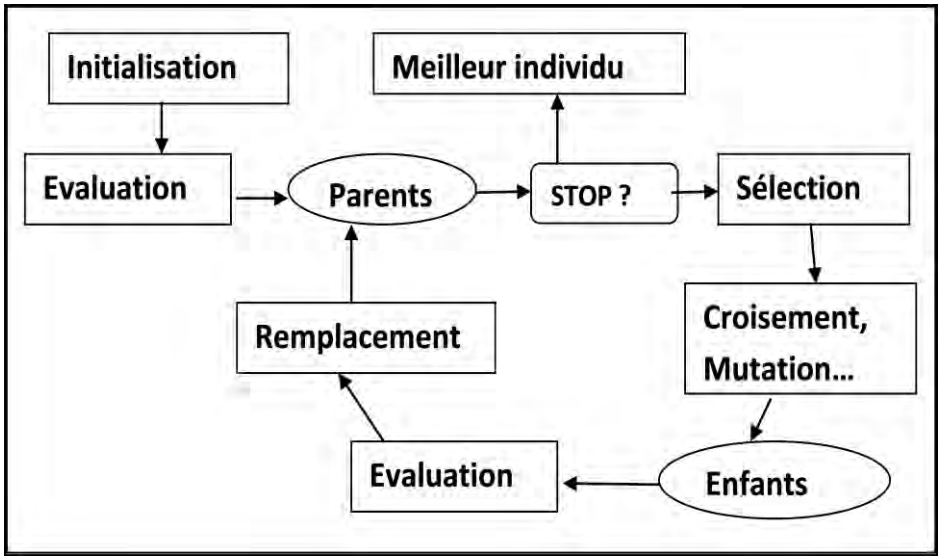


Figure 1.6 – Squelette d'un algorithme évolutionnaire.

On distingue quatre types d'AE (figure 1.7): les Algorithmes Génétiques (AG), les Stratégies d'Évolution (SE), la Programmation Évolutionnaire (PE), et la Programmation

Génétique (PG). Dans les années 90, ces quatre champs ont commencé à sortir de leur isolement et ont été regroupés sous le terme anglo-saxon *d'Evolutionnary Computation* [De Jong, 2006] [Eiben et al., 2003].

Le rapport entre la qualité de la solution finale et le temps d'exécution des métaheuristiques se diffère d'un algorithme à un autre, et dépend étroitement du problème à optimiser. Généralement, la fonction objectif a un grand impact sur la complexité de l'algorithme. En effet, le coût machine d'une optimisation est conditionné par le nombre d'évaluation de la fonction objectif. On trouve plusieurs applications des métheuristques dans de multiples domaines: optimisation de fonctions numériques difficiles (discontinues, multimodales, bruitées…), traitement d'image (alignement de photos satellites, reconnaissance de suspects…), optimisation d'emplois du temps, optimisation de design, contrôle de systèmes industriels, apprentissage des réseaux de neurones, etc.

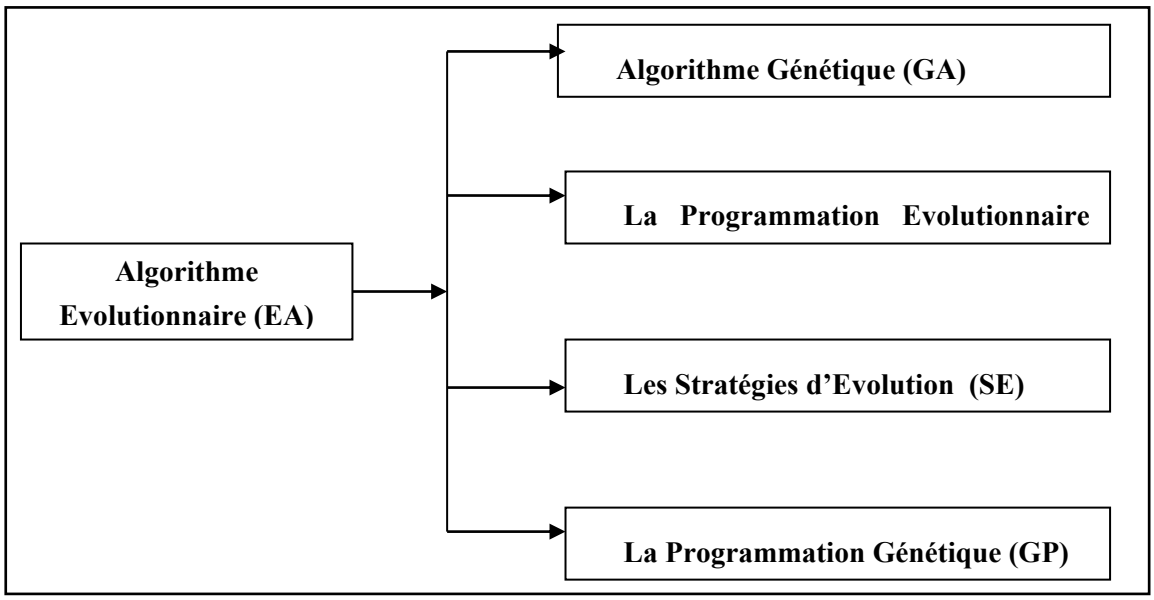


Figure 1.7 – Présentation de types des algorithmes évolutionnaires (EA).

1.7. Algorithmes génétiques

Les algorithmes génétiques sont à la base des algorithmes d'optimisation stochastiques, mais peuvent également servir pour l'apprentissage automatique, par exemple. Les premiers travaux dans ce domaine ont commencé dans les années cinquante, lorsque plusieurs biologistes américains ont simulé des structures biologiques sur ordinateur. Puis, entre 1960 et 1970, John Holland [Holland, 1975], sur la base des travaux précédents, développa les principes fondamentaux des algorithmes génétiques dans le cadre de l'optimisation mathématique. Malheureusement, les ordinateurs de l'époque n'étaient pas assez puissants pour envisager l'utilisation des algorithmes génétiques sur des problèmes réels de grande taille. Le vrai engouement pour les algorithmes génétiques a commencé en 1989 avec

l'ouvrage de référence écrit par D.E. Goldberg [Goldberg, 1989], marquant le début d'un nouvel intérêt pour cette technique d'optimisation, qui reste néanmoins très récente. Goldberg a bien décrit l'utilisation de ces algorithmes dans le cadre de résolution de problèmes concrets en permettant de mieux faire connaître ces derniers dans la communauté scientifique. En même temps, la formalisation mathématique associée aux algorithmes génétiques s'est développée mais reste encore bien limitée du faite de la complexité théorique de ces algorithmes. Parallèlement, des techniques proches ont été élaborées, dont on peut notamment citer la programmation génétique [Koza, 1999] ou les stratégies évolutionnistes [Baeck et al., 1995]. Les algorithmes génétiques possèdent plusieurs applications dans nombreux domaines aussi bien académique qu'industriels.

1.7.1. Les éléments des algorithmes génétiques

Les algorithmes génétiques (AG) sont inspirés de la génétique classique et utilisent le même vocabulaire. De manière générale, un algorithme génétique est constitué d'une population P de solutions appelées individus, dont l'adaptation à leur entourage est mesurée grâce à une fonction d'aptitude g qui retourne une valeur réelle, appelée fitness qui est l'équivalent de la fonction objectif dans la recherche opérationnelle. Le principe général d'un AG consiste à simuler l'évolution d'une population d'individus en utilisant des opérateurs évolutionnaires jusqu'à la satisfaction d'un critère d'arrêt. Avant d'expliquer en détail le fonctionnement d'un algorithme génétique, nous allons présenter quelques mots de vocabulaire relatifs à la génétique. Ces mots sont souvent utilisés pour décrire un algorithme génétique [Heudin, 1994] [Setubal et al, 1997] [Jenny, 2009].

Définition 1.8. (Gène) Un gène est une suite de bases azotées (adénine (A), cytosine (C), guanine (G) et la thymine (T)) qui contient le code d'une protéine donnée. On appellera gène la suite de symboles qui codent la valeur d'une variable. Dans le cas général, un gène correspond à un seul symbole (0 ou 1 dans le cas binaire). Une mutation changera donc systématiquement l'expression du gène muté.

Définition 1.9. (Chromosome) Un chromosome est constitué d'une séquence finie de gènes qui peuvent prendre des valeurs appelées allèles qui sont prises dans un alphabet qui doit être judicieusement choisi pour convenir au problème étudié.

Définition 1.10. (Individu) On appelle individu une des solutions potentielles. Dans la plupart des cas un individu sera représenté par un seul chromosome, dans ce cas, par abus de langage, on utilisera indifféremment individu et chromosome.

Définition 1.11. (Population) On appellera population l'ensemble des solutions potentielles qu'un AG utilise.

Définition 1.12. (Génération) On appelle génération l'ensemble des opérations qui permettent de passer d'une population Pi à une population Pj. Ces opérations sont

généralement : sélection des individus de la population courante, application des opérateurs génétiques, évaluation des individus de la nouvelle population.

Définition 1.13. (La fitness ou fonction d'évaluation) : La fonction de fitness est la pièce maitresse dans le processus d''optimisation, c''est l''élément qui permet aux algorithmes génétiques de prendre en compte un problème donné. Pour que le processus d''optimisation puisse donner de bons résultats, il faut concevoir une fonction de fitness permettant une évaluation pertinente des solutions d''un problème sous forme chiffrée. Cette fonction est déterminée en fonction du problème à résoudre et du codage choisi pour les chromosomes. Pour chaque chromosome, elle attribue une valeur numérique, qui est supposée proportionnelle à la qualité de l'individu en tant que solution. Le résultat renvoyé par la fonction d'évaluation va permettre de sélectionner ou de refuser un individu selon une stratégie de sélection [Vose, 1998].

1.7.2. Principes généraux des algorithmes génétiques

Nous allons présenter d''une manière abstraite le principe et le fonctionnement des algorithmes génétiques. Les opérations successives utilisées dans les algorithmes génétiques sont illustrées par la figure ci-après (figure1.8) [Goldberg, 1994].

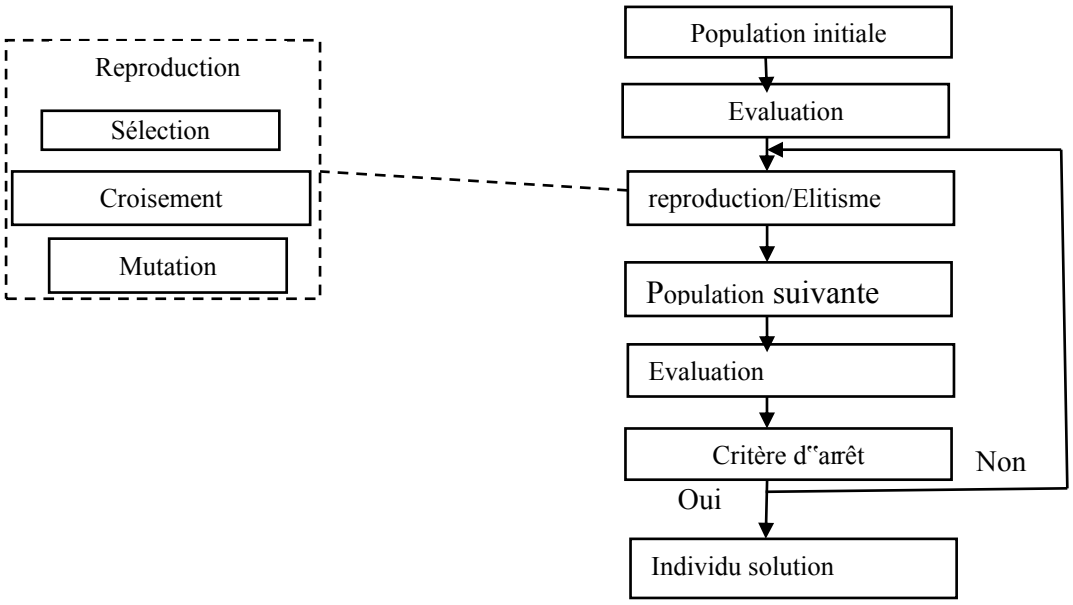


Figure 1.8– Les opérations successives utilisées dans les algorithmes génétiques.

L''algorithme génétique est constitué des étapes suivantes :

- Tout d''abord une population d'individus est générée de façon aléatoire.
- On procède à l''évaluation de l''ensemble des individus de la population initiale, cette évaluation est utilisée comme un critère de sélection lors de l''étape de la sélection.

- On sélectionne un certain nombre d'individus dans la population, afin de produire une population intermédiaire qui va générer la nouvelle génération.
- On sélectionne deux chromosomes parents P1 et P2 en fonction de leurs adaptations. On applique aléatoirement l'opérateur de croisement avec une probabilité Pc pour générer deux chromosomes enfants C1 et C2. L'opération de croisement est suivie par l'opération de mutation avec une probabilité Pm, ce qui produit deux nouveaux individus C'1 et C'2 pour lesquels on évalue leurs fitness avant de les insérer dans la nouvelle population. Contrairement à la reproduction et au croisement qui favorisent l'intensification, cet opérateur favorise la diversification des individus. On réitère les opérations de sélection, de croisement et de mutation afin de compléter la nouvelle population, ceci termine le processus d'élaboration d'une génération.
- On réitère les opérations précédentes à partir de la seconde étape jusqu'à la satisfaction du critère d'arrêt.

De manière plus formelle, voici un algorithme génétique de base :

Algorithme 1.1 : Algorithme génétique de base

Début
1: Générer une population aléatoire de n chromosomes.
2: Evaluer la fitness des chromosomes avec la fonction f (x)
3: **Répéter**
4: Calculer la fonction fitness f (x), pour tout chromosome x
5: Appliquer l'opération de sélection
6: Appliquer l'opération de croisement avec une probabilité PC
7: Appliquer l'opération de mutation avec une probabilité PM
8: Ajouter les nouveaux chromosomes à la nouvelle population
9: Calculer la fonction fitness f (x), pour tout chromosome x
10: Appliquer l'opération de remplacement
11 : **Jusqu'à** la satisfaction des conditions de terminaison
Fin

1.7.3. Codage

C'est une modélisation d'une solution d'un problème quelconque en un chromosome. Le choix du codage est important, souvent il est délicat. En effet, le choix du type de codage ne peut pas être effectué de manière évidente. La méthode actuelle à appliquer dans le choix du codage consiste à choisir celui qui semble le plus naturel en fonction du problème à traiter. L'objectif est d'abord de pouvoir coder n'importe quelle solution. Mais il est souhaitable, au-

delà de cette exigence, d'imaginer soit un codage tel que toute chaîne de caractères représente bien une solution réalisable du problème, soit un codage qui facilite ensuite la conception du croisement de telle sorte que les « enfants » obtenus à partir de la recombinaison de leurs « parents » puissent être associés à des solutions réalisables, au moins pour un grand nombre d'entre eux [Banzhaf, 1999] [Charbonneau, 2002]. Parmi les codages les plus utilisés on peut citer :

1.7.3.1 Codage binaire

Ce type de codage est certainement le plus utilisé car il présente plusieurs avantages. Son principe est de coder la solution selon une chaîne de bits (les valeurs 0 ou 1). Le codage binaire a permis certes de résoudre beaucoup de problèmes, mais il s'est avéré obsolète pour certains problèmes d'optimisation numérique. En effet, il est plus pratique d'utiliser un codage réel des chromosomes.

1	**0**	**0**	**1**	**1**	**0**	**1**

Figure1.9 – codage binaire d'un chromosome (problème Max Sat).

1.7.3.2 Codage réel

Une autre approche semblable est de coder les solutions en tant que suites de nombres entiers ou de nombres réels, où chaque position représente encore un certain aspect particulier de la solution. Cette approche tient compte d'une plus grande précision et de complexité que le codage basé sur les nombres binaires seulement. Ce type de codage est intuitivement plus proche à l'environnement des problèmes à résoudre.

3	**5**	**2**	**1**	**4**	**6**

Figure 1.10 – Codage réel d'un chromosome
(Problème TSP).

1.7.3.3 Codage à l'aide de suite alphabétique

Une troisième approche est de représenter des individus dans un algorithme génétique comme une suite de caractères, où chaque caractère représente encore un aspect spécifique de la solution. Ce type de codage est utilisé dans de nombreux cas poussés d'algorithmes génétiques comme en bioinformatique.

A	**-**	**G**	**T**	**C**

Figure 1.11 – Codage alphabétique d'un chromosome
(Problème bioinformatique).

1.7.3.4 Codage sous forme d'arbre

Ce codage utilise une structure arborescente avec une racine de laquelle peuvent être issus un ou plusieurs fils. Un de leurs avantages est qu'ils peuvent être utilisés dans le cas de problèmes où les solutions sont de taille infinie.

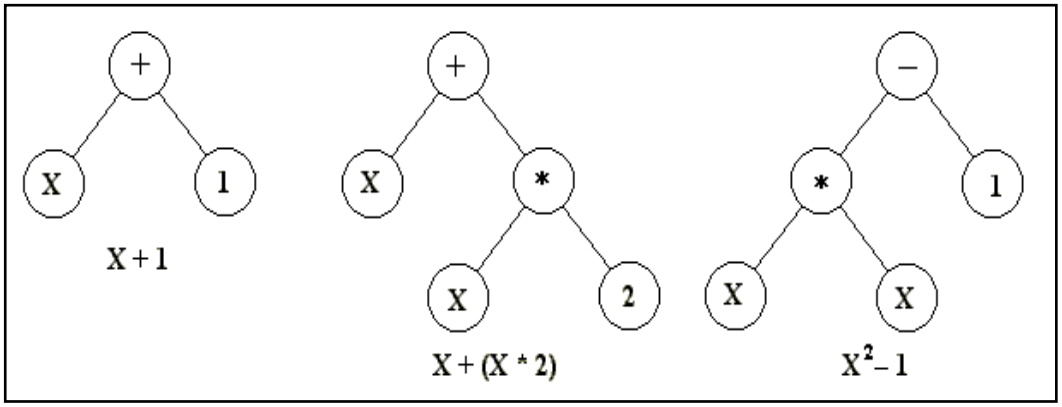


Figure 1.12 – Codage en arbre : Trois arbres simples de programme de la sorte normalement utilisée dans la programmation génétique.

1.7.4 Les opérateurs d'un algorithme génétique

Quatre opérateurs caractérisent les algorithmes génétiques et rappellent l'origine de ces méthodes. Ils vont permettre à la population d'évoluer, par la création d'individus nouveaux construits à l'aide des individus anciens. Ces opérations sont : la sélection, le croisement, la mutation et le remplacement [Goldberg, 1994].

1.7.4.1 La sélection

La sélection permet d'identifier statistiquement les meilleurs individus de la population courante qui seront autorisés à se reproduire, cette opération est fondée sur la performance des individus, estimée à l'aide de la fonction d'adaptation. Il existe différentes méthodes de sélection :

- *Sélection par roulette (Wheel)*

Elle consiste à associer à chaque individu un segment dont la longueur est proportionnelle à sa fitness. Ces segments sont ensuite concaténés sur un axe gradué que l'on normalise entre 0 et 1 (figure 1.13). On tire alors un nombre aléatoire de distribution uniforme entre 0 et 1, puis on regarde quel est le segment sélectionné, et on reproduit l'individu correspondant. Avec cette technique, les bons individus seront plus souvent sélectionnés que les mauvais, et un même individu pourra avec cette méthode être sélectionné plusieurs fois. Néanmoins, sur des populations de petite taille, il est difficile d'obtenir exactement l'espérance mathématique de sélection à cause du faible nombre de tirages. Le cas idéal d'application de cette méthode est bien évidemment celui où la population est de taille infinie. On aura donc un biais de sélection plus ou moins fort suivant la dimension de la population. Un autre problème

rencontré lors de l'utilisation de la sélection par roulette est lorsque la valeur d'adaptation des chromosomes varie énormément. Si la meilleure fonction d'évaluation d'un chromosome représente 90% de la roulette alors les autres chromosomes auront très peu de chance d'être sélectionnés et on arriverait à une stagnation de l'évolution.

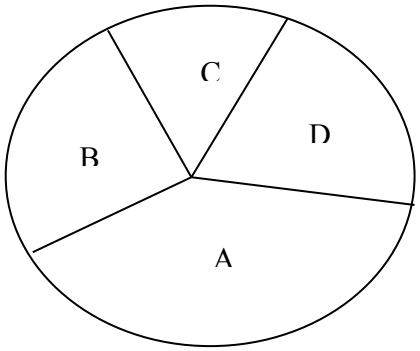


Figure 1.13 – Sélection par roulette.

- *Sélection par rang*

La sélection par rang est basée sur le tri de la population selon la fitness de chaque individu. Elle consiste à attribuer un entier de 1 à N pour une population de N chromosomes, du mauvais au meilleur. La sélection par rang d'un chromosome est similaire à la sélection par roulette, mais les proportions sont en relation avec le rang plutôt qu'avec la valeur de l'évaluation. Avec cette méthode de sélection, tous les chromosomes ont une chance d'être sélectionnés. Cependant, son grand inconvénient est la convergence lente vers la bonne solution. Ceci est dû au fait que les meilleurs chromosomes ne diffèrent pas énormément des plus mauvais. Vous pouvez voir dans la figure ci-après, comment la situation change après avoir transformé la fitness en rang.

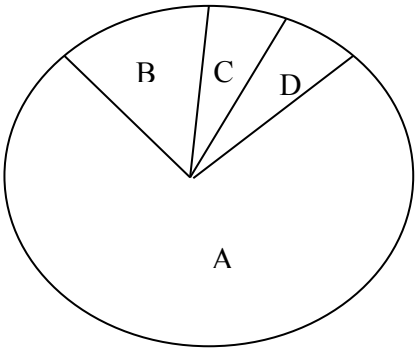
- *Sélection par tournoi*

Cette méthode est celle avec laquelle on obtient les résultats les plus satisfaisants. Elle consiste à choisir aléatoirement k individus (le nombre de participants à un tournoi) et à les confronter entre eux par le biais de la fonction d'adaptation, et de sélectionner ensuite le meilleur parmi eux. On répète ce processus autant de fois de manière à obtenir les n individus de la population qui serviront de parents. La variance de cette méthode est élevée et le fait d'augmenter ou de diminuer la valeur de *k* permet respectivement de diminuer ou d'augmenter la pression de la sélection.

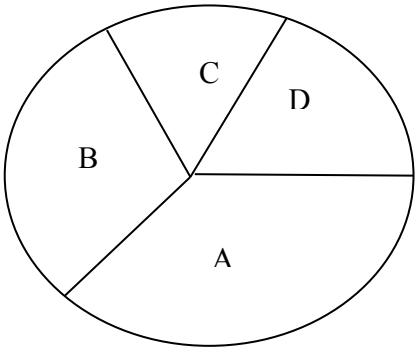
- *Sélection Steady-State*

Ce n'est pas une méthode particulière de sélection des chromosomes parents. L'idée principale est qu'une grande partie de la population puisse survivre à la prochaine génération. L'algorithme génétique fonctionne alors de la manière suivante. A chaque génération quelques

chromosomes sont sélectionnés parmi ceux qui ont le meilleur coût, afin de créer des chromosomes fils. Ensuite les chromosomes les plus mauvais sont retirés et remplacés par les nouveaux. Le reste de la population survie à la nouvelle génération.



a. Situation avant le tri (graphe de fitnesses)



b. Situation après le tri (graph d'ordre)

Figure 1.14 – Sélection par rang.

- *Elitisme*

L'élitisme consiste à conserver à chaque génération un certain nombre des meilleurs chromosomes de la population qui pourraient disparaître par les opérations de mutation, croisement ou sélection. Elle consiste à copier un ou plusieurs des meilleurs chromosomes dans la nouvelle génération. Ensuite, on génère le reste de la population selon l'algorithme de reproduction usuel. Cette méthode améliore considérablement les algorithmes génétiques, car elle permet de ne pas perdre les meilleures solutions.

1.7.4.2 Le croisement

Le croisement a pour but de produire une nouvelle génération d'individus en recombinant les individus sélectionnés par l'opérateur de sélection. C'est l'opérateur de l'algorithme génétique qui permet le plus souvent de se rapprocher de l'optimum d'une fonction en combinant les gènes. Ainsi, dans un premier temps, les individus sélectionnés sont répartis aléatoirement en couples de parents. Puis, chaque couple de parents subit une opération de

recombinaison afin de générer un ou deux enfants. Bien qu'il soit aléatoire, cet échange d'informations offre aux algorithmes génétiques une part de leur puissance : quelque fois, de "bons" gènes d'un parent viennent remplacer les "mauvais" gènes d'un autre et créent des fils mieux adaptés aux parents.

Le type de croisement le plus ancien est le croisement à découpage de chromosomes, ou *croisement 1-point* (figure 1.15 à gauche). Pour effectuer ce type de croisement sur des chromosomes constitués de L gènes, on tire aléatoirement une position inter-gènes dans chacun des parents. On échange ensuite les deux sous-chaînes de chacun des chromosomes ce qui produit deux enfants C1 et C2. Ce mécanisme présente l'inconvénient de privilégier les extrémités des individus. Selon le codage choisi, il peut générer des fils plus ou moins proches de leurs parents. Pour éviter ce problème, on peut étendre ce principe en découpant le chromosome non pas en 2 sous-chaînes mais en 3, 4, voir plus de sous-chaines. On parle alors de croisement *k-point ou multi-point* (figures 1.15 à droite).

L"autre type de croisement est le *croisement uniforme*. Le croisement uniforme consiste à générer un enfant en échangeant chaque gène des deux individus parents avec une probabilité uniforme égale à 0.5 (voir Figure 1.16). Cet opérateur peut être vu comme le cas extrême du croisement multi-point dans lequel le nombre de points de coupure est déterminé aléatoirement au cours de l'opération.

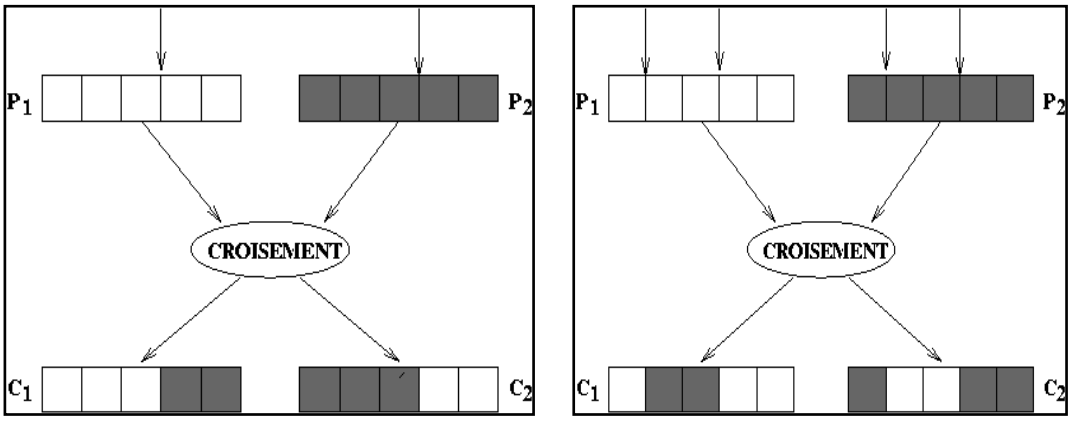


Figure 1.15 – L'opération de croisement k-point. À gauche Croisement 1-point.

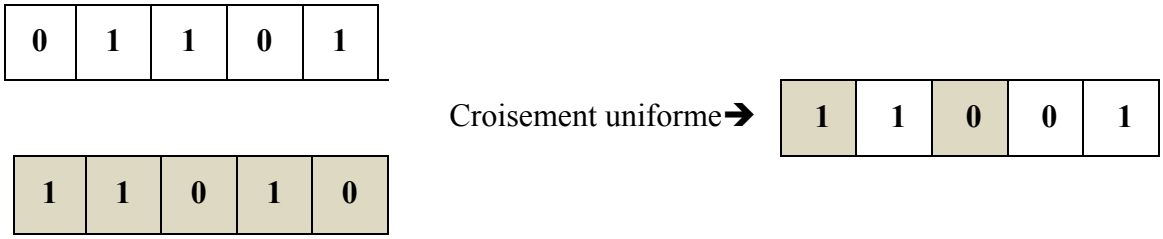


Figure 1.16 – L'opération de croisement uniforme.

1.7.4.3 La mutation

Contrairement au croisement qui est un opérateur d'exploitation, la mutation est principalement un opérateur d'exploration de l'espace de recherche. Le rôle de cet opérateur consiste à modifier aléatoirement, avec une certaine probabilité, la valeur d'un gène d'un chromosome (figure 1.17). Dans le cas du codage binaire, chaque bit $a_i \in \{0,1\}$, est remplacé selon une probabilité p_m par son complémentaire : $\tilde{a}_i = 1 - a_i$. Malgré, l'aspect marginal de la mutation dans la mesure où sa probabilité est en général fixée assez faible (de l'ordre de 1%), elle donne aux algorithmes génétiques une grande aptitude d'exploration de tous les points de l'espace de recherche. Cet opérateur joue un rôle primordial dans le processus d'optimisation, et il est loin d'être marginal, comme on le voit dans la figure 1.18, sans mutation, la solution pourrait converger vers une solution locale optimale. Une mutation réussie peut créer des solutions complètement aléatoires, conduisant à des solutions "inexplorées" qui ne peuvent être atteint via l'opération de croisement une fois que la population a convergée. L'opération de mutation a de fait un double rôle : celui d'effectuer une recherche locale et/ou éviter une stagnation autour d'optima locaux (figure1.18). On distingue dans la littérature plusieurs types de mutations selon la nature de codage utilisé, ex : mutation 1-bit, mutation réelle.

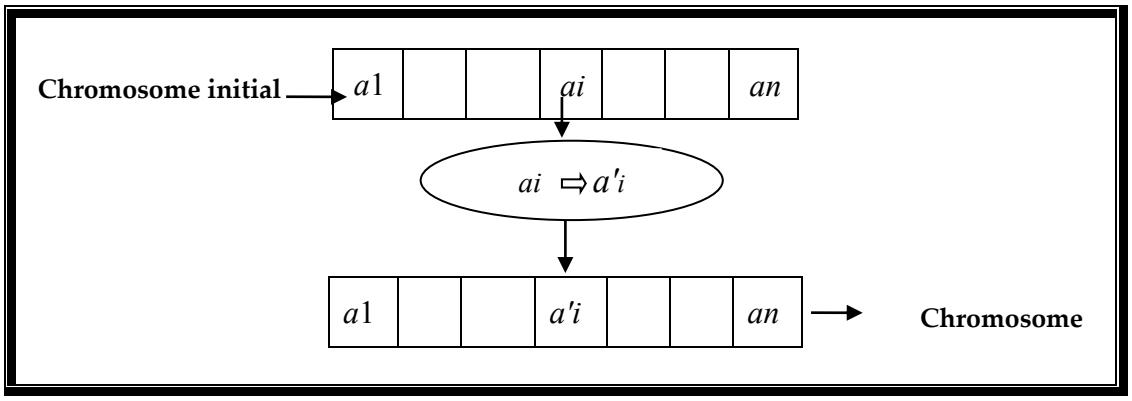


Figure 1.17 – Principe de l'opérateur de mutation.

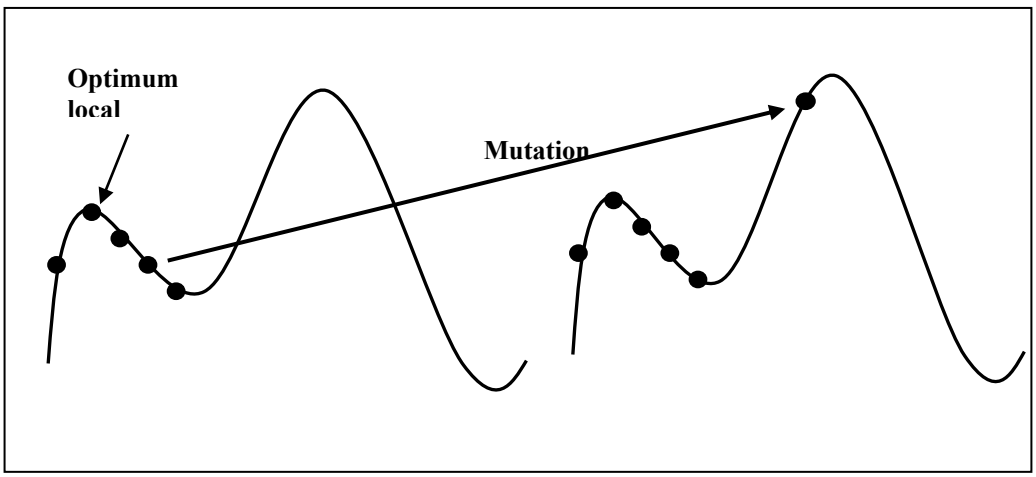


Figure 1.18 – La diversification de l'opérateur de mutation.

1.7.4.3 Opérateur de remplacement

Cet opérateur est basé, comme l'opérateur de sélection, sur la fitness des individus. Son travail consiste à déterminer les chromosomes parmi la population courante constituant la population de la génération suivante. Cette opération est appliquée après l'application successive des opérateurs de sélection, de croisement et de mutation. On trouve essentiellement 2 méthodes de remplacement différentes :

• *Le remplacement stationnaire :* dans ce cas, les enfants remplacent automatiquement les parents sans tenir compte de leurs performances respectives. Le nombre d'individus de la population est constant tout au long du cycle d'évolution, ce qui implique donc d'initialiser la population initiale avec un nombre suffisant d'individus.

• *Le remplacement élitiste :* dans ce cas, on garde au moins les individus possédant les meilleures performances d'une génération à la suivante. En général, on peut partir du principe qu'un nouvel individu (enfant) prend place au sein de la population s'il remplit le critère d'être plus performant que le moins performant des individus de la population précédente. Donc les enfants d'une génération ne remplaceront pas nécessairement leurs parents comme dans le remplacement stationnaire et par conséquent la taille de la population n'est pas figée au cours du temps. Ce type de stratégie améliore les performances des algorithmes évolutionnaires dans certains cas, mais présente aussi un désavantage en augmentant le taux de convergence prématuré.

1.7.5 Critères d'arrêt

Le critère d'arrêt indique que la solution est suffisamment approchée de l'optimum. Plusieurs critères d'arrêt de l'algorithme génétique sont possibles. On peut arrêter l'algorithme après un nombre de générations suffisant pour que l'espace de recherche soit convenablement exploré, ce critère peut s'avérer coûteux en temps de calcul si le nombre d'individus à traiter dans chaque population est important. L'algorithme peut aussi être arrêté lorsque l'évolution de la population est ralentie. On peut aussi envisager d'arrêter l'algorithme lorsque la fonction d'adaptation d'un individu dépasse un seuil fixé au départ. Nous pouvons également faire des combinaisons des critères d'arrêt précédents.

1.7.6 Avantages et limites des algorithmes génétiques

Un des grands avantages des algorithmes génétiques est qu'ils autorisent la prise en compte de plusieurs critères simultanément, et qu'ils parviennent à trouver de bonnes solutions sur des problèmes très complexes. L'avantage principal des algorithmes génétiques par rapport aux autres techniques d'optimisation combinatoire consiste en une combinaison de l'exploration de l'espace de recherche, et de l'exploitation des meilleures solutions disponibles à un moment donné. Ils doivent simplement déterminer entre deux solutions

quelle est la meilleure, afin d'opérer leurs sélections. Leur utilisation se développe dans des domaines aussi divers que l'économie, la bioinformatique ou la vérification formelle.

L'inconvénient majeur des algorithmes génétiques est le coût d'exécution important par rapport à d'autres métaheuristiques. Les algorithmes génétiques nécessitent de nombreux calculs, en particulier au niveau de la fonction d'évaluation. Mais avec les capacités calculatoire des ordinateurs récents, ce problème a perdu de son importance. D'un autre côté, l'ajustement d'un algorithme génétique est délicat : des paramètres comme la taille de la population ou le taux de mutation sont parfois difficiles à déterminer. Or le succès de l'évolution en dépend et plusieurs essais sont donc nécessaires, ce qui limite encore l'efficacité de l'algorithme. Néanmoins, la limite la plus importante des algorithmes génétiques est celle des optima locaux. En effet, lorsqu'une population évolue, il se peut que certains individus, qui à un instant occupent une place importante au sein de cette population, deviennent majoritaires. À ce moment, il se peut que la population converge vers cet individu et s'écarte ainsi d'individus plus intéressants mais trop éloignés de l'individu vers lequel on converge. Il faut mentionner également le caractère indéterministe des algorithmes génétiques. Comme les opérateurs génétiques utilisent des facteurs aléatoires, un algorithme génétique peut se comporter différemment pour des paramètres et populations identiques. Afin d'évaluer correctement l'algorithme, il faut l'exécuter plusieurs fois et analyser statistiquement les résultats.

1.8. Système immunitaire artificiel

En intelligence artificielle, dans le but de résoudre des problèmes complexes, des idées glanées à partir de mécanismes naturels ont été exploités pour développer des heuristiques inspirées de la nature. L'optimisation naturelle a été largement investiguée durant la dernière décennie ouvrant la voie à de nouveaux paradigmes de calcul comme les réseaux de neurones, les systèmes évolutionnaires ou encore les systèmes immunitaires artificiels (SIA) par les principes et les processus outre du système immunitaire naturel (SIN) [Garrett, 2005]. Les algorithmes exploitent typiquement les caractéristiques intéressantes des systèmes immunitaires naturels, comme la mémorisation, la reconnaissance de formes, l'apprentissage, et les capacités d'adaptation [De Castro, 2000] [Layeb et al, 2007].

Un système immunitaire naturel est un système complexe qui peut être analysé à différents niveaux : molécules, cellules et organes. Les interactions complexes entre les entités de chaque niveau résultent en un système complexe capable de protéger notre corps de n'importe quelle entité dangereuse appelée antigène. Les éléments de base des SIN sont les globules blancs ou lymphocytes. Pour pouvoir identifier les autres molécules, des lymphocytes particuliers appelés cellules B produisent des récepteurs appelés anticorps. La partie de l'anticorps responsable de la reconnaissance de l'antigène est appelée paratope (figure 1.19). Le paratope se lie à une partie spécifique de l'antigène appelée épitope. La liaison entre un

paratope et un épitope est d'autant plus forte que leurs formes soient complémentaires. La force de cette liaison est appelée affinité. La reconnaissance d'un antigène par une cellule B est fonction de l'affinité entre les anticorps de la cellule B et de cet antigène. Les cellules B qui vont le mieux reconnaître l'antigène vont proliférer en se clonant. C'est le principe de la sélection clonale. Les clones subissent alors des mutations somatiques qui vont promouvoir leur variation génétique. Lorsque la population atteint la maturité, les clones se différencient en cellules mémoires et cellules plasma. Cette expansion clonale confère au système immunitaire sa mémoire. D'un autre côté, les cellules B avec une faible affinité sont soit mutés, soit détruites par la sélection négative [De Castro, 2000].

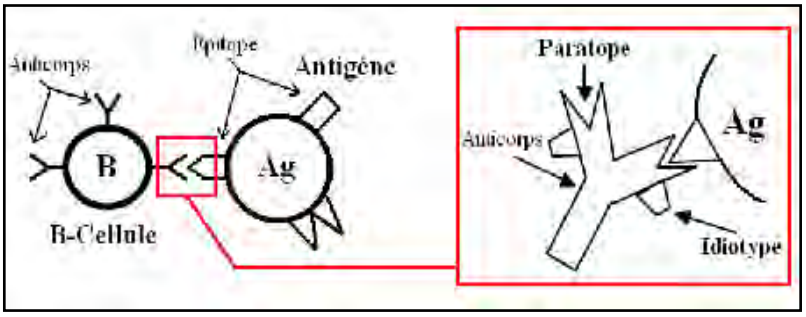


Figure 1.19 – Interaction entre l'anticorps et l'antigène.

La figure suivante illustre le mécanisme d'adaptation. Dans le cas (a) la cellule B n'a pas reconnu l'antigène et aucune réaction ne s'est produite. Dans le cas (b) la B-Cellule a reconnu l'antigène, elle sera affectée par la mutation qui changera la structure de son anticorps. Dans le cas (d) l'affinité entre l'anticorps de la cellule B et l'antigène n'a pas été amélioré. Cependant dans le cas (c) la mutation a considérablement amélioré l'affinité et c'est pour ça que ces cellules B seront clonées afin de créer des cellules de mémoire pour améliorer la réponse à des futures attaques par le même antigène.

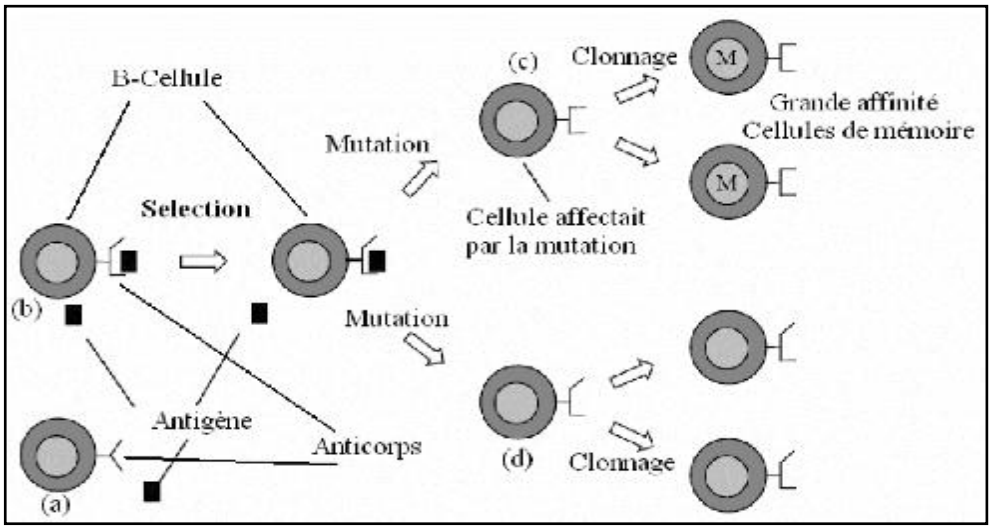


Figure 1.20 – Prolifération et maturation d'affinité.

Les systèmes immunitaires artificiels SIA sont des systèmes adaptatifs inspirés de modèles métaphoriques du système immunitaire naturel. Ils utilisent les mécanismes et concepts inhérents aux systèmes immunitaires naturels pour concevoir des approches adaptatives et robustes. Un SIA peut être vu comme une composition de méthodologies intelligentes inspirées des systèmes immunitaires naturels pour résoudre des problèmes du monde réel. Un SIA partage des similarités avec d'autres systèmes inspirés comme les algorithmes génétiques et les réseaux de neurones. Plusieurs modèles ont été inspirés des systèmes immunitaires naturels pour résoudre des problèmes variés: la sélection négative, les réseaux immunitaires et la sélection clonale. Les caractéristiques intéressantes des systèmes immunitaires ont encouragé leur adaptation au domaine informatique pour la résolution d"une large gamme de problèmes : sécurité de réseau [Dasgupta et al, 2002], alignement d'image [Bendiab et al, 2003], etc. La principale difficulté d'utilisation des SIA est de trouver le modèle biologique à partir duquel s'inspirer pour résoudre un problème ainsi que la représentation de ce problème sous forme biologique.

1.8.1 Algorithmes des systèmes immunitaires artificiels

Afin de créer des approches inspirées du système immunitaire, des algorithmes immunitaires ont été proposés.

1.8.1.1 Algorithme de la sélection clonale

L'algorithme de la sélection clonale [De Castro et al, 2002] est très connu dans le domaine des SIA. Son fonctionnement est le suivant :

Quand les anticorps d'une cellule B se lient avec un antigène, la cellule B devient active et commence à se multiplier. Les nouvelles cellules B qui sont produites sont des copies exactes de la cellule B parente, mais elles subissent le mécanisme d'hyper mutation somatique pour produire des anticorps qui sont spécifiques à l'antigène envahissant. Le principe de la sélection clonale est le terme employé pour décrire les propriétés de base d'une immuno-réaction adaptative à un stimulus antigénique. Elle établit l'idée que seulement les cellules capables d'identifier un stimulus antigénique prolifèreront et subiront un choix qui ne garde que les clones qui ont les plus grands degrés d'affinité. Généralement, pour résoudre un problème en utilisant les principes de la sélection clonale, chaque solution possible est représentée par un anticorps et le problème est un antigène. Plusieurs modèles ont été inspirés des systèmes immunitaires naturels pour résoudre des problèmes variés, par exemple les algorithmes de sélection négative et les réseaux immunitaires ont été proposés. L"algorithme de la sélection clonale a été utilisé avec succès pour la résolution de plusieurs problèmes d'optimisations, tels que l"alignement multiple de séquences [Layeb et al, 2007], optimisation et reconnaissance des caractères [Meshoul et al, 2005], etc. L"algorithme détaillé de la sélection clonale est le suivant :

Algorithme 1.2: Algorithme général de la sélection clonale

1. Générer aléatoirement une population de N anticorps;
2. Choisir les *n1* cellules qui ont la plus grande affinité à l'antigène ;
3. produire des copies identiques de ces cellules choisies. Le nombre de copies est proportionnel aux affinités: plus l'affinité est haute, plus le nombre de clone est grand;
4. Changer la structure des cellules choisies (hyper-mutation). Le taux de changement est proportionnelle à leurs affinités: plus l'affinité est haute, plus le taux de Changement est petit ;
5. Sélectionner les *n2* cellules qui ont la plus grande affinité à l'antigène pour composer le nouveau répertoire ;
6. Remplacer quelques cellules qui possèdent des valeurs d'affinité faible par les nouvelles cellules ;
7. Répéter les étapes 2 à 6 jusqu'à ce qu'un critère d'arrêt donné soit rencontré.

1.8.1.2 Algorithme de la sélection négative

La sélection négative est inspirée du processus de sélection positive et négative qui se produit pendant la maturation des cellules T dans le thymus. Ce processus est appelé tolérance des cellules T. C"est le processus qui permet de distinguer le soi du non soi. La sélection négative a été appliquée à des problèmes de détections d'anomalies, son algorithme est le suivant:

Algorithme 1.3: Algorithme général de la sélection négative

1. Produire aléatoirement des anticorps et placez-les dans un ensemble *P*,
2. Déterminer l'affinité de touts les anticorps dans *P* avec toutes les cellules de l'individu *S*,
3. Si l'affinité d'un anticorps *de P* avec au moins une cellule de *S* est supérieur ou égal à un seuil donné α, alors l'anticorps identifie l'individu et doit être éliminée; sinon l'anticorps est accepté.

1.8.1.3 Algorithme du réseau immunitaire

La sélection clonale est le moyen utilisé par le système immunitaire pour vaincre des antigènes étrangers tandis que la sélection négative lui permet de se protéger contre des antigènes du soi. Le réseau immunitaire quant à lui définit la façon dont les cellules réagissent entre elles dans le système immunitaire. Les réseaux immunitaires ont attiré l"ïntention des chercheurs sur l'intelligence artificielle [Jerne, 1973], car premièrement, elle

présente un système dynamique capable d'exécuter des interactions entre ses propres composants et entre ces derniers et l'environnement externe, deuxièmement, elle offre les possibilités d'ajuster la structure et les paramètres du réseau immunitaire sur l'environnement. L'algorithme général est :

Algorithme 1.4: Algorithme général du réseau immunitaire

Initialisation: initialiser un réseau de cellules immunisées ;

Répéter pour chaque antigène :

2.1. Identification antigénique : calculez les affinités des anticorps face à l'antigène ;

2.2. Interactions du réseau : calculez les affinités entre tous les paires d'anticorps ;

2.3. Métadynamique : ajoutez des nouveaux anticorps aux réseaux et supprimez qui sont inutile (le choix est basé sur un certain critère) ;

2.4. Niveau de stimulation: évaluez le niveau de stimulation de chaque cellule du réseau tenant compte des résultats des étapes précédentes ;

2.5. Dynamique de réseau : mettre à jour la structure et les paramètres du réseau selon le niveau de stimulation de différentes cellules ;

Jusqu'à la satisfaction d'un critère donné d'arrêt;

1.9. Algorithme à Evolution Différentielle

Le domaine des algorithmes évolutionnaires a connu un grand développement ces dernières années. L'Evolution Différentielle (ED) [Feoktistov, 2006] est l'un de ces algorithmes, qui a été présenté par Storn et Price en 1997 [Storn et al, 1997]. L'algorithme à évolution différentielle est inspiré par les algorithmes génétiques et l'évolution des stratégies combinées à une technique de recherche géométrique. Toutefois, l'ED s'écarte encore un peu plus des principes de la génétique en abandonnant le codage génotype-phénotype et en faisant varier directement les paramètres réels proches du phénotype. L'ED est donc valide pour tout type de fonction objectif à valeurs réelles et peut tenir compte des propriétés mathématiques générales de ces fonctions. En revanche, tout comme l'AG, l'ED utilise une succession de générations définies par des opérations de mutation et de recombinaison de réels et donc possède des notions d'« enfants » et de « parents ».

ED a été conçu comme une méthode de recherche parallèle, directe, et stochastique. Chaque solution est encodée avec un vecteur n-dimensionnel basées sur des nombres à virgule flottante. La taille de la population (m) ne change pas au cours du processus de minimisation (maximisation). A chaque génération, de nouveaux vecteurs sont générés par la combinaison des vecteurs choisis au hasard de la population actuelle. La nouveauté de cet algorithme réside

dans l'opérateur de mutation. Contrairement à la mutation des algorithmes génétiques, l'ED utilise un concept de mutation auto référentiel se limitant à des combinaisons de mutations déjà présentes dans la population. L'opérateur de mutation consiste à générer de nouveaux vecteurs par le calcul de la différence pondérée de deux (ou quatre) autres vecteurs, selon la formule suivante [Storn et al, 1997]:

$$\vec{v}_i = \vec{a}_{r1} + F \times (\vec{a}_{r2} - \vec{a}_{r3}) \quad (1.4)$$

Où i = 1,2,..., m, et les indices aléatoires r_1, r_2, $r_3 \in [1,2,..., m]$ sont mutuellement différents et distincts de l'indice i. Cet opérateur est utilisé en liaison avec liaison de l'opérateur de croisement. Ce dernier est introduit en vue d'augmenter la diversité de la population. Il combine un vecteur muté obtenu par l'opération de mutation avec le vecteur avec l'un des vecteurs de la population. Finalement, L'opérateur de sélection compare seulement la valeur de la fonction objectif des deux vecteurs en compétition et le meilleur individu est sélectionné pour la population de la prochaine génération.

Au cours des dix dernières années, plusieurs problèmes scientifiques et industriels on été résolus en utilisant les algorithmes à évolution différentielle. Parmi eux, on peut citer : ordonnancement de tâches d'un satellite, registration et traitement d'image, décision multicritère, entraînement des réseaux de neurones, ajustement des fonctions floues, etc. Un algorithme d'évolution différentielle peut être présenté comme suit :

Algorithme 1.5: Algorithme à évolution différentielle

```
début
    Génération aléatoire de la population de vecteurs ;
    Évaluer chaque solution;
    repter
        Appliquer la mutation différentielle ;
        Exécuter un croisement différentiel ;
        Évaluer la nouvelle solution ;
        Appliquer la sélection différentielle ;
    Jusqu'à (le nombre de génération est atteint)
Fin
```

1.10. Méthodes de recherche locale

En informatique, la recherche locale est une métaheuristique utilisée pour résoudre des problèmes dans plusieurs domaines comme l'intelligence artificielle, mathématiques, la recherche opérationnelle, l'ingénierie et la bioinformatique. Elle peut être utilisée sur des problèmes de recherche d'une solution maximisant (minimisant) un critère parmi un ensemble de solutions candidates. Ceci est dû à sa rapidité et à son principe relativement simple. Le principe de la recherche locale est le suivant : l'algorithme débute avec une solution initiale réalisable. Sur cette solution initiale, on applique une série de modifications locales (définissant un voisinage de la solution courante), tant que celles-ci améliorent la qualité de la fonction objectif [Hous et al, 2005]. La figure 1.21 illustre bien le fonctionnement de l'algorithme général des méthodes de recherche locale.

Le passage d'une solution vers une autre se fait grâce à la définition de structure de voisinage qui est un élément très important dans la définition de ce type de méthode. Le voisinage d'une solution est défini en fonction du problème à résoudre. On définit l'espace de recherche comme l'espace dans lequel la recherche locale s'effectue. Cet espace peut correspondre à l'espace des solutions possibles du problème étudié. Habituellement, chaque solution candidate a plus d'une solution voisine, le choix de celle vers laquelle se déplacer est pris en utilisant seulement l'information sur les solutions voisines de la solution courante, d'où le terme de recherche locale [Aarts el al., 1997].

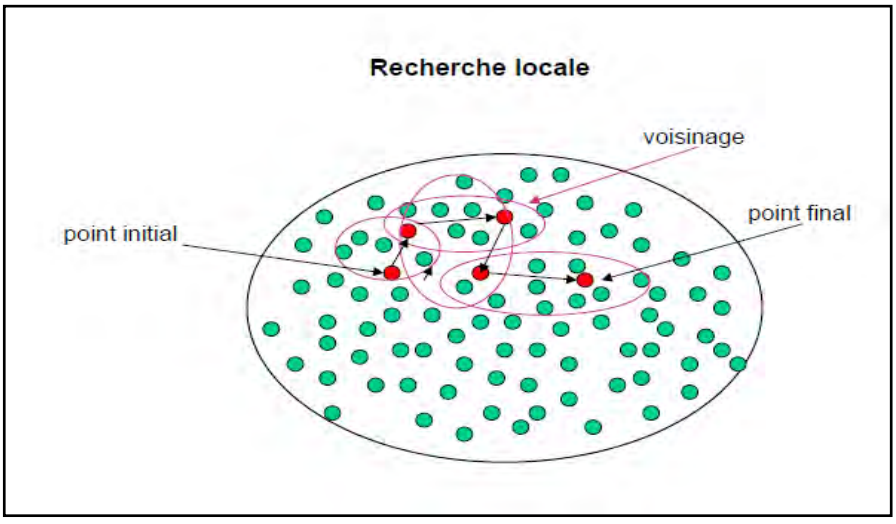


Figure 1.21 – Représentation de la procédure générale de la recherche locale.

D'une manière abstraite, une recherche locale peut se résumer de la façon suivante :

(i) Démarrer avec une solution.

(ii) Améliorer la solution.

(iii) Répéter le processus jusqu'à la satisfaction des critères d'arrêt.

Les méthodes de recherche locales utilisent principalement trois fonctions: une fonction d'évaluation, une fonction de voisinage et une fonction d'amélioration basée sur la fonction d'évaluation. D'un autre côté, le critère d'arrêt de la recherche locale peut être une limite en durée, une autre possibilité est de s'arrêter quand la meilleure solution trouvée par l'algorithme n'a pas été améliorée depuis un nombre donné d'itérations. Par ailleurs, les algorithmes de recherche locale sont des algorithmes sous-optimaux puisque la recherche peut s'arrêter alors que la meilleure solution trouvée par l'algorithme n'est pas la meilleure de l'espace de recherche. Cette situation peut se produire même si l'arrêt est provoqué par l'impossibilité d'améliorer la solution courante car la solution optimale peut se trouver loin du voisinage des solutions parcourues par l'algorithme. En effet, la plupart des méthodes de recherche locale utilisent une seule structure de voisinage ce qui rend très difficile l'optimisation de problèmes contenant plusieurs minima locaux. On trouve dans la littérature un grand nombre de méthodes de recherche locale [Hous et al., 2005], nous évoquerons dans la suite trois méthodes générales, souvent classées dans la catégorie des métaheuristiques qui sont : méthode de la descente, le recuit simulé et la recherche tabou. La procédure générale de la recherche locale est décrite par l'algorithme ci-dessous. Les bases de la recherche locale peuvent être définies ainsi. Soient :

– f () la fonction qu'on l'on souhaite maximiser,
– S la solution courante,
– S* la meilleure solution connue,
– N(S) le voisinage de S

Algorithme 1.6 : Algorithme de recherche locale simple

Étape 1 (initialisation)
a) choisir une solution initial s ∈S.
b) s* ←s (i.e. s* mémorise la meilleure solution trouvée)

Étape 2 *(choix et terminaison)*
a) *choisir* s'∈ N(s)
b) s ←s' (i.e. remplacer s par s')
c) fin si la condition d'arrêt est vérifiée

Étape 3 *(mise à jour)*
a) s* ←s si f(s) < f(s*)
b) aller à l'Étape 2

1.10.1. La Méthode de la Descente (Hill-Climbing)

Cette méthode appelée aussi Hill-Climbing est assez ancien, mais d'une grande simplicité. Elle consiste à se déplacer dans l'espace de recherche en choisissant toujours la meilleure

solution parmi le voisinage de la solution courante [Hous et al., 2005]. On part d'une solution si possible bonne et on balaie l'ensemble des voisins de cette solution. S'il n'existe pas de voisin meilleur que notre solution, on a trouvé un optimum local et on arrête; sinon, on choisit le meilleur des voisins et on recommence (L'algorithme 1.6). La convergence vers un optimum local pouvant être très lente, on peut éventuellement fixer le nombre d'itérations maximum, si on veut limiter le temps d'exécution. Cette méthode a l'inconvénient de rester bloquée dans un optimum local. En effet, une fois un optimum local trouvé, on s'arrête, même si ce n'est pas l'optimum global. Selon le paysage des solutions, l'optimum local peut être très bon ou très mauvais par rapport à l'optimum global. Si la solution de départ est donnée par une heuristique déterministe, l'algorithme sera déterministe. Si elle est tirée au hasard, on a un algorithme non déterministe et donc plusieurs exécutions différentes sur la même instance pourront donner des solutions différentes et de qualités différentes.

Il est tout à fait clair que dans ce type de méthodes, la notion de voisinage est primordiale. Si les voisins sont très nombreux, on a de fortes chances de trouver l'optimum global. Malgré cela, visiter un voisinage peut être très long par ce qu'on visitera une grande partie de l'espace des solutions. D'un autre côté, si le voisinage est très restreint, on risque fort de rester bloqué dans un optimum local de mauvaise qualité. Par conséquent, le choix de la notion de voisinage est un compromis entre efficacité et qualité.

Algorithme 1.7 : Schéma général du Hill-Climbing

```
Début
Générer et évaluer une solution initiale s
 Tant que La condition d'arrêt n'est pas vérifiée faire
  Modifier s pour obtenir s' et évaluer s'
  si (s' est meilleure que s) alors
   Remplacer s par s'
  finsi
 Fin tant que
 retourner s
Fin
```

On distingue différents types de descente en fonction de la stratégie de génération de la solution de départ et du parcours du voisinage : la descente déterministe, la descente stochastique et la descente vers le premier meilleur. On peut par exemple remarquer que dans la formulation précédente, il n'est pas mentionné que l'aléatoire est mis en jeu pour la génération de la solution initiale ou sa modification. L'algorithme 1.7 décrit la version

stochastique *Random Hill-Climbing* (RHC). Il faut alors définir un pas de recherche qui détermine la taille du voisinage pouvant être exploré.

Algorithme 1.8 : Schéma général du Random Hill-Climbing

```
Début
  Générer aléatoirement une solution initiale s et l'évaluer
  tant que La condition d'arrêt n'est pas vérifiée faire
    Modifier aléatoirement s pour obtenir s' et évaluer s'
    si (s' est meilleure que s) alors
      Remplacer s par s'
    fin si
  fin tantque
  retourner s
Fin
```

La méthode hill-climbing possède plusieurs avantages. C'est une méthode intuitive, simple à implémenter par rapport à d'autres méthodes de recherche locale. Typiquement, Il n'existe pas de notion d'arbre de recherche ni de retour en arrière. La méthode hill-climbing donne souvent de bonnes solutions. Cependant, cette méthode présente quelques limitations comme le problème du minimum local et la lenteur de la méthode.

1.10.2. Le recuit simulé

La méthode du Recuit Simulé (RS) est une technique de recherche locale inspirée d'un processus utilisé en métallurgie [Aarts et al, 1989]. Ce processus alterne des cycles de refroidissement lent et de réchauffage ou de recuit qui tendent à minimiser l'énergie du matériau. Le recuit simulé s'appuie sur l'algorithme de Metropolis-Hastings, qui permet de décrire l'évolution d'un système thermodynamique. Par analogie avec le processus physique, la fonction à minimiser deviendra l'énergie E du système. On introduit également un paramètre fictif, la température T du système.

L'algorithme du recuit simulé part d'une solution donnée, et la modifie itérativement jusqu'à le refroidissement du système. Les solutions trouvées peuvent améliorer le critère que l'on cherche à optimiser, on dit alors qu'on a fait baisser l'énergie du système, comme elles peuvent les dégrader. Si on accepte une solution améliorant le critère, on tend ainsi à chercher l'optimum dans le voisinage de la solution de départ. Contrairement autres méthodes de recherche locale, le recuit simulé peut accepter des solutions de qualité moindre en fonction

de la dégradation de la solution considérée. Le schéma général de l'algorithme RS est décrit par l'algorithme suivant :

Algorithme 1.9 : Schéma général du Recuit simulé

```
Engendrer une configuration initiale S0 de S ; S := S0
Initialiser T en fonction du schéma de refroidissement
Répéter
  -Engendrer un voisin aléatoire S' de S
  -Calculer Δ = f(S') – f(S)
  -Si CritMetropolis (Δ, T), alors S : = S'
  -Mettre T à jour en fonction du schéma de refroidissement
Jusqu'à <condition fin>
Retourner la meilleure configuration trouvée
```

L'acceptation d'une mauvaise solution permet alors d'explorer une plus grande partie de l'espace de solution et tend à éviter de s'enfermer trop vite dans la recherche d'un optimum local. Afin de fixer le niveau de dégradation acceptable à une itération donnée le paramètre de température T est utilisé. Ce paramètre décroît au cours du temps. Néanmoins, Une difficulté spécifique à l'algorithme du recuit simulé est l'ajustement du paramètre de la température T. En effet, Un refroidissement trop rapide mènerait vers un optimum local pouvant être de très mauvaise qualité. Un refroidissement trop lent serait très coûteux en temps de calcul. Le nouvel état est accepté si l'énergie du système diminue, sinon il est accepté avec une probabilité [Kirkpatrick et al., 1983]. Le recuit simulé possède plusieurs applications dans des domaines différents : Traitement d'images, Problème d'ordonnancement, Bioinformatique, etc. Le recuit simulé donne généralement des solutions de bonnes qualités. C'est une méthode générale et facile à programmer, Souple d'emploi (de nouvelles contraintes peuvent être facilement incorporées). Cependant, les inconvénients principaux sont le nombre important de paramètres à ajuster et le temps de calcul excessif dans certaines applications.

1.10.3. La recherche Tabou (tabu search)

La recherche tabou est une méthode de recherche locale avancée qui fait appel à un ensemble de règles et de mécanismes généraux pour guider la recherche de manière intelligente [Glover et al., 1997]. Le principe de la recherche tabou est d'explorer le voisinage et choisir la position dans ce voisinage qui minimise la fonction objectif à partir d'une position donnée. Lorsque tous les points du voisinage ont une valeur plus élevée, cette opération peut conduire à augmenter la valeur de la fonction. On peut ainsi repérer les minima locaux. Pour

éviter à l'étape suivante de retomber dans un minima local, il faut stocker en mémoire les dernières positions explorées d'où le nom tabou. Les positions sont enregistrées dans une pile FIFO de taille paramétrable. Toutefois, la mémorisation des solutions complètes rend la gestion de la liste Tabou trop coûteuse en temps de calcul et en place mémoire; particulièrement si le nombre de variables est très élevé. On peut réussir à palier ce problème en ne stockant que les mouvements et la valeur de la fonction à minimiser. Il a été démontré pour la recherche tabou que la convergence existe dans des conditions strictes rarement mis en pratique. On notera qu'il existe de nombreuses variantes aussi bien au niveau de la définition du voisinage que de la façon de gérer la mémoire [Chelouah et al, 2000]. La recherche tabou possède plusieurs avantages. En effet, elle est une méthode rapide, facile à implémenter, donne souvent de bonnes solutions, et possède un fonctionnement intuitif. Cependant, la recherche tabou possède certaines limitations comme le manque de modélisation mathématiques. En effet, chaque cas est différent et il faut donc adapter la recherche à chaque problème particulier. D"un autre le grand nombre de paramètre à régler rend l"utilisation de la méthode tabou difficile. Finalement, la recherche tabou a été appliquée avec succès pour résoudre de nombreux problèmes difficiles d'optimisation combinatoire : problèmes de routage de véhicules, problèmes d'affectation quadratique, problèmes d'ordonnancement, problèmes de coloration de graphes, etc. L"algorithme général de la recherche tabou est le suivant [Rodriguez-Tello et al, 2005] :

Algorithme 1.10 : Schéma général de la recherche tabou

```
Début
  S = solution initiale ;
  Best = S ;
   Répéter
     Sbest = meilleur voisin de S ; //le meilleur voisin est choisi
      si (fitness(Sbest) < fitness (Best)) Alors
         Best = Sbest ;
     Mise à jour de la liste Tabou
     S = Sbest ;
   Jusqu'à(critère de fin est vérifié)
  Retourner Best ;
Fin
```

1.10.4. Discussion des méthodes de recherche locale

Bien que les méthodes de recherche locale ont démontré leurs efficacités dans plusieurs problèmes d"optimisation, la difficulté essentielle liée à l"utilisation de ces méthodes consiste

à réussir une exploration de l'espace de recherche sans stagner dans des minima locaux. La recherche se trouve dans un minimum local lorsque plus aucun des voisins de l'affectation courante ne l'améliore, par exemple, dans la figure 1.22, on observe plusieurs minima locaux. Pour sortir de ces minima, les algorithmes doivent avoir la capacité de détériorer l'affectation courante afin de l'améliorer ultérieurement en atteignant l'optimum global. Cette technique est connue sous le nom du *chemin aléatoire* (Random Walk). Elle consiste à effectuer, de temps en temps, un mouvement aléatoire pour diversifier la recherche et ainsi espérer se rapprocher de la bonne solution. Le fait d'accepter la détérioration de l'affectation courante permet de sortir de certains minima locaux mais peut induire un allongement du temps d'exécution. On trouve une technique similaire à celle-ci dans l'algorithme de recuit simulé [Kirkpatrick et al., 1983]. Une autre technique couramment utilisée est celle de la *relance* (Multiple Start). Elle consiste à répéter une nouvelle recherche à partir d'une autre solution initiale appartenant à une autre région de l'espace de recherche. Cette technique peut être réitérée plusieurs fois dans l'espérance d'atteindre différentes zones prometteuses de l'espace de recherche [Barichard, 2003].

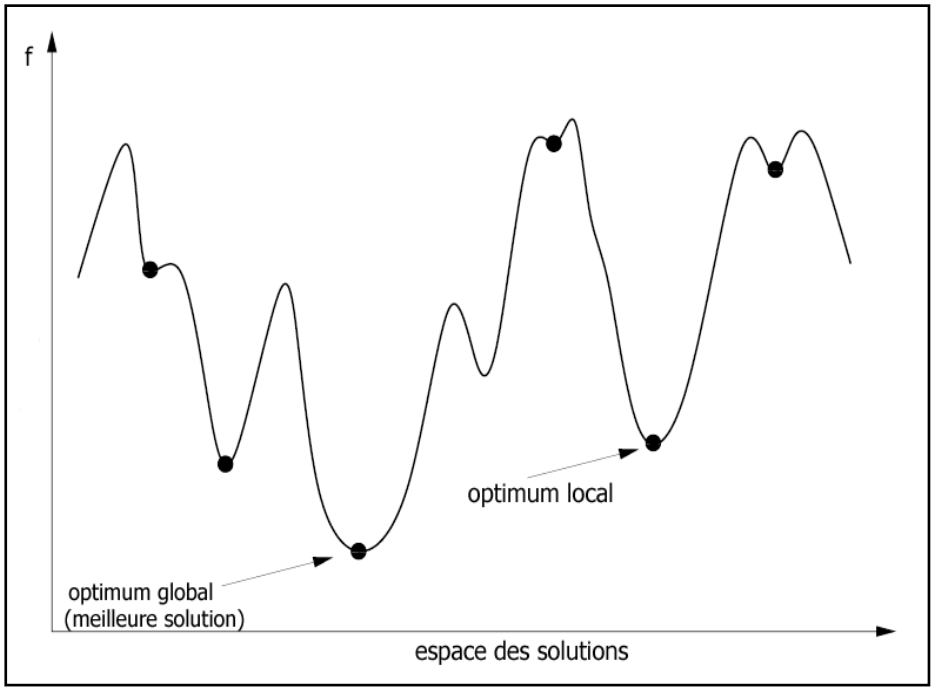


Figure 1.22– Le problème de minimum local dans la recherche locale.

1.11. Méthodes hybrides

L'hybridation est une tendance observée dans de nombreux travaux réalisés sur les métaheuristiques ces dix dernières années. Elle exploite la puissance de plusieurs algorithmes, et les combine en un seul méta-algorithme [Sbihi, 2003]. Il existe plusieurs manières pour faire cette hybridation.

Une des techniques les plus populaires d'hybridation concerne l'utilisation de métaheuristiques à base d'une solution unique avec des métaheuristiques à population. La plupart des applications réussies des métaheuristiques à population sont complétées par une phase de recherche locale [Bontoux et al., 2008], afin d'augmenter leurs capacités d'intensification. Les individus d'une population tentent d'explorer l'espace de solutions afin de trouver les zones prometteuses, lesquelles sont ensuite explorées plus en détail par des méthodes de recherche locale comme la recherche tabou ou le hill-climbing, ou par un algorithme de type 2-opt. Il est à noter également que les deux types de recherche, globale et locale, peuvent être lancées de façon asynchrone, sur des processeurs différents afin d'augmenter la vitesse d'exécution de l'algorithme hybride [Galinier et al., 1999].

Une deuxième manière d'hybrider consiste à lancer en parallèle l'exécution de la même métaheuristique, voire même plusieurs fois la même métaheuristique mais avec des paramètres différents. Ces processus parallèles communiquent entre eux régulièrement pour échanger de l'information sur leurs résultats partiels. Presque toutes les métaheuristiques classiques ont donné naissance à des versions parallèles plus ou moins fidèles à leur origine.

Enfin une troisième forme d'hybridation combine les métaheuristiques avec des méthodes exactes [Haouari et al, 2003]. Une méthode exacte peut être utilisée pour la génération du meilleur voisin d'une solution. On peut par exemple utiliser la méthode branch-and-bound sur une partie du problème de petite taille pour générer des solutions de bonnes qualités. Par ailleurs, une métaheuristique peut être utilisée pour fournir des bornes à une méthode exacte de type branch-and-bound[Pessan et al., 2007].

Les algorithmes hybrides sont sans aucun doute parmi les méthodes les plus puissantes. Malheureusement, les temps de calcul nécessaires peuvent devenir prohibitifs à cause du nombre d'individus manipulés dans la population. Une voie pour résoudre ce problème est la parallélisation de ces algorithmes sur des machines parallèles ou sur des systèmes distribués. En effet, le coût de calcul des méthodes hybrides peut être supporté avec le recours aux grilles de calcul [Melab, 2005].

1.12. Conclusion

Dans ce chapitre, nous avons essayé de rassembler un état de l'art de quelques méthodes d'optimisation combinatoire en allant des méthodes exactes aux méthodes métaheuristiques. Nous avons pu constater au fur et à mesure d'un court état de l'art pour chaque méthode, qu'elles sont adaptables à un très grand nombre de problèmes combinatoires. D'abord, nous avons présenté avec un peu de détail le fonctionnement de l'algorithme référence branch-and-bound représentant les méthodes exactes. Dans une deuxième partie, On a fait une présentation détaillée de deux algorithmes à population : algorithmes génétiques et les systèmes immunitaires artificiels. Ces deux méthodes d'optimisation globales constituent des outils efficaces pour une classe de problèmes très large. Ensuite, nous avons présenté les méthodes de recherche locale qui se caractérisent par la simplicité, mais elles sont piégées de premier minimum local trouvé. Et enfin, nous avons évoqué la tendance actuelle vers les méthodes hybrides. Le mode d'hybridation concerne aussi bien la combinaison entre les méthodes de voisinage et l'approche d'évolution que les méthodes approchées et exactes. L'idée essentielle de cette hybridation consiste à exploiter pleinement les puissances de chaque méthode afin de créer des métaheuristiques efficaces et puissantes.

Dans le chapitre suivant on va faire une introduction à l'informatique quantique et ses principes de base. Nous allons parler également sur les algorithmes évolutionnaire quantiques.

CHAPITRE 2

Principes de base de l'informatique quantique

"No exponential is forever. Your job is to delay forever."

– Andrew Gordon Moore 2003.

L'Informatique quantique est un domaine récent en informatique qui se base sur les principes de la mécanique quantique. Il essaie de donner des solutions à des problèmes non encore résolus par l'informatique classique. En effet, l'informatique quantique offre des capacités de traitement et de stockage exponentielles grâce à des principes de la mécanique quantique tels que la superposition, l'enchevêtrement, l'interférence, etc. Cependant, l'utilisation de ce nouveau paradigme est dépendante de la réalisation pratique des machines quantiques. Mais cela n'a pas empêché les chercheurs de combiner les algorithmes classiques et les principes de l'informatique quantique afin de tirer profit des capacités de ce dernier. Contrairement aux algorithmes quantiques purs, les algorithmes inspirés du quantique tels que les Algorithmes Evolutionnaires Quantiques AEQs ne nécessitent pas la présence de machines quantiques. Les algorithmes inspirés du quantique ont démontré leur efficacité dans plusieurs problèmes d'optimisation combinatoire.

Ce chapitre est constitué de deux parties, la première partie contient les fondements de base de l'informatique quantique et la deuxième partie présente les principes des algorithmes inspirés du quantique.

Sommaire

2.1. Introduction

Dans la vie courante, nous sommes fréquemment confrontés à des problèmes de nature combinatoire. Les problèmes d'optimisation combinatoire apparaissent dans plusieurs domaines très variés des sciences et de l'ingénierie. Malheureusement, La plupart de ces problèmes sont caractérisés par des grandes complexités spatiotemporelles et nécessitent des capacités de traitement et de stockage énorme. Pour ces raisons, les chercheurs essayent de trouver d'autres architectures qui offrent en même temps des ressources de stockage et de traitement puissantes. En effet, plusieurs nouveaux paradigmes informatiques ont immergé ces dernières années. Ces nouveaux paradigmes se basent sur différents concepts comme la biologie ou la physique. Parmi les nouveaux paradigmes informatiques qui ont suscité beaucoup d'intérêt, on distingue l'informatique biomoléculaire. Ce paradigme utilise les caractéristiques de l'ADN pour résoudre les problèmes NP-difficiles. Un autre paradigme très intéressant est l'informatique quantique[Le Bellac ,2003]. L'informatique quantique utilise les spécificités de la mécanique quantique pour le traitement et la transformation de l'information.

L'informatique quantique est un domaine en émergence faisant appel à plusieurs spécialités : physique, génie, chimie, informatique et mathématiques. L'objectif visé par cette intégration de connaissances est la réalisation d'un ordinateur quantique. L'utilisation d'un tel ordinateur est pour effectuer certains calculs beaucoup plus rapidement qu'avec un ordinateur fonctionnant de façon standard (ordinateur classique). Cette accélération est rendue possible en tirant profit des phénomènes quantiques tels que les superpositions d'états, l'enchevêtrement et l'interférence. Le bit quantique (qubit) est l'unité d'information de base en informatique quantique. Un qubit peut être dans l'état 0, 1, ou une superposition de 1 et 0 ce qui permet de représenter un nombre exponentiel d'états avec un ensemble restreint de qubits (pour n qubits on peut représenter 2^n états). Une autre caractéristique de la mécanique quantique est la linéarité des opérations. Cette caractéristique permet en informatique quantique un parallélisme exponentiel. En effet, une opération est appliquée en parallèle sur tous les états présents en superpositions [Shor, 1998].

Les recherches dans ce domaine sont accélérées aussi bien sur le plan pratique que théorique. Théoriquement, plusieurs algorithmes quantiques ont vu le jour en essayant de démontrer la puissance de l'informatique quantique. Parmi les algorithmes qui ont mis l'informatique en scène, l'algorithme de factorisation des nombres de Peter Shor [Shor, 1994]. Cet algorithme a une complexité polynomiale or le meilleur algorithme classique a une complexité exponentielle. Un autre algorithme très important est l'algorithme de Grover [Grover, 1996] pour la recherche d'une donnée dans une liste désordonnée de données. Cet algorithme trouve l'élément recherché dans un temps quadratique. Du coté pratique, IBM a pu construire une machine quantique à 7 qubits. Cependant, la construction d'une machine contenant des centaines voir des milliers de qubits n'est pas une chose facile à cause des

erreurs qui peuvent surgir dues à la décohérence du système quantique. Cette réalisation nécessite des outils théoriques et technologiques plus sophistiqués. En espérant voir dans les années à venir la conception à grande échelle d"un ordinateur quantique.

Mais la recherche en informatique quantique ne s'est pas arrêtée à ce niveau à cause de non disponibilité des machines quantiques. Plusieurs chercheurs s"intéressent à la combinaison entre des algorithmes classiques et les principes de la mécanique quantique. L"avantage de ces algorithmes est qu"ils ne nécessitent pas la présence d"une machine quantique. Plusieurs algorithmes ont été créés en se basant sur cette idée comme les Algorithmes Quantiques Evolutionnaires AQEs [Han et al ,2001], les Algorithmes Quantique Génétiques AQG [Han et al, 00], Algorithme de Recherche Dispersé Quantique [layeb et al, 2008f], etc.

Ce chapitre est divisé en deux volets. Dans le premier volet on va présenter les fondements de l'informatique quantique. Des circuits quantiques et un algorithme quantique démonstratif sont présentés. Dans le deuxième volet, on va présenter les algorithmes inspirés du quantique. Des définitions et des exemples sont également cités.

2.2. Notions de base

Définition 2.1 (l'Informatique Quantique): L'informatique quantique est le sous-domaine de l'informatique qui traite des ordinateurs quantiques utilisant des phénomènes de la mécanique quantique, par opposition à ceux de l'électricité exclusivement, pour l'informatique dite «classique». Les phénomènes quantiques utiles sont l'enchevêtrement quantique et la superposition. Les opérations ne sont plus basées sur la manipulation de bits, mais de qubits [PHQ, 2009]**.**

Définition 2.2 (La théorie Quantique): La théorie quantique a été introduite suite à la non capacité de la mécanique classique d'interpréter certains phénomènes physiques, certains phénomènes ne semblent pas de nature continue comme par exemple, la relation de proportionnalité entre la force et l'accélération. Cette théorie est basée sur un formalisme mathématique solide [Preskil, 1998] [Jaeger, 2006] [Shor, 2004]. La mécanique quantique est fondée sur les quatre postulats suivants:

- ***Etat (states)*** **:** Un système physique dans la mécanique quantique est complètement décrit par un vecteur d'état dans l"espace de Hilbert.
- ***Observables (observables)*****:** C'est la propriété qu'un système physique peut être mesuré Cependant dans le cas de la mécanique, les propriétés physiques sont représentées mathématiquement par des opérateurs hermétiques,
- ***Mesure (measurement):*** en mécanique quantique, la mesure projette le vecteur d"état du système en un nouveau vecteur d"état.

- ***La dynamique (dynamics):*** L'évolution au cours du temps d'un système quantique isolé est décrite par une transformation unitaire.

Afin de mieux comprendre les phénomènes quantiques, nous allons présenter l'expérience de polarisation des photons.

2.2.1. L'expérience de la polarisation du photon

Cette expérience démontre certains principes de la mécanique quantique à travers les photons et leur polarisation [Rieffel et al, 2000]. Les photons sont les seules particules que nous pouvons directement observer. L'expérience est simple et peut être établie avec un équipement minimal. On a besoin d'une source puissante de lumière comme un pointeur laser et de trois polaroïds (filtres de polarisation). Un faisceau de lumière rayonne sur un écran de projection. Les filtres A, B et C sont polarisés respectivement horizontalement, à 45^o et verticalement. Ils peuvent être placés en intersection avec la lumière. L'expérience est composée des étapes suivantes :

Etape 1 : On insère le filtre *A* et en supposant que la lumière entrante est aléatoirement polarisée. Donc, l'intensité de la lumière sortante va être la moitié de l'intensité de l'entrante. Les photons sortants sont maintenant polarisés horizontalement (figure 2.1).

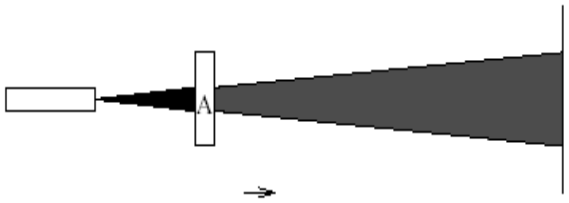


Figure 2.1 – L'insertion du filtre A.

La fonction du filtre A ne peut pas être expliquée telle que A est une passoire qui seulement laisse passer les photons déjà polarisés horizontalement.

Etape 2 : Quand le filtre C est inséré (figure 2.2), l'intensité de la lumière sortante est zéro. Le filtre C ne laisse passer aucun des photons horizontalement polarisés.

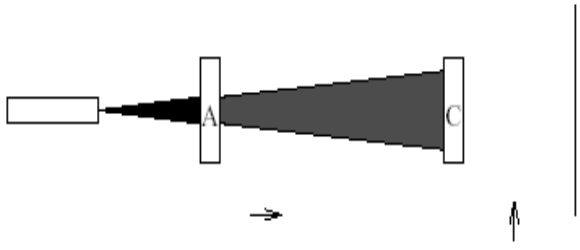


Figure 2.2 – L'ajout du filtre C.

Etape 3 : Finalement, Après l'insertion du filtre B entre A et C (figure 2.3), une petite quantité de la lumière sera visible sur l'écran, exactement un huitième (1/8) de la lumière originale.

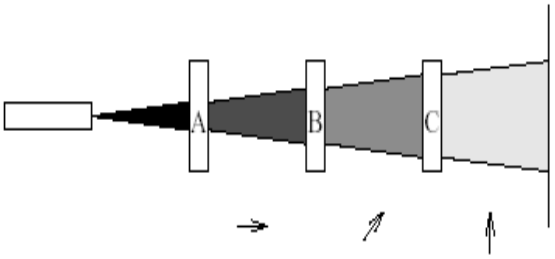


Figure 2.3 – L'ajout du filtre B.

Cette expérience a montré un paradoxe. L'expérience classique suggère que l'ajout d'un filtre doit diminuer seulement la quantité de la lumière or dans cette expérience, la quantité de la lumière a augmenté. On peut modéliser l'état de polarisation du photon par un vecteur pointé dans la direction appropriée. Toute polarisation arbitraire peut être exprimée comme une combinaison linéaire : $a|\uparrow\rangle + b|\rightarrow\rangle$ des deux vecteurs de base :

$|\uparrow\rangle$: Polarisation verticale.

$|\rightarrow\rangle$: Polarisation horizontale.

Le vecteur d'état sera un vecteur unitaire : $|a|^2+|b|^2 = 1$, parce que nous sommes intéressés seulement par la direction de la polarisation, Donc généralement, la polarisation d'un photon peut être exprimée comme : $a|\uparrow\rangle+b|\rightarrow\rangle$. Où a et b sont deux nombres complexes tels que $|a|^2+|b|^2 = 1$.

Le principe de mesure de la mécanique quantique révèle que tout dispositif de mesure d'un système à deux dimensions a un ensemble associé de vecteurs de base. La mesure d'un état est la projection de cet état à l'un des vecteurs du dispositif de mesure associé. La probabilité qu'un état est mesuré selon un vecteur de base $|u\rangle$ est égale au carré de l'amplitude du composant de la projection de l'état original sur le vecteur de base $|u\rangle$ (figure 2.4). Sachant qu'il peut exister plusieurs dispositifs de mesure, ce qui entraîne des résultats de mesure différents. Si on a par exemple la mesure de l'état quantique suivant :

$\psi = a|\uparrow\rangle + b|\rightarrow\rangle$ donne $|\uparrow\rangle$, alors l'état ψ prend définitivement la valeur $|\uparrow\rangle$. C'est-à-dire une nouvelle mesure de cet état va donner certainement $|\uparrow\rangle$ avec la probabilité 1. En mécanique quantique on ne peut pas prédire l'état original d'un état après l'opération de mesure.

Donc, d'après la mécanique quantique, les filtres jouent le rôle des dispositifs de mesure. Le filtre ne laisse passer que les photons qui ne sont pas orthogonaux à sa polarisation. Par exemple, le filtre A mesure la polarisation des photons relativement au vecteur de base $|\rightarrow\rangle$, correspondant à sa polarisation. Cependant, tous ceux qui sont reflétés par le filtre C ont une polarisation $|\uparrow\rangle$. Quant à l'écran, il mesure l'état quantique des photons relativement à sa base. Finalement, le filtre B va mesurer les photons relativement à la base : $\{1/\sqrt{2}\ (|\uparrow\rangle+|\rightarrow\rangle),\ 1/\sqrt{2}\ (|\uparrow\rangle-|\rightarrow\rangle)\}$. C'est ce qu'on écrit comme : $\{|\nearrow\rangle, |\nwarrow\rangle\}$ tel que

$$|\rightarrow\rangle = 1/\sqrt{2}\ (|\nearrow\rangle-|\nwarrow\rangle).$$

$$|\uparrow\rangle = 1/\sqrt{2}\ (|\nearrow\rangle+|\nwarrow\rangle).$$

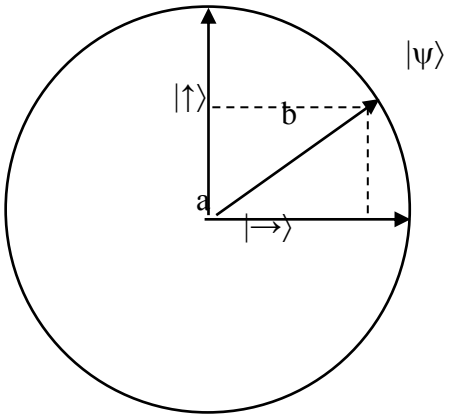


Figure 2.4 – Mesure : Une projection sur la base.

Ensuite, les photons qui sont mesurés selon $|\nearrow\rangle$ passent à travers le filtre. Les photons qui passent par A avec l'état $|\rightarrow\rangle$ vont être mesurés par B comme $|\nearrow\rangle$ avec une probabilité de ½. Par conséquence 50 % des photons passés par A vont passer par B et dans l'état $|\nearrow\rangle$. Idem, les photons seront mesurés par le filtre C comme $|\uparrow\rangle$ avec une probabilité de 1/2. Donc seulement 1/8 (un huitième) des photons originaux vont passer à travers la séquence des filtres : A, B et C (figure 2.5).

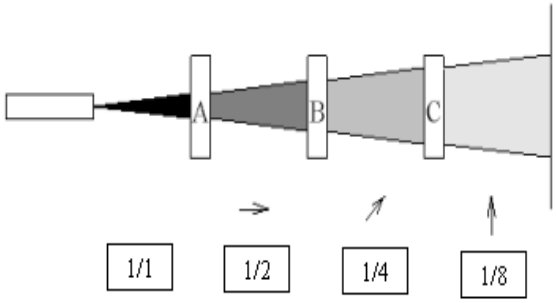


Figure 2.5 – Le pourcentage de la lumière après chaque ajout des trois filtres.

2.2.2. Espaces d'états et notation Bra/Ket

Un espace d'états d'un système quantique est constitué des positions, moments, polarisation, spins…etc, des différentes particules. Il est modélisé par un espace de Hilbert des fonctions des ondes. Dans l'informatique quantique, nous avons besoin seulement des systèmes quantiques finis et il est suffisant de considérer des espaces vectoriels complexes de dimensions finies avec un produit interne qui sont franchis par des fonctions abstraites d'onde comme $|\rightarrow>$. Les espaces d'états quantiques et les transformations actant sur ces états peuvent être décrits en termes de vecteurs et de matrices, ou en utilisant la notation Bra/Ket inventée par Dirac [Dirac, 1959]. Une Kets $|x\rangle$ dénote un vecteur colonne et elle est spécifiquement utilisée pour décrire les états quantiques. Cependant, le "Bra" $\langle x|$: Dénote la transformée conjuguée de $|x\rangle$. Par exemple, Les orthogonaux $\{|0\rangle,|1\rangle\}$ peuvent être exprimés comme $\{(1,0)^T, (0,1)^T\}$. Donc, toute combinaison linéaire complexe de $|0\rangle$ et $|1\rangle$, $a|0\rangle + b|1\rangle$ peut être écrite $(a,b)^T$. le choix de l'ordre des vecteurs de base est arbitraire. Par exemple, on peut représenter $|0\rangle$: $(0,1)^T$ et $|1\rangle$: $(1,0)^T$.

La combinaison de $\langle x|$ et $|y\rangle$ comme dans $\langle x\|y\rangle(\langle x|y\rangle)$, dénote le produit interne des deux vecteurs. Vu que $|0\rangle$ est un vecteur unitaire nous avons : $\langle 0|0\rangle = 1$, et comme $|0\rangle$ et $|1\rangle$ sont orthogonaux nous avons : $\langle 0|1\rangle = 0$. La notation $|x\rangle\langle y|$ est le produit externe de $|x\rangle$ et $\langle y|$. Par exemple, $|0\rangle\langle 1|$ est la transformation qui change $|1\rangle$ à $|0\rangle$ et $|0\rangle$ à $(0,0)^T$.

$$|0\rangle\langle 1|\ |1\rangle = |0\rangle\langle 1|1\rangle = |0\rangle$$

$$|0\rangle\langle 1|\ |0\rangle = |0\rangle\langle 1|0\rangle = 0\ |0\rangle = \begin{pmatrix} 0 \\ 0 \end{pmatrix}.$$

De la même façon, $|0\rangle\langle 1|$ peut être écrite dans une forme matricielle où :

$|0\rangle = (1,0)^T, \langle 0| = (1,0)$, $|1\rangle = (0,1)^T$, $\langle 1| = (0,1)$.

Ce qui donne donc, le produit suivant: $|0\rangle\langle 1| = (1,0)^T\ (0,1) = \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix}$.

Cette notation nous donne une façon adéquate pour indiquer les transformations dans les états quantiques, en termes de ce qui parvient aux vecteurs de base. Par exemple : la transformation qui échange $|0\rangle$ et $|1\rangle$ est donnée par : $X = |0\rangle\langle 1| + |1\rangle\langle 0|$. Mais la plupart des auteurs en informatiques quantiques préfèrent la notation la plus intuitive :

$$X: \quad |0\rangle \rightarrow |1\rangle.$$
$$|1\rangle \rightarrow |0\rangle.$$

Ceci spécifie explicitement le résultat de la transformation dans les vecteurs de base [Rieffel et al, 2000].

2.2.3. Le qubit

Une idée géniale est déduite de cette expérience. On peut utiliser la polarisation des photons pour transmettre de l'information. En comparant avec le paradigme classique, on décide, tout à fait arbitrairement, d'attribuer la valeur 1 du bit à un photon polarisé Suivant Ox($|\rightarrow\rangle$) et la valeur 0 un photon polarisé suivant Oy ($|\uparrow\rangle$), d'où vient l'idée de l'informatique quantique.

L'information quantique est la théorie de l'utilisation des spécificités de la physique quantique pour le traitement et la transmission de l'information. L'unité fondamentale de l'information quantique est le bit quantique ou, plus simplement qubit (de l'anglais „quantum bit"). Ce terme a été introduit par B. Schumacher [Schumacher, 1995]. Comme sa contrepartie classique, un qubit peut prendre deux valeur dénotées |0> et |1>. Un qubit vit donc dans un espace d'Hilbert à deux dimensions et peut être dans une superposition de ces deux états orthogonaux a|0>+ b|1>. La base {|0>, |1>} correspond aux deux polarisations du photon $|\rightarrow\rangle$ et $|\uparrow\rangle$. Il est intéressant de mentionner que, selon le théorème de Holevo, seulement un bit classique peut être extrait d'un bit quantique. Lorsque l'on s'intéresse à un seul qubit, le vecteur de Bloch b(t) sur la sphère de Bloch est une représentation utile (figure 2.6).

Sous leur apparente ressemblance les bits classiques et quantiques offrent des possibilités totalement différentes dues aux étranges propriétés des qubits que nous décrirons dans la table 2.1 [Blais, 02].

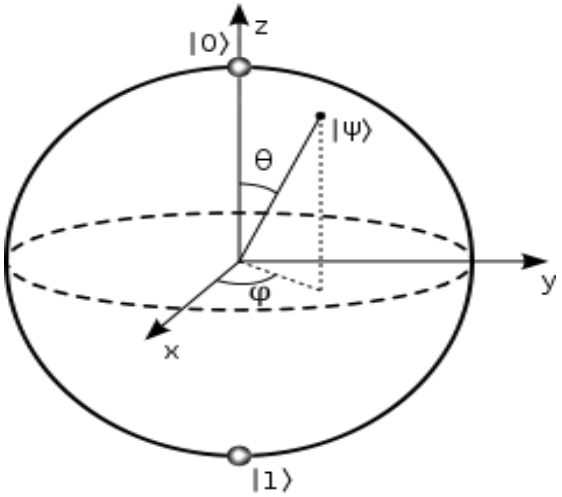


Figure 2.6 – La sphère de Bloch.

Table 2.1. Comparaison entre le bit classique et le qubit.

classique	quantique
Un bit classique a toujours une valeur définie.	On ne peut pas savoir la valeur de l'état d'un qubit non mesuré.
Un bit peut avoir exclusivement 1 ou 0.	Un qubit peut être dans une superposition de 1 et 0 simultanément.
Le bit peut être copié sans être détérioré.	Impossible de copier un qubit dans un état inconnu.
Un bit peut être lu sans affecter sa valeur.	Lire un qubit dans une superposition changera sa valeur.

2.2.4. Représentation d'un état quantique

Il existe plusieurs notations pour représenter un état quantique qu'on peut résumer comme suit [Rieffel et al., 2000] :

- ***Notation de Dirac***

Les physiciens en théorie quantique utilisent souvent une notation pratique inventée par Dirac et qui diffère quelque peu de la notation mathématique usuelle. Un qubit Q est représenté par une combinaison de $|0\rangle$ et $|1\rangle$ comme suit :

$$|Q\rangle = \alpha|0\rangle + \beta|1\rangle \tag{2.1}$$

Tel que :

$$|\alpha|^2 + |\beta|^2 = 1 \tag{2.2}$$

Le qubit peut avoir l'état 0 (1) avec une probabilité $\alpha^2 (\beta^2)$ respectivement.

- ***Notation vectorielle***

Un qubit peut être représenté par une forme matricielle par :

$$0 \rightarrow \begin{pmatrix} 1 \\ 0 \end{pmatrix} \qquad 1 \rightarrow \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

- ***Notation matricielle***

On utilise la matrice de densité pour représenter un qubit :

$\begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix}$: Pour représenter l'état |0>.

$\begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix}$: Pour représenter l'état |1>.

2.2.5. Registre quantique

Une notion très importante en informatique quantique est le registre quantique. Il est constitué d'un ensemble de qubits, c'est une superposition arbitraire de n qubits. Contrairement à un registre classique de n bits qui représente à instant précis une seule valeur parmi les 2^n valeurs possibles, un registre quantique peut contenir les 2^n valeurs possibles en même temps grâce au principe de la superposition d'états [Simon, 1994]. La table 2.2 montre l'équivalence en bits classiques pour avoir la même puissance d'un ensemble de qubits. Un registre est représenté par la notation suivante :

$$\Psi = \sum_{x=0}^{2^k-1} C_x \left| X \right\rangle \tag{2.3}$$

Les amplitudes C_x satisfont la propriété :

$$\sum_{x=0}^{2^k-1} \left| C_X \right|^2 = 1 \tag{2.4}$$

Table 2.2. L'équivalence en bits classique pour avoir la même puissance d'un registre quantique

Nombre de qubits	**Nombre de bits classiques nécessaires pour obtenir la même puissance**
10	1000
20	1 000 000
30	1 000 000 000 000
300	Plus que le nombre d'atomes dans l'univers!

2.2.6. Les fondements de l'informatique quantique

L'informatique quantique est basée principalement sur les cinq principes suivants inspirés de la mécanique quantique : la superposition d'états, l'Interférence, L'enchevêtrement, le non déterminisme et la non duplication [Preskill, 1998].

2.2.6.1 La superposition

Un qubit peut être dans l'état 0 comme il peut être dans l'autre état 1. Mais, il peut être également dans les deux états en même temps. Ce phénomène est appelé la superposition d'état. Ce principe offre donc des capacités de traitement et de stockage exponentielles. En effet, un registre quantique de taille n peut contenir à la fois 2^n valeurs différentes.

2.2.6.2 L'interférence

L'interférence est l'une des caractéristiques des systèmes ondulatoires. D'après les postulats de la mécanique quantique, Les particules quantiques ont une double nature ; corpusculaire et ondulatoire. Donc le phénomène d'interférence est applicable dans la mécanique quantique. Il a comme rôle d'augmenter (interférence constructive) ou diminuer (interférence destructive) l'amplitude d'un état et par conséquence sa probabilité d'être observé. L'interférence est très importante en calcul quantique. Elle permet d'augmenter la probabilité d'avoir les résultats escomptés. Prenant l'exemple suivant :

Soit un qubit décrit par la formule vectorielle suivante : $|\psi\rangle = 1/\sqrt{5}\begin{pmatrix}2\\1\end{pmatrix}$.

Et soit l'opération O qui fait l'interférence décrite par la matrice suivante : $O = 1/\sqrt{2}\begin{pmatrix}1 & 1\\1 & -1\end{pmatrix}$. L'application de O sur le qubit donne le résultat suivant :

$$O|\psi\rangle = \frac{1}{\sqrt{2}}\begin{pmatrix}1 & 1\\1 & -1\end{pmatrix}\frac{1}{\sqrt{5}}\begin{pmatrix}2\\1\end{pmatrix} = \frac{1}{\sqrt{10}}\begin{pmatrix}3\\1\end{pmatrix}.$$

La notation de Dirac pour ce résultat est :

$$|\psi\rangle = \frac{3}{\sqrt{10}}|0\rangle + \frac{1}{\sqrt{10}}|1\rangle.$$

On voit clairement que l'amplitude de l'état $|0\rangle$ a été augmentée tandis que l'amplitude de l'état $|1\rangle$ est diminuée.

2.2.6.3 L'enchevêtrement

Ce phénomène n'a pas d'analogie classique, c'est la corrélation entre deux qubits : par exemple l'état : $|\psi\rangle = 1/\sqrt{2}\left(|00\rangle + |11\rangle\right)$ est en enchevêtrement, si le premier qubit est mesuré en 1(0) alors le deuxième qubit est sans aucun doute vaut 1(0) respectivement. Mathématiquement, l'état global d'un système quantique est dit enchevêtré s'il est impossible de l'écrire sous forme d'un produit tensoriel de deux de ses sous systèmes :

$$|\phi_{AB}\rangle \neq |\phi_A\rangle \otimes |\phi_B\rangle \qquad (2.5)$$

Ce principe permet de réaliser des calculs irréalisables sur des ordinateurs classiques. L'algorithme quantique de Shor [Shor, 1994] pour la factorisation des grands nombres est un bon exemple. Grâce à ce principe, Shor a donné un algorithme pour la factorisation des grands nombres dont la complexité est polynomiale.

2.2.6.4 Le non déterminisme

Contrairement à la physique classique, la mécanique quantique est indéterministe c'est-à-dire que les mêmes causes ne donnent pas forcement les mêmes résultats. La valeur d'un qubit en superposition n'est pas connue avant l'opération de mesure mais on peut augmenter la probabilité d'avoir un état en augmentant son amplitude.

2.2.6.5 Le non clonage

Le clonage d'un état indéterminé est impossible en mécanique quantique. Reprenons l'exemple du photon ; on ne réussi jamais à dupliquer un photon dont on ne sait rien car il est difficile de connaitre complètement son état quantique. Si on essaie dès qu'on commence à mesurer certaines quantités observables par exemple la polarisation selon une direction, on détruit l'état quantique du photon et on ne peut pas mesurer d'autres observables. Ceci est la conséquence directe de la linéarité des transformations unitaires. Supposons que U est une transformation unitaire qui fait le clonage. Ayant deux états quantiques orthogonaux |a> et |b>. L'application de U donne :

U(|a0>)=|aa> et U(|b0>)=|bb>.

Considérons l'état $|c\rangle = 1/\sqrt{2}\,(|a\rangle + |b\rangle)$, L'application de U sur c donne deux valeurs différentes :

- Par linéarité : $U(|c0>) = (U(|a0>) + U(|b0>)) = (|aa> + |bb>)$.
- Par clonage : $U(|c0>) = |cc> = 1/2(|aa> + |ab> + |ba> + |bb>)$

Ce qui est contradictoire.

2.2.7. La mesure quantique

La mesure quantique est une opération très importante en informatique quantique. C'est la possibilité de mesurer les états classiques de la mémoire. Avant l'opération de mesure, un bit quantique n'a pas une valeur fixe contrairement au bit classique qui a toujours une valeur définie. Un bit classique peut prendre une valeur de 0 ou 1, cependant un qubit peut être dans une superposition des deux au même temps. Théoriquement, La mesure quantique est une forme spécifique d'interaction entre l'ordinateur quantique et un dispositif physique externe de mesure. D'une manière simple, la mesure est la projection de l'état d'un qubit sur l'une des bases de l'espace de Hilbert. Par exemple l'état quantique $|\psi\rangle = a|\uparrow\rangle + b|\rightarrow\rangle$ est mesuré comme $|\uparrow\rangle$ avec la probabilité $|a|^2$ et comme $|\rightarrow\rangle$ avec la probabilité $|b|^2$ (figure 2.7). La mesure d'un qubit en superposition donne une seule valeur au qubit et elle dépend des amplitudes de ce qubit. Comme la mécanique quantique est indéterministe le résultat de la mesure est probabiliste. On ne peut pas avoir forcement les mêmes valeurs après deux mesures consécutives [Rieffel et al., 2000].

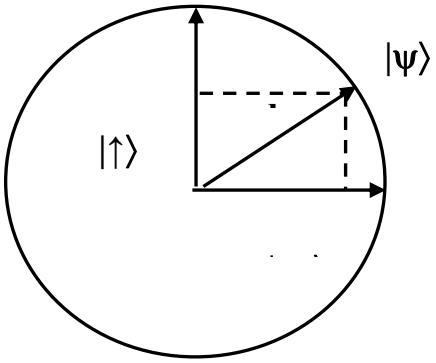


Figure 2.7 – Mesure quantique.

2.2.8. Opérations logiques quantiques

Le quatrième postulat de la mécanique quantique dit : *L'évolution au cours temps d'un système quantique isolé, est décrite par une transformation unitaire*. Donc, une opération dans le paradigme quantique doit satisfaire les conditions suivantes :

- Unicité : $f'*f = I$ (où *f'* est l'inverse de *f*, *I* : matrice d'identité)
- Linéarité : Si $|\psi>=\sum_{i=1}^{n} Ci|bi\rangle$ alors $f(|\psi i>)=\sum_{i=1}^{n} Ci|f(bi)\rangle$.

La dernière caractéristique est l'un des avantages de l'informatique quantique sur l'informatique classique. Grâce à cette caractéristique, une opération est appliquée sur tous les états en superposition parallèlement [Simon, 1994].

2.2.9. Circuits quantiques simples

Afin de pouvoir réaliser un ordinateur quantique, l'informatique quantique doit offrir les circuits de base pour la construction d'une telle machine. Mais, plusieurs portes classiques comme la porte AND sont des opérations irréversibles or les opérations dans le paradigme quantique doivent être réversibles. Heureusement, Deutsch [Deutsch, 1985] a démontré la possibilité de construire des portes quantiques réversibles pour n'importe quelle fonction classique. En fait, il est possible de concevoir une machine de Turing quantique universelle. Plusieurs portes ont été créées qui effectuent des opérations classiques telles que le OR, AND, NOT, ou des opérations propres au paradigme quantique telle que la superposition. Parmi les circuits les plus importants pour la construction des ordinateurs quantiques on distingue les suivants [Rieffel et al., 2000], [Blais et al., 2000]:

- *Circuits de Pauli* : Les circuits de Pauli constituent un ensemble de portes élémentaires permettant de construire toutes les portes logiques. N'importe quelle autre opération M peut être écrite sous la forme d'une combinaison linéaire de ces trois matrices plus la matrice d'identité :

$M = \lambda jI + \sum_{i}^{3} \lambda ipi$. I : matrice d"identité, Pi matrice de Pauli (X, Y, Z).

I : $|0\rangle \rightarrow |0\rangle$ $\quad \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$

$|1\rangle \rightarrow |1\rangle$

X : $|0\rangle \rightarrow |1\rangle$

$|1\rangle \rightarrow |0\rangle$ $\quad \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$

Y : $|0\rangle \rightarrow -|1\rangle$

$|1\rangle \rightarrow |0\rangle$ $\quad \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix}$

Z : $|0\rangle \rightarrow |0\rangle$

$|1\rangle \rightarrow -|1\rangle$ $\quad \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$

Les noms de ces transformations sont conventionnels : **I** : est la transformation identité. **X :** est la négation. **Z**: est la transformation de décalage de phase. **Y = Z* X** est une combinaison des deux.

➢ *La transformation de Hadamard :* C"est une operation tres imporatnte dans les algorithmes quantiques :

$$U_{WH}|0\rangle = \frac{1}{\sqrt{2}}\left(|0\rangle + |1\rangle\right) \qquad U_{WH}|1\rangle = \frac{1}{\sqrt{2}}\left(|0\rangle - |1\rangle\right)$$

Cette porte permet de créer des superpositions cohérentes et de représenter donc 2^n valeurs dans un registre à n qubits comme suit :

$$\begin{aligned} U_{WH}|0\rangle \otimes U_{WH}|0\rangle \otimes \ldots \otimes U_{WH}|0\rangle &= \frac{1}{\sqrt{2}}\left(|0\rangle + |1\rangle\right) \otimes \frac{1}{\sqrt{2}}\left(|0\rangle + |1\rangle\right) \otimes \ldots \otimes \frac{1}{\sqrt{2}}\left(|0\rangle + |1\rangle\right) \\ &= \frac{1}{\sqrt{2^n}}\left(|00\ldots0\rangle + |00\ldots1\rangle + \ldots + |11\ldots1\rangle\right) \\ &= \frac{1}{\sqrt{2^n}}\left(|0\rangle + |1\rangle + |2\rangle + \ldots + |2^n - 1\rangle\right) \end{aligned}$$

➢ *Controlled-Not* : c'est une opération binaire. Elle est très utile pour effectuer des opérations contrôlées. Elle change le deuxième bit si le premier est à 1, et laisse le bit inchangé sinon (figure 2.8).

$$C_{Not}: \quad |00\rangle \rightarrow |0\,0\rangle, \quad |01\rangle \rightarrow |0\,1\rangle, \quad |10\rangle \rightarrow |1\,1\rangle, \quad |11\rangle \rightarrow |1\,0\rangle \qquad \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

Figure 2.8 – Représentation matricielle du C_{Not}.

➢ *La porte de Toffoli :* L"opération AND est une opération non réversible. Par ailleurs, cette porte est très importante pour effectuer des calculs. Heureusement, ce problème a été résolu par la création de la porte Toffoli. Cette porte binaire permet la construction de la porte AND. La porte de Toffoli constitue un élément indispensable pour la construction d"un futur ordinateur quantique. La porte de Toffoli*:*

$$T(|x, y, 0\rangle) = |x, y, x \text{ AND } y\rangle.$$

La figure suivante montre un circuit quantique complexe qui effectue une tâche bien précise comme l"adition est composé d"un ensemble de circuits simples.

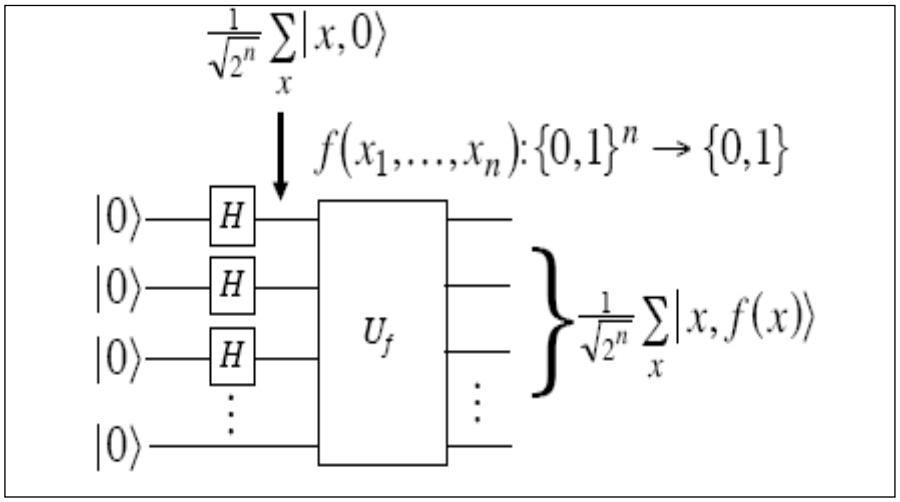


Figure.2.9 – Un circuit quantique complexe.

2.2.10. Le paradigme standard

Le paradigme quantique a été énoncé par D. Divencenzo. Il consiste des cinq critères de base qu'un ordinateur quantique doit satisfaire [Blais, 2002]:

1) ***Qubits :*** Il doit y avoir des qubits.

2) ***Initialisation :*** Il doit être possible d'initialiser le registre quantique dans un état connu, souvent choisi comme : $|0\rangle = |000...00\rangle$.

3) ***Calcul :*** un registre quantique est manipulable par un ensemble universel de portes logiques.

4) ***Cohérence :*** il faut minimiser ou éliminer toute interaction du système quantique avec l'environnement externe pendant les calculs.

5) ***Mesure :*** Chaque qubit doit pouvoir subir une mesure projective et la valeur du bit classique qui en résulte doit être obtenue avec une grande efficacité.

2.2.11. Les algorithmes quantiques

Afin d'exploiter efficacement la puissance de l'informatique quantique, il faut créer des algorithmes quantiques efficaces. Un algorithme quantique est constitué d'une suite d'opérations quantiques sur des systèmes quantiques. Ces opérations quantiques peuvent être des portes logiques quantiques simples ou bien des circuits quantiques. En informatique quantique, on a la possibilité de représenter 2^n valeurs possibles dans un seul registre quantique de taille n et de les traiter parallèlement et cela grâce aux concepts de la superposition d'états, la linéarité des opérations et l'enchevêtrement. Le principe de la superposition permet un stockage exponentiel. Par ailleurs, la linéarité et l'enchevêtrement permettent un traitement massivement parallèle. Les algorithmes quantiques sont donc plus efficaces que les algorithmes classiques. Profitant de ces capacités, plusieurs algorithmes ont vu le jour qui essayent de démontrer l'efficacité de l'informatique quantique pour la résolution des problèmes dont la solution classique est coûteuse en temps et en espace. Cependant, l'écriture d'un algorithme quantique est encore une tâche dure. En effet, Les gens ne sont pas encore familiarisés avec des concepts étranges tels que la superposition d'états et l'enchevêtrement. En 1994 Shor [Shor, 1994] a surpris le monde en donnant un algorithme polynomial pour la factorisation de grands nombres, un autre algorithme qui a rendu l'informatique quantique plus intéressante est celui de Grover [Grover, 1996] pour la recherche dans une base de données non triée dont la complexité est quadratique. Nous présentons dans ce qui suit les grandes lignes de l'algorithme de Grover

2.2.11.1 Algorithme de recherche de Grover

Cet algorithme démontre bien que l'informatique quantique peut être plus puissante que les ordinateurs classiques pour la résolution de certains problèmes NP-difficiles. Grover a démontré que son algorithme permet de trouver un élément satisfaisant une condition donnée dans une base non triée dans un temps quadratique. L'algorithme est constitué des étapes décrites dans l'algorithme 2.1

Dans la première étape on utilise l'opération de Hadamard pour créer une superposition cohérente d'états (figure 2.10). Cette dernière contient donc une superposition de toutes les

solutions possibles. La deuxième étape de l'algorithme de Grover consiste à inverser l'amplitude des bonnes solutions (figure 2.11).

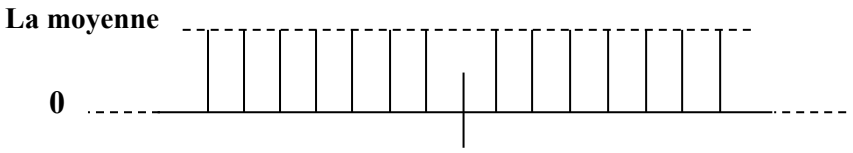


Figure 2.10 – Création d'une superposition d'états.

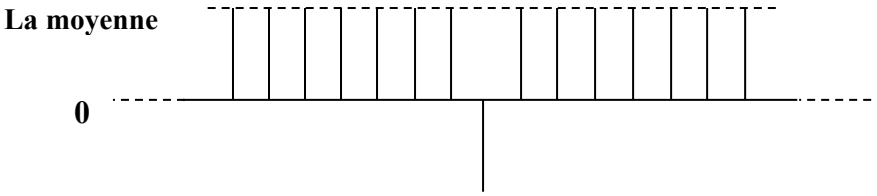


Figure 2.11 – Inversement de l'amplitude de la bonne solution.

La troisième opération consiste à augmenter l'amplitude de la bonne solution en utilisant une opération d'inversement par rapport au moyen (figure 2.12).

$$D = \begin{pmatrix} \frac{2}{N} & \frac{2}{N} & \cdots & \frac{2}{N} \\ \frac{2}{N} & \frac{2}{N} & \frac{2}{N} & \frac{2}{N} \\ \cdots & \cdots & \cdots & \cdots \\ \frac{2}{N} & \frac{2}{N} & \frac{2}{N} & \frac{2}{N} \end{pmatrix}.$$

Figure 2.12 – Une opération d'inversement par rapport au moyen**.**

Apres un certain nombre d'itérations, l'amplitude de la bonne solution sera grande et les amplitudes des autres solutions seront diminuées (figure 2.13). Ce qui augmente la probabilité de mesurer la bonne solution. Finalement on applique une opération de mesure pour extraire la solution.

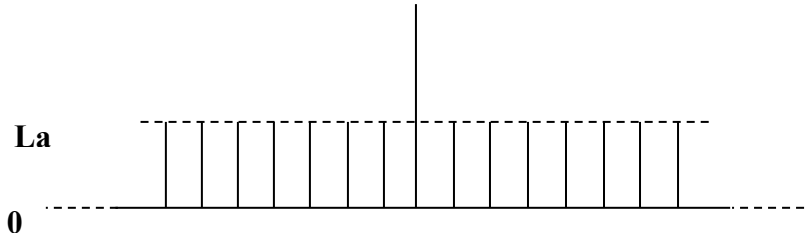


Figure 2.13 – L'augmentation de l'amplitude de la bonne solution.

Grover a démonté qu'après avoir faire $\frac{\pi}{4}\sqrt{2^n}$ d'itérations, le taux d'échec de mesurer la bonne solution est de 2^{-n}. Par ailleurs, l'augmentation des itérations n'augmente pas la probabilité de la mauvaise solution. Dans le cas où il existe b états qui satisfont la condition, la complexité est de l'ordre O ($\sqrt{2^n / b}$).

Algorithme 2.1: Algorithme de recherche de Grover

début

1) Création d'une superposition en appliquant un Hadamard sur un registre de taille n initialisé à|0..0>.
2) répéter les étapes suivantes $\sqrt{2^n}$ fois :
3) pour tout état qui satisfait l'oracle appliquer une opération shift (inverser la phase)
4) appliquer sur le registre une opération de rotation par moyenne
5) mesurer le registre : on obtient l'état désiré avec une grande probabilité.

fin

2.2.11.2 Algorithme de Grover revisité

L'algorithme de Grover revisité est plus général que l'algorithme classique de Grover. Ce dernier s'applique seulement dans le cas d'une superposition d'amplitudes uniforme (le nombre d'états dans la superposition égale à 2^n (n le nombre de qubit). Si on l'applique sur une superposition d'amplitudes arbitraire on aura une mauvaise probabilité d'avoir la bonne solution. La raison est simple : après la première itération on aura deux types d'états indésirables : les états qui ne satisfont pas la condition de la recherche, et les états qui n'existent pas dans la superposition initiale et qui surgissent après l'application de l'opération d'inversement par la moyenne de Grover. Pour remédier à ce problème, Biron introduit une nouvelle rotation qui sera appliquée après la première itération de l'algorithme classique. Cette rotation inverse la phase de l'état désiré et de tous les états qui se trouvent dans la superposition initiale. Puis on applique l'algorithme ordinaire de Grover. Cet algorithme comprend les étapes décrites pas l'algorithme suivant [Biron et al, 1998]. L'algorithme de Grover est très utile dans la résolution de plusieurs problèmes NP-difficile comme le problème du voyageur du commerce.

Algorithme 2.2: Algorithme de recherche de Grover revisité

Début

1. Initialisation : constitution d'une superposition arbitraire de taille n.
2. Inverser la phase de l'état qui satisfait l'oracle.
3. Appliquer la matrice de rotation par la moyenne de Grover.
4. Appliquer la rotation de Biron sur tous les états initiaux de la superposition.
5. Appliquer la matrice de rotation par la moyenne de Grover.
6. **Si** le nombre d'itération est inférieur à ($\sqrt{n}$ -1) alors exécuter la boucle suivante, **sinon** aller à **fin**.
   - Inverser la phase l'état qui satisfait l'oracle.
   - Appliquer la matrice de rotation par la moyenne de Grover.

fin

2.2.12. Réalisation physique de l'ordinateur quantique

La construction à grande échelle des ordinateurs quantiques sera des systèmes quantiques complexes contenant plusieurs parties. La réalisation d"un tel système doit faire face à des difficultés expérimentales et théoriques énormes et non encore résolues. L"un des problèmes les plus difficiles dans le développement des systèmes informatiques quantiques est le maintien de la cohérence du système jusqu"à la fin des calculs. Cette perte de cohérence (décohérence) est due à l"interaction des qubits avec le monde extérieur. Plus le nombre de qubits est grand plus la probabilité qu"un qubit interagisse avec l"environnement. Une découverte essentielle pour réduire le taux d"erreur est l"existence des correcteurs d"erreurs. Le principe utilisé est de coder l"état d"un qubit de manière redondante sur une superposition judicieusement choisi de plusieurs qubits. L"altercation d"un qubit est donc détectée et corrigée sans extraire l"information sur l"état du qubit initial. Ce qui préserve la cohérence du système [Brassard, 1996]. Un ordinateur quantique peut être construit en utilisant n'importe quelle petite particule qui peut avoir deux états. Des ordinateurs quantiques peuvent être construits à partir d'atomes qui sont à la fois excités et non excités au même moment. Ils peuvent être construits à partir de photons de lumière qui sont à deux endroits en même moment. Ils peuvent être construits à partir de protons et de neutrons ayant un spin soit positif soit négatif ou les deux en même temps. Les différentes techniques pour la réalisation d'un ordinateur quantique peuvent être trouvées dans [Blais, 2002]. De nombreux projets sont en

cours à travers le monde pour construire concrètement des qubits viables et les réunir dans un circuit. Ces recherches mettent en œuvre de la physique théorique pointue. En 2009, des chercheurs de l'université de Yale ont créé le première micro processeur rudimentaire à base de deux qubits. La puce quantique a été en mesure d'exécuter des algorithmes élémentaires. Chacun des deux atomes artificiels (ou qubits) étaient constitués d'un milliard d'atomes d'aluminium, mais ils ont agi comme une seule et qui peuvent occuper deux états d'énergie différents. En parallèle la société américaine D-Wave se concentre sur le développement des processeurs basés sur la supraconductivité capables d'exécuter des algorithmes quantiques adiabatiques pour résoudre des problèmes optimisation quadratique sans contrainte binaire. En 2008, la même société a conçu une machine avec 28 qubits.

- *Remarque*

Les ordinateurs quantiques peuvent être plus rapides que les ordinateurs classiques (de von-Neumann), mais les ordinateurs quantiques ne peuvent pas résoudre tous les problèmes que les ordinateurs classiques ne peuvent pas résoudre. D"autre part, une machine de Turing peut simuler un ordinateur quantique, ainsi un ordinateur quantique ne pourrait pas résoudre un problème indécidable.

« L'existence des ordinateurs quantiques ne réfute pas la thèse Church-Turing »

2.3. Les algorithmes inspirés du quantique

Vu que les algorithmes quantiques purs sont difficiles à concevoir en raison de la non disponibilité actuellement des machines quantiques. Plusieurs chercheurs s"intéressent à la combinaison des algorithmes classiques aux principes de l"informatique quantique. L"avantage des algorithmes inspirés de la mécanique quantique est qu"ils ne nécessitent pas la présence d'ordinateurs quantiques [de Garis et al, 2003]. Le premier algorithme inspiré du quantique est celui de Moore [Moore, 1995] pour le tri de nombres. La complexité de son algorithme est $\sqrt{n}\left[n+2\left(\sqrt{n}-1\right)\right]$. Cependant, la complexité de l"algorithme classique pour le tri de Knuth est ½n(n-1) . En 2000, Han et Kim [Han et al, 2000] ont proposé le premier algorithme génétique inspiré du quantique. En 2001, ils proposent un algorithme quantique évolutionnaire [Han et al ,01]. Cet algorithme combine les idées de l'informatique quantique aux algorithmes évolutionnaires classiques. Les algorithmes inspirés de la mécanique quantique ont donné de bons résultats face à des problèmes d"optimisation combinatoire comme le problème du recalage d"image [Draa et al., 2004], [Talbi et al., 2006], le problème de sac à dos [Han et al. ,2001], le problème d'allocation de disque [Kim et al., 2003], etc.

2.3.1. La méthodologie des méthodes inspirées du quantique

Les méthodes inspirées du quantique doivent avoir les caractéristiques suivantes [Moore, 1995] :

1. Le problème doit être exprimé sous forme numérique.
2. La configuration initiale doit être déterministe.
3. Les critères d'arrêt doivent être définis avec concision.
4. Le problème peut être divisé en sous problème.
5. Le nombre de population doit être défini.
6. Assigner un sous problème pour chaque population.
7. Le calcul dans les différentes populations est effectué en parallèle.
8. Il doit y avoir quelque interaction inter populations. L'interférence doit améliorer la solution ou donner une information pour la population pour trouver la solution.

Les caractéristiques ci-dessus donnent une vue utile pour désigner les méthodes inspirées de la mécanique quantique, mais elles ne montrent pas comment déterminer la configuration initiale. Probablement qu'il existe une relation directe entre le type de problème et la configuration utilisée. La division de problème en sous problèmes est l'élément clé des méthodes inspirées du quantique.

2.3.2. Algorithmes quantiques évolutionnaires

Comme nous avons vu dans le chapitre précédent, les algorithmes génétiques sont des métaheuristiques très performantes pour donner des solutions approchées proches de l'optimum. Cependant, ils présentent quelques limitations comme la lenteur et la grande taille de la population utilisée pour trouver la meilleure solution. Récemment plusieurs chercheurs étudient l'effet d'ajouter d'autres opérations inspirées d'autres domaines comme l'informatique quantique dans les algorithmes évolutionnaires. Une nouvelle classe d'algorithmes est donc née : les Algorithmes Quantiques Evolutionnaires AQEs.

Définition 2.3 : (Un Algorithme Quantique Evolutionnaire) : Un Algorithme Quantique Evolutionnaire (AQE) est un Algorithme Evolutionnaire (AE) qui combine entre les algorithmes évolutionnaires classiques et les concepts et principes de l'informatique quantique, tel que le bit quantique, la superposition d'états, la mesure, etc. un AQE a les mêmes caractéristiques d'un AE classique tel que la représentation des individus, la fonction d'évaluation et la dynamique de la population. Néanmoins, l'AQE utilise une représentation quantique au lieu des représentations usuelles des AEs. En effet, un AQE utilise le bit quantique (qubit) comme une représentation probabiliste, défini comme la plus petite unité d'information [Han et al, 2001].

2.3.3. Une représentation quantique des chromosomes

Une nouvelle représentation est adoptée pour les algorithmes quantiques évolutionnaires. Chaque population est constituée d'un ensemble d'individus quantiques. Les individus en AQEs sont représentés par des registres de qubits. Un qubit est la plus petite unité d'information dans un AQE, défini par une paire de nombres (α, β) comme suit :

$$|\psi\rangle = \begin{pmatrix} \alpha \\ \beta \end{pmatrix} \text{ Où } \ |\alpha|^2 + |\beta|^2 = 1$$

$|\alpha|^2$, $|\beta|^2$ est la probabilité d'avoir les états „0", „1" respectivement. Selon les principes de l'informatique quantique, un qubit peut être à l'état "1", à l'état „0" ou dans une superposition des deux. Un chromosome quantique est une chaîne de qubits défini par :

$$\begin{pmatrix} \alpha_1 \, \alpha_2 \ldots \alpha_m \\ \beta_1 \, \beta_2 \ldots \beta_m \end{pmatrix} \text{ Où : } |\alpha_i|^2 + |\beta_i|^2 = 1 \text{ , } i=1,2,\ldots, m.$$

En comparant avec les représentations classiques des chromosomes, Cette représentation quantique a l'avantage de représenter une superposition linéaire des états. D'une autre façon, elle encode une superposition d'un ensemble de solutions en utilisant un seul registre de qubits. Cependant, la représentation classique ne permet qu'une seule représentation d'une solution. Par exemple : imaginons un problème dont la solution est représentée sur un nombre binaire de trois bits. Soit le système à trois qubits avec trois paires d'amplitudes suivant :

$$\begin{pmatrix} 1/\sqrt{2} & 1/\sqrt{2} & 1/\sqrt{2} \\ 1/\sqrt{2} & -1/\sqrt{2} & \sqrt{3}/\sqrt{2} \end{pmatrix}$$

Ce système représente les états $|000\rangle, |001\rangle, |010\rangle, |011\rangle, |100\rangle, |101\rangle, |110\rangle$ et $|111\rangle$.

$$1/4|000\rangle + \sqrt{3}/4\,|001\rangle - 1/4|010\rangle - \sqrt{3}/4\,|011\rangle + 1/4|100\rangle + \sqrt{3}/4\,|101\rangle - 1/4|110\rangle - \sqrt{3}/4\,|111\rangle$$

La représentation quantique en ket indique que les probabilités de représenter les états précédents sont respectivement : $1/16$, $3/16$, $1/16$, $3/16$, $1/16$, $3/16$, $1/16$ et $3/16$.

On voit clairement que la représentation quantique a une meilleure diversité que les autres représentations classiques. En effet, On a seulement besoin de trois qubits pour représenter les huit états possibles. Contrairement à cette représentation, il faut avoir huit registres classiques de trois bits pour encoder tous les états possibles.

2.3.4. Les opérations quantiques

Comme les algorithmes évolutionnaires classiques, les AQEs utilisent des opérations quantiques afin de diversifier la recherche, et de mener les individus vers les meilleures solutions. Ces opérations doivent satisfaire les conditions de la linéarité et de la réversibilité de l'informatique quantique. Dans [Han et al., 2001], une porte quantique est définie comme un opérateur de variation de l'AQE par lequel le qubit mis à jour doit satisfaire la condition de normalisation $|\alpha'|^2+|\beta'|^2=1$ où α' et β' sont les nouvelles valeurs du qubit mis à jours. Généralement, trois portes sont massivement utilisées dans un AQE [Han et al, 2001] :

- *La porte de rotation* : Comme dans l'algorithme de Grover, la porte de rotation est utilisée comme une porte quantique dans l'AQE afin d'augmenter la probabilité de la bonne solution :

$$U(\Delta\theta_i)=\begin{bmatrix}\cos(\Delta\theta_i) & -\sin(\Delta\theta_i)\\ \sin(\Delta\theta_i) & \cos(\Delta\theta_i)\end{bmatrix}$$

Où : $\Delta\theta_i, i=1,2,...,m$ est un angle de rotation de chaque qubit vers l'état 0 ou 1 dépendant de son signe. $\Delta\theta_i$ est un paramètre empirique dépendant du problème d'application. La figure 2.14 montre la courbe polaire de la porte de rotation. La table 2.3 peut être utilisée comme des paramètres d'angle pour la porte de rotation. Le pseudo code détaillé de la porte de rotation est décrit par l'algorithme 2.3 [Han et al., 2001]:

Algorithme 2.3 : Procédure Update (q)

```
begin
    i ← 0
    while (i < m) do
    begin
    i ← i + 1
    determine Δθ_i with the lookup table
    obtain (α_i', β_i') from the following :
     if (q is located in the first/third quadrant)
     then [α_i', β_i']^T = U(Δθ_i)[α_i, β_i]^T
     else [α_i', β_i']^T = U(−Δθ_i)[α_i, β_i]^T
    end
    q ← q′
end
```

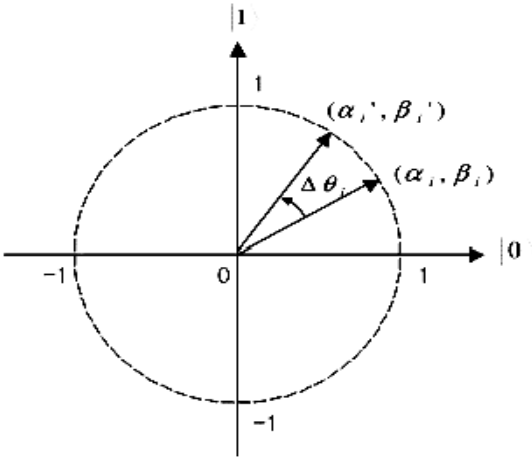


Figure 2.14 – Rotation d'un qubit

Table 2.3. Lookup table.

Angle	Valeur du bit de référence	β	α
$+\delta\theta$	**1**	**> 0**	**> 0**
$-\delta\theta$	**0**	**> 0**	**> 0**
$-\delta\theta$	**1**	**< 0**	**> 0**
$+\delta\theta$	**0**	**< 0**	**> 0**
$-\delta\theta$	**1**	**> 0**	**< 0**
$+\delta\theta$	**0**	**> 0**	**< 0**
$+\delta\theta$	**1**	**< 0**	**< 0**
$-\delta\theta$	**0**	**< 0**	**< 0**

- *La porte NOT* : Cette porte opère de la même façon qu"une mutation naturelle. Elle opère sur un qubit en permutant les deux amplitudes de l'état 1(0) pour avoir l'état 0(1) respectivement (figure 2.15). Cette porte est très utile pour s"échapper du problème de l"optimum local.

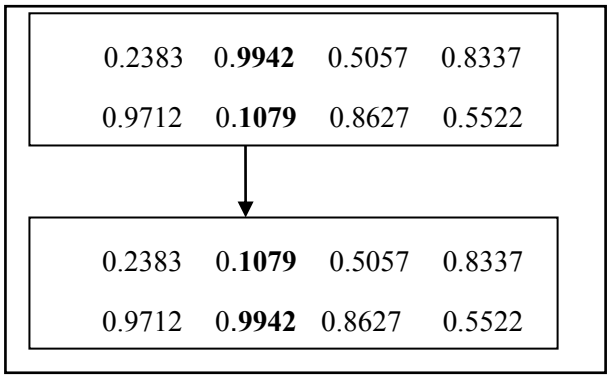


Figure 2.15 – La porte NOT.

- *La porte Controlled-NOT* : Cette porte est utilisée pour effectuer des instructions contrôlées. Dans ce cas l'un des deux qubits doit être un bit de contrôle. Si le bit de contrôle est à 1, alors l'opérateur NOT est appliqué à l'autre bit (figure 2.16). cette porte est recommandée dans les problèmes contenant de grandes dépendances entre deux qubits.

$\mathbf{C_{Not}}$:

$$|00\rangle \rightarrow |00\rangle \qquad \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

$$|01\rangle \rightarrow |01\rangle$$

$$|10\rangle \rightarrow |11\rangle$$

$$|11\rangle \rightarrow |10\rangle$$

Figure 2.16 – La porte Controlled-NOT.

- *La porte de Hadamard* : Cette porte est appropriée pour les algorithmes qui utilisent l'information de phase des qubits aussi bien que l'information amplitude (figure 2.17).

$$U_{WH}|0\rangle = \frac{1}{\sqrt{2}}\left(|0\rangle + |1\rangle\right) \qquad U_{WH}|1\rangle = \frac{1}{\sqrt{2}}\left(|0\rangle - |1\rangle\right)$$

Figure 2.17 – La porte de Hadamard.

2.3.5. Algorithmes quantiques génétiques

Afin d'augmenter la vitesse de convergence des AQEs plusieurs portes quantiques inspirées des algorithmes génétiques ont été introduites. Un nouveau type des AQEs est donc apparu, il s'agit des Algorithmes Quantiques Génétiques (AQG) [Han et al., 2001]. Ces algorithmes se distinguent sur les AQEs par l'introduction des opérateurs génétiques tels que la mutation et le croisement quantique.

Définition 2.4 : (Un algorithme quantique génétique) Un algorithme quantique génétique est similaire à un algorithme génétique classique. Cependant, il est enrichi par des opérations quantiques tels que l'enchevêtrement, la mesure, l'interférence, etc [Han et al., 2000]. Sa structure ressemble aux algorithmes quantiques évolutionnaires.

2.3.6. Opérateurs génétiques quantiques

Dans l'expérience de knapsack, Han et Kim n'ont pas utilisé la mutation et le croisement dans leur AQG. Dans ce problème la mesure et l'interférence sont largement suffisantes pour avoir de bonnes solutions [Han et al., 2000]. Cependant, les travaux de [Talbi et al., 2004] et [Draa et al., 2004] ont montré que les opérations de mutation et de croisement peuvent être utiles dans quelques problèmes d'optimisation tel que le recalage d'image.

- *Croisement quantique:* Le croisement quantique a le même principe que le croisement dans les algorithmes génétiques classiques. Cependant, il opère sur des chromosomes quantiques. Les chromosomes sont choisis d'une manière aléatoire. Ensuite, les deux chromosomes quantiques choisis échangent quelque qubits entre eux. L'exemple suivant explique le fonctionnement d'un croisement quantique. Le croisement combine entre les chromosomes quantiques *a* et *b* et produit deux chromosomes quantiques fils *c* et *d* en sortie. L'avantage de croisement quantique est l'exploration de nouvelles solutions [Draa et al., 2004]. L'exemple suivant montre un croisement quantique au point 2.

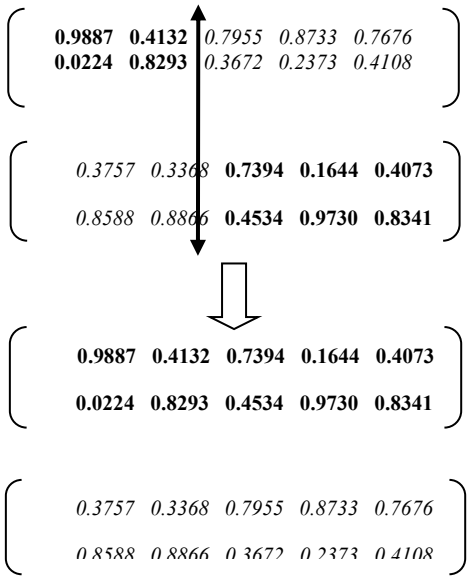


Figure 3.18 – Croisement quantique.

- *Mutation quantique:* Cette mutation est basée sur la porte NOT (figure 2.15). Mais on peut adapter plusieurs types de mutation selon la nature de problème [Layeb et al., 2007 et 2006].

2.3.7. Algorithme de Recherche Dispersée Quantique

La Recherche Dispersée (RD) [Laguna, 2002] [Glover, 1998] est un nouvel algorithme d'optimisation qui combine entre les principes des algorithmes génétiques et les techniques de la recherche Tabou. Ses bases sont dérivées des stratégies proposées pour combiner les règles de décision et des contraintes dans le cadre de la programmation en nombres entiers. Dans le

processus de d"optimisation par la recherche dispersée, chaque solution est codée sur un vecteur n-dimensionnel. La méthode de recherche dispersée emploie des stratégies de combinaison entre les vecteurs de solution qui ont été montrées efficaces dans plusieurs problèmes d'optimisation. Pour éviter une convergence prématurée vers un optimum local, la RD gère différemment sa population, en y incluant: une sous population élite comprenant tous les meilleurs éléments trouvés et une sous-population diversifiée, comprenant des solutions "différentes" de celles de la première sous-population. Comparé aux algorithmes génétiques, la taille de la population dans la recherche dispersée est petite. La recherche dispersée est composée des cinq méthodes suivantes: la méthode de génération de diversification, la méthode d"amélioration, la méthode de mise à jour d'ensemble de référence, la méthode de génération de sous-ensemble et la méthode de combinaison [Glover, 1998].

L"algorithme de *recherche dispersée quantique* est un nouvel algorithme qu"on a proposé dans [Layeb et al, 2008f] qui intègre les principes de l"informatique quantique dans un algorithme de recherche dispersée. Cet algorithme est constitué des cinq méthodes suivantes :

- *La méthode de génération de diversification :* L'idée derrière la méthode de génération de diversification est de créer un ensemble de solutions diverses. La nature probabiliste de l'algorithme quantique offre une bonne manière de produire une nouvelle solution. En conséquence, nous appliquons l'opération de mesure plusieurs fois afin de produire de l'ensemble initial de solutions d'essai.
- *La méthode d'amélioration:* La méthode d'amélioration est employée pour améliorer une solution par l'intermédiaire d'un procédé de transformation. Dans notre algorithme, nous avons employé un opérateur simple d'échange. Cet opérateur exécute la permutation entre deux qubits. Elle laisse se déplacer de la solution courante à une de ses voisines. Cet opérateur aide à explorer de nouvelles solutions et à renforcer ainsi le potentiel de diversification du processus de recherche. En outre, nous appliquons un autre opérateur inspiré du quantique : Interférence Quantique
- *La méthode de mise à jour d'ensemble de référence* : Le but est de produire un ensemble de solutions de qualité et de solutions diverses. La méthode de mise à jour d'ensemble de référence suit chaque application de la méthode d'amélioration. Généralement, elle s'est exécutée bien après la méthode d'amélioration en raison de son rôle d"enchaînement avec la méthode de génération de sous-ensemble. La méthode de mise à jour d'ensemble de référence est basée sur l"évaluation de la qualité d'une solution de candidat. Afin d'éviter une convergence prématurée de notre algorithme, l'ensemble de référence doit préserver la qualité et la variété. Par conséquent, l'ensemble de référence RefSet est divisé en deux groupes : RefSet1 qui contient l'ensemble des meilleures solutions et du RefSet2 qui contient l'ensemble de mauvaises solutions a comparé à ceux du RefSet1. Cette étape remplace la plus mauvaise solution par la nouvelle meilleure solution trouvée. En

conséquence, des solutions sont incluses dans l'ensemble de référence par qualité ou diversité.

- *La méthode de génération de sous-ensemble* : Cette méthode produit des sous-ensembles de solution qui sont employés dans la méthode de combinaison. Chaque sous-ensemble produit contient deux, trois éléments ou plus. Ainsi, nous prenons aléatoirement des solutions des deux groupes de ReftSet1 et ReftSet2 afin de produire de nouveaux sous-ensembles divers. Par conséquent, le sous-ensemble contient les bonnes solutions et les mauvaises solutions.
- *La méthode de combinaison* : Cet opérateur effectue la combinaison des solutions. Elle consiste en la création de nouvelles solutions en employant une combinaison linéaire des matrices de quantum (figure.2.19). Dans notre algorithme, nous avons employé l'opération linéaire simple de combinaison. Nous choisissons aléatoirement trois matrices de quantum. Puis, nous produisons un nouveau vecteur de quantum des trois vecteurs. Afin de maintenir l'unité caractéristique du qubit, nous employons les angles représentés par chaque qubit. En fait, nous pouvons donner à chaque qubit *Q* (a_i, b_i) *une* représentation géométrique : *Q1 (cosinus (Φ1), sinus (Φ1)).*

0.5644	0.9219	1.0000	0.2442	0.6121
0.8255	0.3875	0	0.9697	0.7908
		−		
0.7216	0.8913	0.8343	0.2574	0.6174
0.6923	0.4534	0.5514	0.9663	0.7866
		+		
0.8394	0.3333	0.4703	0.2028	0.1987
0.5435	0.9428	0.8825	0.9792	0.9801

Figure 2.19 – Combinaison linéaire de trois vecteurs quantiques

2.3.8. Les avantages des algorithmes quantiques évolutionnaires

Les algorithmes inspirés du quantique présentent plusieurs avantages. ils permettent un traitement parallèle de plusieurs données. Contrairement aux AG et AE classiques, le principe de la superposition permet de réduire considérablement la taille de la population dans les AQEs sans perdre d'informations. Les AQEs permettent une balance automatique entre la recherche locale et la recherche globale. Une grande caractéristique des AQEs est que l'historique des individus n'est pas perdu dans les itérations suivantes dû à la nature probabiliste. Finalement, L'opération d'interférence permet l'accélération du processus de convergence vers la meilleure solution.

2.4. Conclusion

Il est tout à fait clair que le calcul quantique offre un gain considérable en temps et en espace. L'informatique quantique est caractérisée par des capacités de traitement et de stockage énormes. Cependant, les idées de l'informatique quantique ne sont pas encore exploitables vu le manque de moyens théoriques est surtout pratiques pour la conception d'un tel système. En l'attente de la construction des machines quantiques puissantes, la combinaison entre les approches classiques et l'informatique quantique constitue un domaine prometteur. En effet, les algorithmes quantiques évolutionnaires ont démontré leur efficacité dans plusieurs problèmes d'optimisation combinatoire. Pour cela nous avons utilisé ce paradigme comme un cadre de résolution pour les problèmes traités dans cette thèse.

CHAPITRE 3

Diagrammes de décision binaire

« I seem to have been only like a boy playing on the seashore, and diverting myself in now and then finding a smoother pebble or a prettier shell than ordinary, whilst the great ocean of truth lay all undiscovered before me. »

Isaac Newton

Le but de la vérification est de s'assurer qu'un programme ou un matériel informatique fonctionncorrectement et répond aux exigences des clients. Certaines techniques de vérification permettent seulement de détecter des erreurs, et d'autres comme la vérification par évaluation de modèle permettent de déterminer avec certitude le bon ou le mauvais fonctionnement d'un système. Un des principaux problèmes en vérification est le problème de l'explosion du nombre d'états dans les modèles à vérifier. Une des solutions proposées pour ce problème est l'utilisation des diagrammes de décision binaire pour représenter l'espace d'états du modèle ainsi que ses transitions.

Par conséquent, ce chapitre permet d'explorer le domaine des diagrammes de décision binaire (BDDs) ainsi que le problème d'ordre des variables d'un BDD.

Sommaire

3.1. Introduction

L'objectif de la vérification des applications et des circuits électriques est de déceler les erreurs qu'ils contiennent ou de démontrer qu'ils fonctionnent correctement. La vérification se fait soit directement sur le système soit sur un modèle représentatif de celui-ci. Il existe plusieurs approches de vérification qu'on peut classer en deux catégories: les approches informelles comme la simulation et les tests, et les approches formelles telles la démonstration de théorème et la vérification par évaluation de modèle (figure 3.1).

La vérification par évaluation de modèle (model-checking en anglais) [Malik et al., 1988] est une technique entièrement automatique basée sur les modèles permettant de décider si un matériel informatique ou un programme, exprimé comme un système de transition concurrent, satisfait un ensemble de propriétés exprimées dans une logique temporelle telle que CTL. La vérification d'un système est effectuée en parcourant toutes les exécutions possibles de celui-ci, qu'il s'agit d'un logiciel ou d'un matériel informatique. Pour ce faire, l'outil de vérification doit analyser tous les états et les chemins dans l'espace d'états. D'une manière ou d'une autre, il faut pouvoir réserver un espace en mémoire pour stocker tous les états et les transitions qui existent entre ceux-ci. Par conséquent, l'un des problèmes rencontré dans le domaine de la vérification formelle est l'explosion combinatoire. Par exemple dans le model-checking, le nombre d'états dans les graphes des transitions peut atteindre des niveaux prohibitifs ce qui rend leur manipulation difficile, voire impossible [McMillan, 1992]. Pour cela, on utilise des méthodes de compression afin de réduire la taille des graphes d'états. La compression se fait en utilisant des structures de données afin de représenter de façon concise des ensembles d'états. Dans ce cas, les opérations se font alors sur des ensembles d'états plutôt que sur des états explicites. Cette technique est souvent qualifiée de symbolique.

Les fonctions booléennes sont devenues de plus en plus grandes et complexes. Pour surmonter ce problème, de nouvelles techniques de représentation des fonctions booléennes ont été développées. Elles sont généralement basées sur la récursivité du théorème de Shannon [Shannon, 1940] et non plus sur les théorèmes habituels du calcul. Parmi ces nouvelles techniques, on distingue la représentation par diagrammes de décision binaire (BDD, de Binary Decision Diagrams) [Akers, 1978]. Un BDD est une structure de données particulièrement efficace pour représenter et manipuler les fonctions booléennes. Cette technique a d'abord été appliquée avec succès à la vérification formelle de circuits combinatoires et séquentiels, puis à d'autres domaines comme la synthèse de circuits, la démonstration automatique et l'analyse de fiabilité de systèmes complexes. Cependant, la taille des BDDs dépend étroitement de l'ordre de variables choisi lors de la construction du BDD représentant une fonction [Bollig et al., 1996]. De ce fait, il est important de trouver un bon ordre de variables qui minimise le nombre de nœuds dans un BDD. Malheureusement, cette tâche n'est pas facile vu qu'on a un nombre exponentiel d'ordres possibles. En conséquence, le problème de l'ordre des variables d'un BDD a été démontré NP-difficile

[Bollig et al., 1996]. Pour cela, plusieurs méthodes ont été proposées pour trouver le bon ordre de variables d‟un BDD et qu‟on peut classer comme suit. La première classe tente d‟extraire le bon ordre en inspectant les circuits logiques [Fujii et al., 1993] et [Fujita et al., 1988]. Par ailleurs, la deuxième classe est basée sur l‟optimisation dynamique d‟un ordre donné [Costa et al., 2000] et [Ishiura et al., 1991].

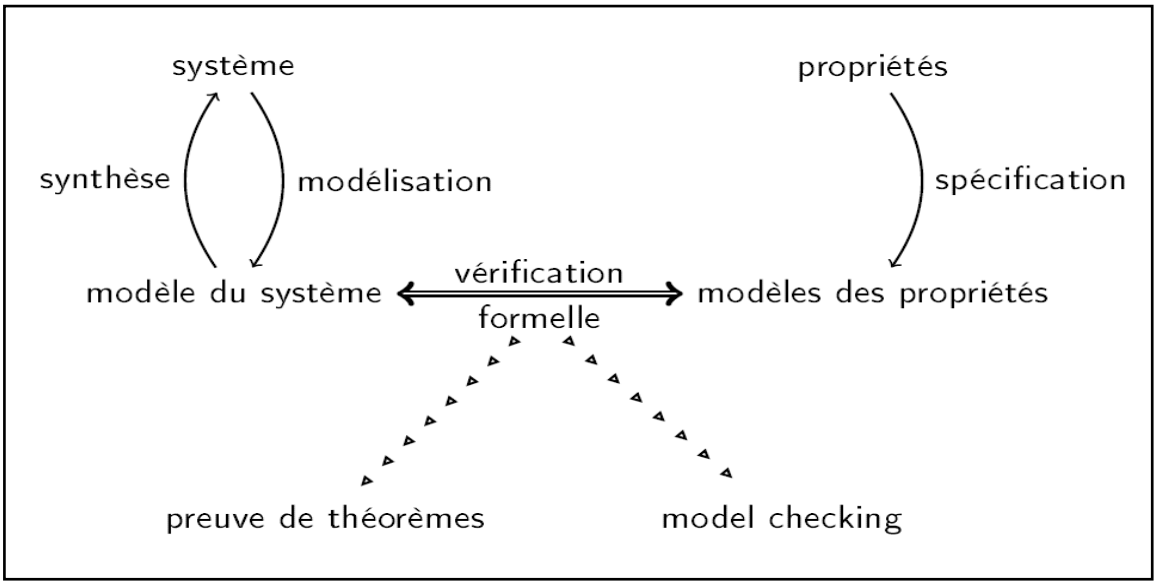


Figure 3.1 – La vérification Formelle.

3.2. Arbres de décision binaire

Les arbres de décision binaires constituent un modèle de représentation des fonctions booléennes. Un arbre de décision binaire est un arbre orienté, composé d‟une racine, de sommets intermédiaires et de sommets terminaux valant 0 ou 1 (figure 3.2). La racine et les sommets intermédiaires sont indexés et possèdent deux sommets « fils », un fils gauche et un fils droit. Le fils gauche est atteint en empruntant la branche 0, le fils droit en empruntant la branche 1. Un arbre de décision binaire est obtenu en appliquant récursivement la première forme du théorème de Shannon sur l‟ensemble des variables de la fonction [Doyle et al., 1995]. Le résultat fondamental sur lequel sont fondés les arbres de décision binaires est le *théorème d'expansion* de Shannon [Shannon, 1940].

Soit la formule booléenne suivante :

$F(x1, x2,..., xi,..., xn) = \neg xi \wedge F(x1, x2,..., 0,..., xn) \vee xi \wedge F(x1, x2,..., 1,..., xn).$

Ce théorème est appliqué à F pour x1, puis aux deux sous-fonctions obtenues pour x2, et ainsi de suite jusqu‟à xn.

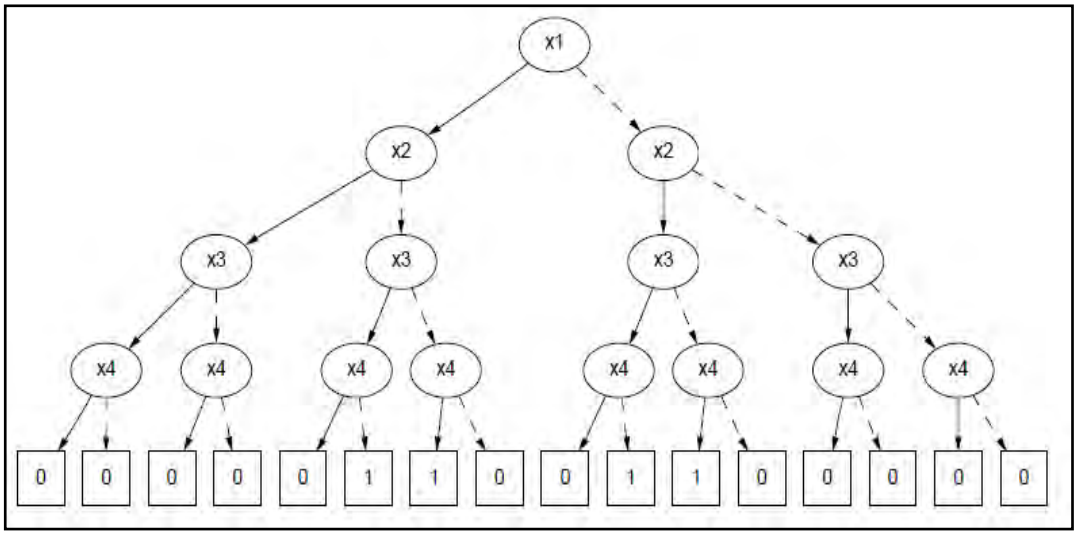


Figure 3.2 – Arbre de décision binaire associé à la fonction $F = (x1 \oplus x2) \wedge (x3 \oplus x4)$

Ces relations peuvent être exprimées sous forme d'arbre de décision binaire comme il est montré sur la figure 3.2. La taille d'un arbre peut être réduite en s'arrêtant dans l'application du théorème de Shannon dès qu'on trouve une constante 0 ou 1(figure 3.3).

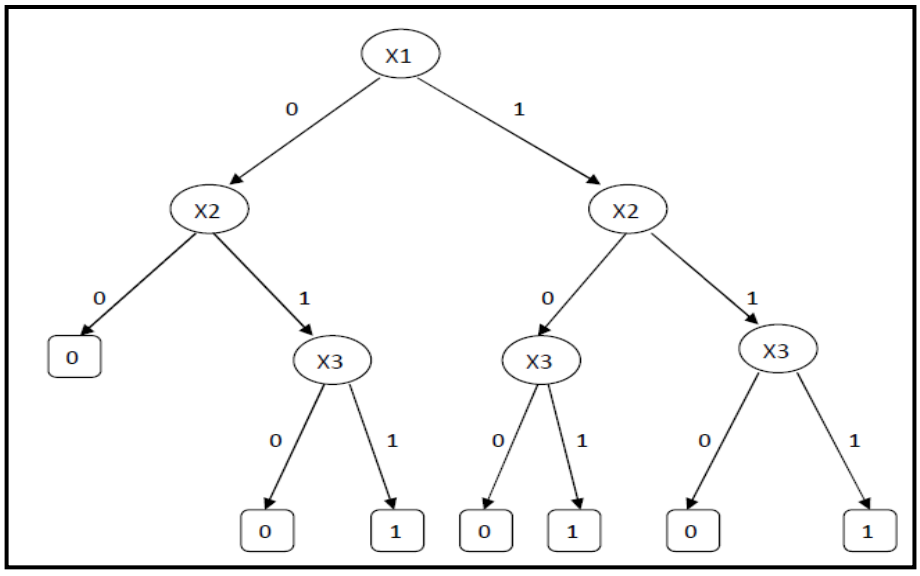


Figure 3.3 – Arbre de décision binaire simplifié associé à la fonction $F = \overline{X_1} \wedge X_2 \wedge X_3 \vee X_1 \wedge X_3$

3.3. Diagrammes de décision binaire

Depuis longtemps, les tables de vérité sont utilisées pour représenter les fonctions booléennes, c'est-à-dire les fonctions de la forme $f : \{0, 1\}^n \rightarrow \{0, 1\}$. Malheureusement, cette représentation est caractérisée par une grande complexité spatiale qui est exponentielle par rapport au nombre de variables d'entrées. En effet, il faut 2^n lignes dans la table de vérité pour représenter une fonction ayant n variables booléennes. Cette complexité a rendu l'utilisation

des tables de vérité limitée. Il est nécessaire ainsi d‟avoir une méthode plus compacte pour représenter les fonctions booléennes. L‟utilisation des diagrammes de décision binaire (BDD) constitue donc une autre façon pour représenter les fonctions booléennes [Somenzi, 2001]. La représentation par diagramme de décision binaire est inspirée de la représentation de Shannon pour les fonctions booléennes.

Mathématiquement, un arbre de décision binaire peut être réduit par transformation en un graphe acyclique orienté appelé Diagramme de Décision Binaire (BDD). De ce fait, Un diagramme de décision binaire peut être vu comme un graphe enraciné, dirigé, acyclique, qui se compose des nœuds de décisions et de deux nœuds terminaux appelées 0-terminal et 1-terminal (figure 3.4). La racine et les nœuds intermédiaires sont indexés et possèdent deux nœuds fils appelés un fils gauche (high) et un fils droit (low). Il faut noter également que la représentation par les diagrammes de décision binaire possèdent exactement la même signification que la représentation de Shannon [Drechsler et al., 1998] [Somenzi, 2001].

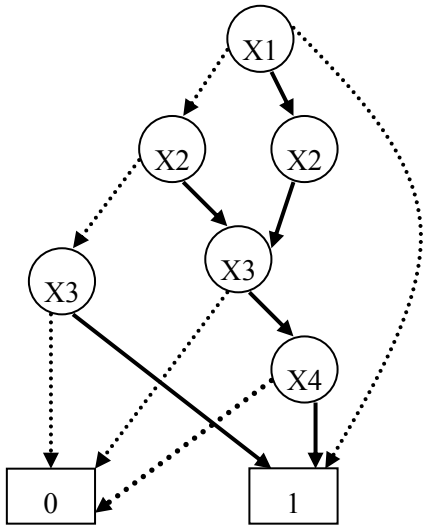


Figure 3.4 – Un BDD représentant la fonction $(x1 \vee x3) \wedge (x2 \Rightarrow x4)$

3.3.1 Représentation d'une formule propositionnelle par un BDD

Un diagramme de décision binaire est une structure de données compacte utilisée pour encoder l‟arbre sémantique associé à une formule, et donc en particulier toutes les interprétations qui la satisfont.

Si φ est une formules du calcul propositionnel, $\varphi\{P \rightarrow \top\}$et $\varphi\{P \rightarrow \bot\}$ sont les formules obtenues en remplaçant P par $\top$ dans φ (resp. P par $\bot$ dans φ) puis en simplifiant par les règles suivantes :

- $\varphi \wedge \top \Rightarrow \varphi$ $\quad \top \wedge \varphi \Rightarrow \varphi$ $\quad \bot \wedge \varphi \Rightarrow \bot$ $\quad \varphi \wedge \bot \Rightarrow \bot$
- $\varphi \vee \bot \Rightarrow \varphi$ $\quad \bot \vee \varphi \Rightarrow \varphi$ $\quad \top \vee \varphi \Rightarrow \top$ $\quad \varphi \vee \top \Rightarrow \top$
- $\neg\top \Rightarrow \bot$ $\quad \neg\bot \Rightarrow \top$ $\quad \bot \rightarrow \varphi \Rightarrow \top$ $\quad \top \rightarrow \varphi \Rightarrow \varphi$
- $\varphi \rightarrow \bot \Rightarrow \neg\varphi$ $\quad \varphi \rightarrow \top \Rightarrow \top$

Un diagramme de décision est un quadruplet (V, l, d, E) où V est un ensemble fini de sommets, l est une application de V dans P∪ {⊤,⊥ }, d est une application de V dans $F_0(P)$ et E ⊆V ×V tel que :

1) G est enraciné et connexe :

$\exists r \in V, \forall v \in V, v \neq r \rightarrow \exists v_1 = r, \ldots, v_n = v \in V, (v_1,v_2), \ldots, (v_{n-1},v_n) \in E$

2) G est acyclique (pas de cycle $(v_1, v_2), \ldots (v_n,v_1) \in E$)

3) Le degré sortant de chaque nœud est 2 ou 0 : $\forall v \in V, |\{v' \in V \mid (v,v') \in E\}| \in \{0,2\}$

Deux graphes de décision G1, G2 sont isomorphes (ce qui est noté G1 ∼ G2) s'il existe une bijection f de V_1 dans V_2 telle que :

- pour tous v_1, v∈ V_1,$(v_1,v) \in E_1$ ssi $(f(v1),f(v)) \in E_2$
- pour tout $v_1 \in V_1$, $l(v_1) = l(f(v_1))$.

Définition 3.1 (Un diagramme de décision binaire) : Un diagramme de décision binaire (BDD) associé à la formule φ sur l'ensemble {P1,...,Pn} de variables propositionnelles est un graphe de décision tel que d(r)= φ et tel que:

–Si v ∈V est tel que l(v) ∈ {⊥,⊤},alors v n'a pas de successeur et d(v) est valide (si l(v)= ⊤) ou non satisfaisable (si l(v)=⊥).

–Si l(v) ∈P, alors v a exactement deux successeurs et les sous-graphes enracinés dans ces deux successeurs ne sont pas isomorphes.

–Si un nœud N est tel que l(N)= Pi, ses successeurs sont des nœuds N1, N2 tels que :

φ1 = d(N1) |=|d(N){Pi→T}et φ2 = d(N2) |=|d(N){Pi→⊥ } et Var(φ1 ∨ φ2) ⊆{Pj |j>i}.

3.3.2 Construction des BDDs

Afin de garantir la canonicité de la représentation, les contraintes suivantes sont imposées :

- chaque variable ne peut apparaître qu'au plus une fois sur chaque chemin entre la racine et une feuille.
- les variables sont ordonnées de telle façon que si un sommet de label x_i a un fils de label x_j alors $ord(x_i) < ord(x_j)$.

On distingue deux stratégies principales pour la construction des BDDs. Ces deux stratégies sont basées sur des approches différentes [Drechsler et al., 1998] et [Pravossoudovitch, 2009]:

3.3.2.1 Méthode top-down

La construction s‟effectue de la racine vers les feuilles (top-down). Cette méthode est utilisée lorsqu‟on démarre d‟une formule algébrique. Pour la construction d‟un BDD, On utilise la formule de Shannon [Sasao et al., 1996].

Par exemple, soit la formule booléenne *f(X1, X2, X3)=* $X1 \wedge \overline{X}2 \wedge \overline{X3} \vee X1 \wedge X3$ avec *ord(X1)<ord(X2)<ord(X3)*. Si l‟on fait l‟expansion par rapport à X1 on obtient la Figure.3.5.a. En faisant l‟expansion par rapport à X2 on obtient la Figure.3.5.b. En faisant l‟expansion par rapport à X3 on obtient la Figure.3.5.c.

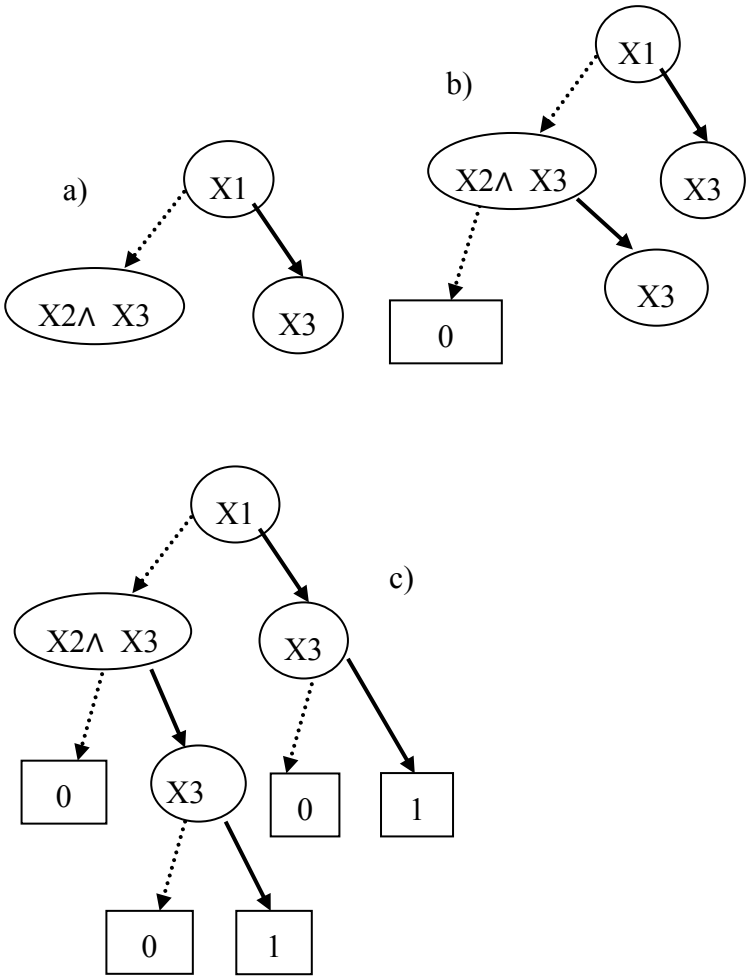


Figure 3.5 – Construction du BDD de f(X1, X2, X3)= $X1 \wedge \overline{X}2 \wedge \overline{X3} \vee X1 \wedge X3$.

3.3.2.2 Méthode bottom-up

Dans cette méthode, on commence par les feuilles et on monte vers la racine (bottom-up) lorsque l‟on part d‟une description structurelle du circuit. Le niveau d‟une formule est défini par :

- les variables d‟entrée sont considérées de niveau 0
- chaque sous-formule *f = g <Op> h* a un niveau égal à *Max(niveau(g),niveau(h)) + 1*

Méthode pour construire le BDD d'une fonction f de n variables :

étape 1) construire les BDDs associés aux variables

étape 2) construire les BDDs des formules de niveau 1 à l'aide de la fonction : appliquer (f1 : BDD, f2 : BDD, opérateur <Op>) : BDD

étape 3) répéter l'étape 2 jusqu'à ce que tous les niveaux soient considérés.

3.4. Diagrammes de décision binaire ordonnés (OBDD)

Nous avons mentionné auparavant que l'une des caractéristiques des BDDs est l'unicité de la représentation, cette dernière dépend de l'ordre des variables utilisé lors de la construction des BDDs. Un tel BDD s'appelle « Ordonné » (OBDD) si les différentes variables apparaissent dans le même ordre sur tous les chemins à partir de la racine.

Définition 3.2 Soient S un ensemble d'états, V un ensemble non-vide de variables booléennes, l'opération $<$ est un ordre total sur les variables de V et var : $S \rightarrow V \cup \{0, 1\}$ est une fonction qui associe au sommet $s \in S$ une variable. Un OBDD sur $(V, <)$ est un graphe orienté acyclique avec un ensemble non-vide S de sommets qui sont de deux types :

- des sommets terminaux s étiquetés par une valeur booléenne, *var(s) = 0* (= *1*), et
- des sommets non-terminaux s étiqueté par une variable booléenne, *var(s)* $\in V$,

qui ont deux fils left(s) et right(s) tels que pour tout sommet t :

$t \in \{left(s), right(s)\} \Rightarrow ((var(s) < var(t))$ ou (t est terminal)).

La deuxième condition exige que chaque variable apparait au plus une seule fois et dans le même ordre pour tous les chemins dont le début est la racine et qui se terminent à une feuille. C'est pour cette raison qu'on dit que le BDD est ordonné (OBDD). La figure 3.6 est un exemple de BDD non ordonné.

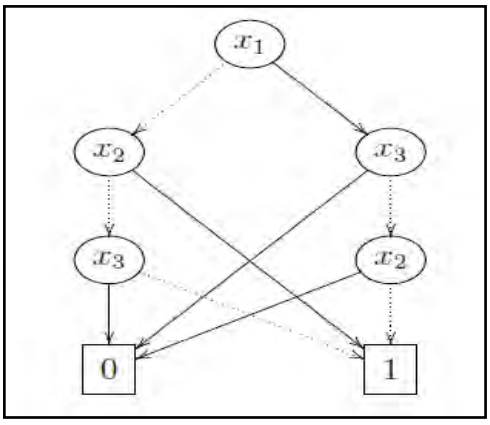


Figure 3.6 – Exemple d'un graphe orienté sans cycle non-ordonné.

Remarquons qu'un OBDD n'est plus nécessairement un arbre contrairement aux BDDs car la version non-orientée du graphe contient des cycles. C'est pourquoi un OBDD est défini comme un graphe orienté sans cycle. Nous discuterons plus tard de l'importance de l'ordre des variables. Cette définition est moins restreignante que celle des BDDs et elle nous permet d'imposer des contraintes aux OBDDs pour que ceux-ci aient une taille plus petite.

3.5. Diagrammes de décision binaire réduite ordonnés (ROBDD)

Il est possible de transformer un BDD en un graphe qui représente la même fonction booléenne mais qui est de taille inférieure. Un BDD peut être réduit en utilisant trois règles définies par Bryant [Bryant, 1992]. Ces trois contraintes sont appelées les règles de réduction parce qu'à chacune correspond une transformation du OBDD pour que celui-ci respecte la contrainte. Un OBDD est réduit lorsque toutes les règles de réduction ont été appliquées et qu'aucune d'elles ne peut encore changer le graphe (figure 3.7). Les OBDD réduits sont souvent appelés ROBDD de l'anglais *Reduced OBDD*. Dans ce qui suit, lorsque nous parlerons d'un BDD, il sera réduit et ordonné (ROBDD). Il est important de noter que pour un ordre donné de variables, le graphe de décision binaire minimal est unique. Cette unicité peut évidemment être utilisée pour montrer l'équivalence de deux expressions logiques [Pravossoudovitch, 2009].

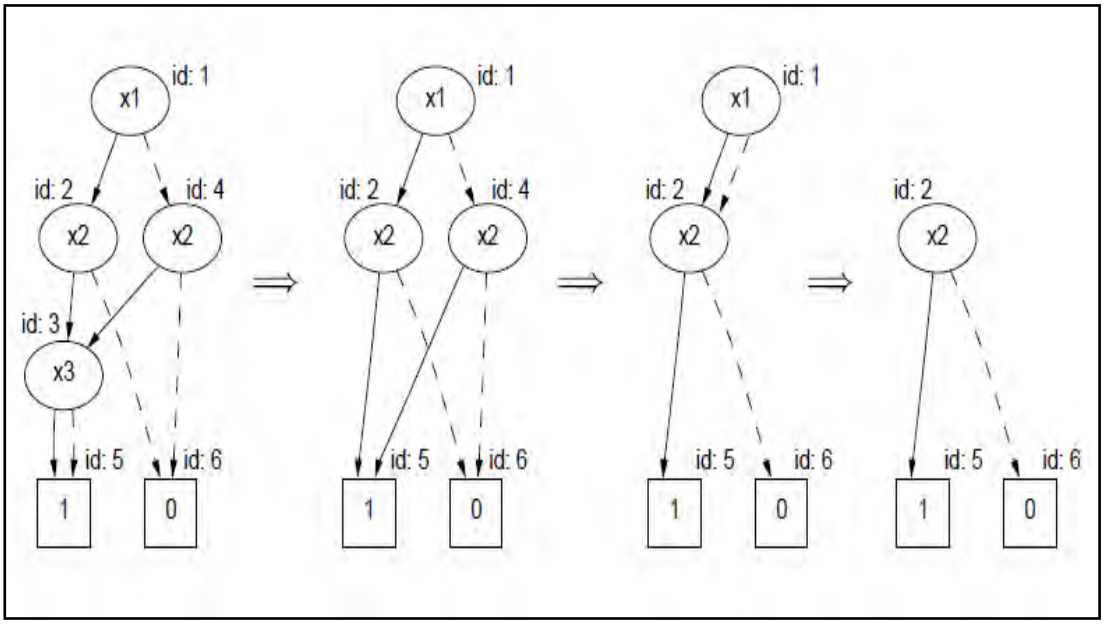


Figure 3.7– la réduction d'un ROBDD.

Les trois règles de réduction d'un BDD sont les suivantes :

Régle 1 : Pour tous nœuds terminaux s et t,

var(s) = var(t) ⇒ s = t

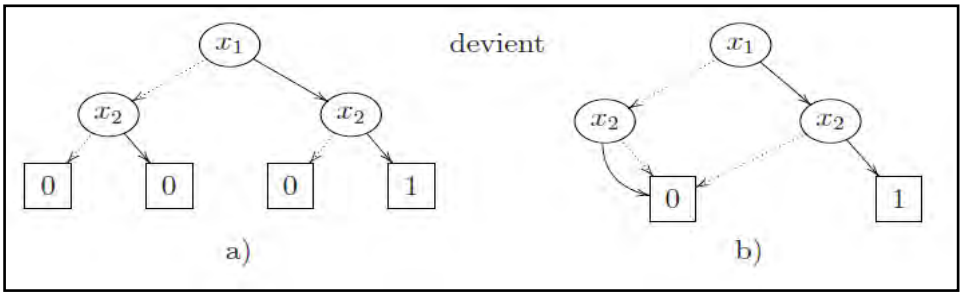


Figure 3.8 – Exemple de réduction par la règle 1.

Régle 2: Pour tout nœud s,

left(s) ≠right(s)

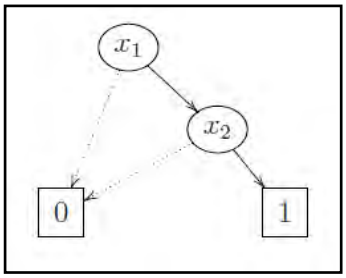


Figure 3.9 – Exemple de réduction par la règle 2.

Régle 3: Pour tous nœuds s et t,

((var(s) = var(t)) ∧ (left(s) = left(t)) ∧ (right(s) = right(t))) ⇒ s = t

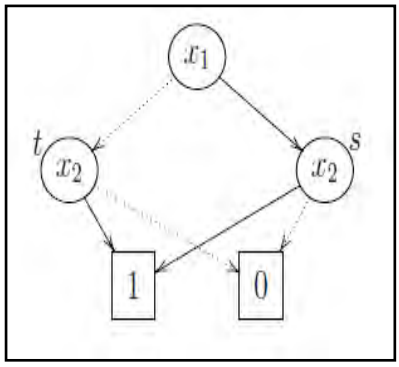


Figure 3.10– Exemple de réduction par la règle 3.

Remarque :

- La complexité de la procédure de réduction d"un BDD est O(nlog(n)).
- La structure du BDD et en particulier sa taille dépend de l"ordre dans lequel sont considérées les variables. Ainsi, pour deux ordres différents, les BDDs correspondants peuvent être de tailles considérablement différentes.

L'algorithme suivant sert à réduire un BDD. La réduction se fait en traversant le BDD de bas en haut en donnant des étiquettes à chaque nœud. L'étiquette du nœud v est notée id(v) et E est l'ensemble des étiquettes.

Algorithme 3.1 : Schéma de l'algorithme de réduction des BDDs

```
Début
  Pour toute feuille v
   Si var(v) = 0 alors id(v) = 0
   Si var(v) = 1 alors id(v) = 1
  Fin Pour tout
  Pour toute hauteur à partir de la hauteur 1
    Pour tout sommet v
     Si id(left(v)) = id(right(v)) alors id(v) = id(left(v))
     Sinon
       Pour tout v'
        Si var(v) = var(v'), id(left(v')) = id(left(v)) et
        id(right(v')) = id(right(v))alors
        id(v) = id(v')
        Sinon
        id(v) = x où x ∈ E
        E := E \ {x}
       Fin Pour tout
     Fin Si
    Fin Pour tout
  Fin Pour tout
Fin
```

3.6. Ordre des variables et taille des OBDDs

Nous avons évoqué précédemment l'importance de trouver un bon ordre des variables pour diminuer la taille des OBDDs. Il est très important de prendre en considération l'ordre de variables à utiliser quand on utilise les BDDs en pratique. Par exemple, la fonction $f(x1, x2, x3, x4) = (x1 \vee x3) \wedge (x2 \Rightarrow x4)$ avec l'ordre de variable x1 < x2 < x3 < x4 est

représentée par l'OBDD de la figure 3.11 à gauche. Par contre, si pour la même fonction nous choisissons l'ordre x1 < x3 <x2 < x4, nous obtenons un OBDD de taille inférieure, celui de la figure 3.11 à droite. D'une façon générale, on peut avoir un nombre exponentiel d'ordres possibles (permutations). Le problème d'ordre de variables d'un BDD a été démontré NP-difficile pour un certain nombre de fonctions booléennes (table 3.1) [Bollig et al., 1996]. Le meilleur algorithme exact connu pour résoudre ce problème s'exécute en $O(3^n \cdot n^2)$ où n étant le nombre de variables de l'OBDD. De ce fait, l'utilisation d'un tel algorithme pour des grandes instances est impraticable [Friedman et al, 1987]. Dans ce qui suit nous allons présenter quelques méthodes utilisées pour la résolution de ce problème.

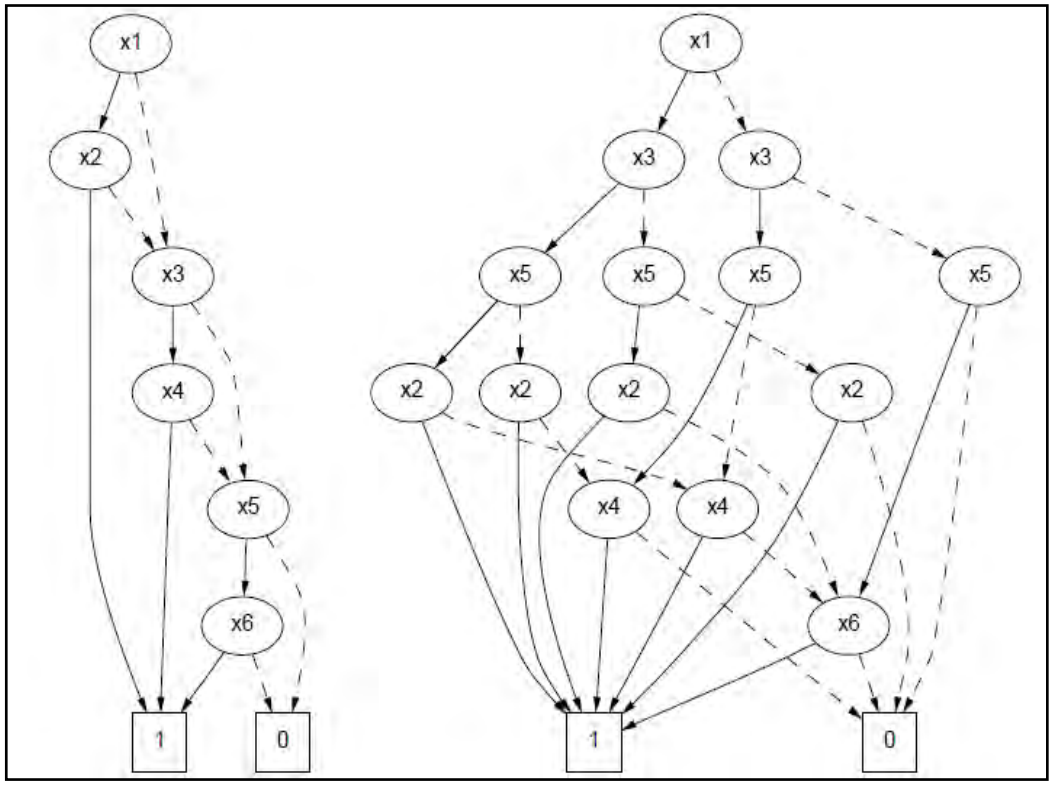


Figure 3.11 – Deux BDD différents représentant la fonction $(x1 \wedge x2) \vee (x3 \wedge x4) \vee (x5 \wedge x6)$, à gauche l'ordre des variables est : x1, x2, x3, x4, x5, x6 ; et à droite l'ordre est : x1, x3, x5, x2, x4, x6.

Table 3.1: Comportement asymptotique de 3 classes de circuits Booléennes.

Classe de Fonction	**Meilleure Complexité**	**Pire Complexité**
Symétrique	Linéaire	Quadratique
Addition entière (n'importe quel bit)	Linéaire	Exponentielle
Multiplication entière (bits moyens)	Exponentielle	Exponentielle

3.7. Méthodes de minimisation d'un ROBDD

Nous avons mentionné précédemment l'importance de trouver un bon ordre des variables pour minimiser la taille des OBDDs. Nous rappelons également qu'il est NP-Difficile de trouver l'ordre des variables optimal et ainsi, pour résoudre en pratique ce problème, nous ne pouvons que trouver des méthodes heuristiques qui donnent de bonnes solutions sans toutefois être optimales. Il existe quatre classes principales de méthodes utilisées pour résoudre le problème d'ordre de variables d'un BDD. La plupart des heuristiques utilisent des informations sur les variables qui constituent la formule booléenne qu'on veut représenter. Il existe des règles empiriques à suivre lors de la recherche du meilleur ordre, par exemple : Prendre en priorité

- Variables qui contrôlent le plus la fonction
- Variables apparaissant sous la même forme dans tous les monômes
- Variables constituant un monôme à elles seules
- Variables d'occurrence maximale

3.7.1. Méthodes exactes

La méthode la plus directe pour obtenir l'ordre optimal des variables est de tester toutes les permutations possibles des variables afin de trouver celui qui donne le plus petit BDD. En supposant qu'il existe n variables en entrée, nous devons tester n! permutations différentes. dans le pire des cas, la complexité de BDDs est illustrée par une approche simpliste de $O(n!2^n)$, mais cela n'est pas possible que pour les petits circuits. D'autre part, une approche plus sophistiquée utilise des techniques de la programmation dynamique lors de l'exécution de telles recherches. Cette méthode découle de la théorie que, compte tenu d'une division sur la largeur du BDD, la taille du BDD en dessous de la coupe est indépendante de l'ordre de l'ensemble des variables au-dessus de la coupe [Friedman et al., 1987]. Même si cette technique permet de réduire considérablement la complexité de trouver l'ordre optimal des variables d'un BDD, elle est encore trop complexe pour toutes les applications temps réel. En effet, il s'est avéré que le problème de trouver l'ordre de variables optimal pour un BDD est NP-complet. En conséquence, les chercheurs ont commencé à se tourner vers des algorithmes basés sur des heuristiques.

3.7.2. Ordonnancement dérivé des circuits

L'une des principales méthodes utilisées sur les OBDDs représentant des circuits est la recherche en profondeur sur le circuit. L'algorithme s'exécute comme suit : le parcours débute par la sortie du circuit et se complète par une recherche en profondeur, comme sur un graphe. L'ordre des variables est déterminé par l'ordre de rencontre des entrées. Dans [Berman, 1989] et [McMillan ,1992] on a calculé la limite théorique supérieure de plusieurs

classes de circuits basés sur les propriétés structurelles de leurs réseaux logiques. Ces résultats constituent la base théorique pour les efforts qui ont exploité les structures de circuits en vue de trouver le bon ordre des variables d‟un BDD. Par ailleurs on a également introduit la notion de la largeur de la réalisation d'un graphe logique qui repose sur l‟ordre topologique du graphe. Ainsi on a exprimé la limite supérieure d‟un BDD en fonction de la largeur de la structure graphique du circuit.

À titre d‟exemple si nous souhaitons déterminer un ordre de variable efficace pour l‟OBDD qui représente le circuit de la figure suivante. En appliquant l‟heuristique classique, l‟ordre trouvé pourrait être x1 < x2 < x3 ou x3 < x2 < x1 selon l‟implémentation de l‟algorithme de recherche en profondeur.

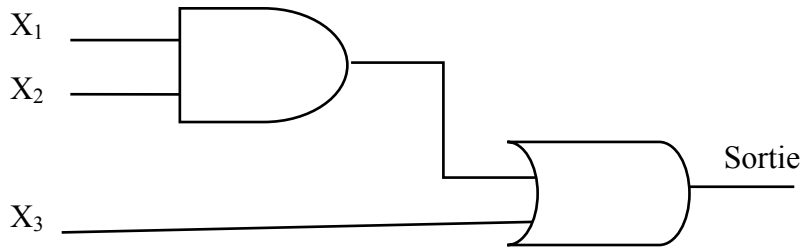


Figure 3.12 – Exemple de circuit.

3.7.3. Les algorithmes heuristiques statiques

Les algorithmes d‟ordonnancement heuristique sont les premières techniques qui ont réussi à permettre aux chercheurs d'utiliser les BDDs pour la manipulation des grands circuits industriels. La plupart des algorithmes heuristiques exploitent les propriétés structurelles des graphes logiques des circuits numériques afin d'en tirer les bons ordres des variables d‟un BDD. L'objectif principal de ces algorithmes est d'éviter l'explosion combinatoire. Voici quelques-uns des algorithmes heuristiques qui ont donné de bons résultats:

3.7.3.1 Ordonnancement basé sur les paires de variables

Quand une fonction booléenne est exprimée avec la formule de somme des produits (SOP), le couplage (binateness) est une mesure de la fréquence d'une variable qui apparaît explicitement aussi bien positive que négative dans l'expression. Cet algorithme met les variables les plus couplés en haut du BDD de telle sorte que le nombre total de termes dans les expressions de SOP de ses co-facteurs sont réduits au minimum. L‟heuristique principale de cet algorithme est que la somme des produits avec moins de termes de produits, va réduire la taille du sous-graphe enraciné au sommet de son représentant.

3.7.3.2 Méthode de recherche en profondeur (Depth-First Search)

C'est un algorithme de recherche qui progresse à partir d'un sommet S en s'exécutant récursivement pour chaque sommet voisin de S. Contrairement à l'algorithme de parcours en

largeur, cet algorithme explore en fait à fond tous les chemins un par un. Pour chaque sommet, il prend le premier sommet voisin jusqu'à ce qu'un sommet n'ait plus de voisins (ou que tous ses voisins soient marqués), et revient alors au sommet père. [Malik et al., 1988] propose une simple traversée de DFS du réseau logique utilisé pour obtenir l'ordre de variables du BDD. Cette heuristique est particulièrement efficace pour l'obtention d'un ordre des variables non-entrelacé. Ainsi, cette méthode donne des ordres de variables optimaux pour des circuits simples d'arithmétique comme les additionneurs. Cette technique peut être considérée comme un algorithme de recherche d'une topologie minimale pour un circuit.

3.7.3.3 Méthode de rang de variables

Cette heuristique tente de classer les variables en fonction de leur influence sur la valeur de sortie (s) de la fonction booléenne. Un système d'attribution de poids est obtenu en parcourant le graphe du circuit à partir des sommets de sortie jusqu'aux sommets d'entrée, ce qui tente de caractériser le degré d'importance des entrées. Les variables avec de grands poids sont placées en haut du BDD.

3.7.4. Les méthodes d'ordonnancement dynamique

Actuellement, les algorithmes qui permettent de changer l'ordre des variables dynamiquement au cours de la construction d'un BDD sont les plus efficaces des algorithmes d'ordonnancement de variables des BDDs. L'ordonnancement dynamique des variables est une technique par laquelle une variable est passée successivement à chaque position dans la liste des ordres d'un BDD avant qu'elle ne soit définitivement attribuée à sa meilleure position qui donne le plus petit BDD [Rudell, 1993] [Panda et al., 1995]. Le plus connu et efficace de ces algorithmes est sans doute l'algorithme de *sifting* qui consiste à changer l'ordre de deux variables en les permutant. En appliquant cette technique à plusieurs paires de variables on arrive à choisir un ordre qui diminue la taille de l'OBDD. Le sifting est apparu comme une excellente heuristique pour atteindre un bon compromis entre les exigences du temps CPU et la qualité des résultats trouvés [Rudell, 1993].

L'algorithme *windows-sifting* est un raffinement de l'algorithme de sifting dans lequel une petite fenêtre de k variables consécutives est créée et le sifting est effectué uniquement dans cette fenêtre. La fenêtre est avancée à maintes reprises jusqu'à ce que toute la longueur du BDD soit couverte. Une autre optimisation de cet algorithme qui accélère le processus de sifting, consiste à réduire le nombre de sift requis dans l'étape de groupage de variables.

Il existe des algorithmes qui permettent de permuter deux variables très rapidement et ce, en gardant l'OBDD réduit. Ces techniques donnent de bons résultats mais elles sont toutefois assez coûteuses en temps de calcul si on décide de tester plusieurs permutations, ce qui est nécessaire si on veut réduire largement la taille de l'OBDD. Les heuristiques dynamiques sont souvent utilisées dans la vérification des circuits électriques.

3.8. Manipulation des OBDDs

Les opérations logiques applicables sur les OBDDs ont été définies par Bryant. Ces opérations sont les opérations de complémentation, de test d'implication, de ET logique, de OU logique, et de OU Exclusif. L'algorithme permettant de déterminer l'OBDD résultant d'une opération logique entre deux fonctions est basé sur l'application récursive du théorème de Shannon. L'OBDD résultant d'une opération logique entre deux fonctions peut être conçu à partir des deux OBDDs originaux en appliquant la procédure suivante :

Considérer les sommets racines des OBDDs. En fonction de la nature de ces deux sommets, appliquer récursivement l'une des règles suivantes (R1, R2, et R3). Le processus est réitéré jusqu'à la génération des sommets terminaux.

Règle 1 : Si les deux sommets Sf et Sg sont des sommets terminaux alors le sommet résultant est un sommet terminal de valeur Valeur (Sf) <Op> Valeur(Sg) (f et g deux fonctions).

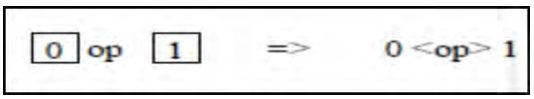


Figure 3.13 – Illustration la règle 1 de combinaison sur un OBDD.

Règle 2 : Si le sommet Sf est un sommet terminal mais pas le sommet Sg, créer le sommet Sg en lui associant comme fils droit le résultat de la comparaison (Sg, fils droit de Sf) et comme fils gauche le résultat de la comparaison (Sg, fils gauche de Sf).

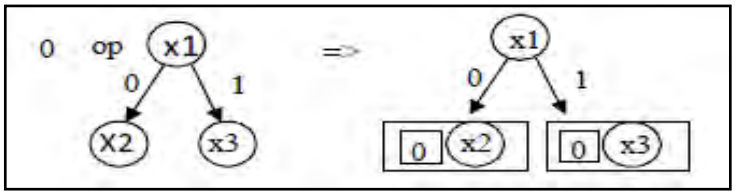


Figure 3.14 – Illustration la règle 2 de combinaison sur les OBDDs

Règle 3 : Si ni le sommet Sf ni le sommet Sg ne sont des sommets terminaux alors :

Règle 3.1 : Si Sf et Sg représentent la même variable, créer un sommet représentant la variable en lui associant comme fils droit le résultat de la comparaison (fils droit de Sg, fils droit de Sf) et comme fils gauche le résultat de la comparaison (fils gauche de Sg, fils gauche de Sf).

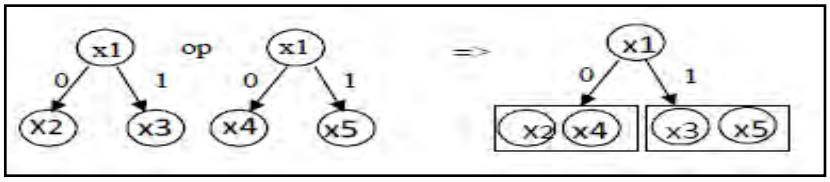


Figure 3.15 – Illustration la règle 3.1 de combinaison sur les OBDDs.

Règle 3.2 : Si Sf et Sg ne représentent pas la même variable, créer un sommet S représentant la variable intervenant la première dans l'ordonnancement considéré (S=min [ord(Sf), ord(Sg)] en lui associant comme fils droit (gauche) le résultat de la comparaison entre le fils droit (gauche) de min [ord(Sf), ord(Sg)] et l'autre sommet (max [ord(Sf), ord(Sg)]).

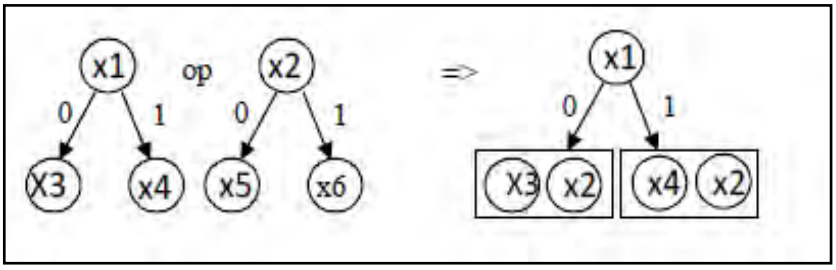


Figure 3.16 – Illustration la règle 3.2 de combinaison sur les OBDDs.

Remarque: Etant données deux fonctions F et G dont les BDDs respectifs ont n et m sommets. En termes d'appels récursifs, la complexité de la procédure d'application d'une opération entre ces deux BDDs est $2*n*m$.

Remarque: Cette méthode permet (outre les opérations sur entrées) :

- de construire un BDD de façon ascendante,
- de calculer l'inverse d'une fonction en faisant "f $\oplus$1"
- de calculer une implication f1 => f2 <-> not (f1 . not(f2))

3.8.1. Méthodes utilisées pour la manipulation des OBDDs

- ***Négation*** Ayant l'OBDD O représentant une fonction f, pour déterminer l'OBDD de $\neg$f nous n'avons qu'à échanger les deux feuilles 0 et 1 de O. Il est clair que cette opération donne bien la négation voulue et que l'OBDD est réduit.
- ***La méthode apply*** est utilisée pour calculer des opérations binaires, comme le « *et* » et le *ou* logiques, entre deux OBDDs. Etant donnés deux OBDDs Bf et Bg, représentant les fonctions f et g, l'appel de la fonction apply(op,Bf ,Bg) avec op $\in$ {$\vee$,$\wedge$,$\oplus$, ...}, retourne l'OBDD réduit représentant la fonction booléenne f op g.
- ***Restriction de variables*** **:** L'algorithme *restrict* sert à fixer la valeur d'une variable dans l'OBDD. Ainsi, restrict(B, xi, b) retourne l'OBDD B dans lequel la valeur de la variable xi a été fixée à la valeur booléenne b. Cette opération se fait facilement en dirigeant tous les arcs pointant sur chaque nœud n étiqueté par xi vers le fils gauche (arc pointillé) de n si b = 0 et vers le fils droit (arc plein) de n si b = 1.

- ***Abstraction de variables*** **:** Un autre algorithme utile est l'algorithme *exists*. Celui-ci sert à supprimer des variables dans un OBDD en considérant les deux valeurs possibles pour ces variables, c'est-à-dire, exists(Bf , xi) = f([xi := 0]) ∨ f([xi := 1]). Enoncé ainsi, il est facile de voir que calculer apply(∨,restrict(B, xi, 0),restrict(B, xi, 1)) nous donne exists(B, xi), il est cependant possible d'améliorer l'algorithme en remplaçant directement dans l'OBDD les nœuds étiquetés xi par le résultat de l'application du ∨ aux deux fils des nœuds.
- ***La Fonction ITE :*** On peut manipuler simplement les ROBDDs à l'aide de la fonction *ITE (f,g,h) (if then else)* définie par: ***ITE(f,g,h) = fg + f'h*** *(i.e. If(f) Then (g) Else h).* De plus toutes les opérations habituelles peuvent être traduites en termes de l'opérateur ITE :

Table 3.2 : L'équivalence des opérations habituelles en termes de l'opérateur ITE.

Table	Nom	Expression	Forme
0000	0	0	0
0001	AND(f,g)	f.g	ite(f,g,0)
0010	f > g	f.g'	ite(f,g',0)
0011	f	f	f
0100	f < g	f'.g	ite(f,0,g)
0101	g	g	g
0110	XOR(f,g)	f ⊕g	ite(f,g',g)
0111	OR(f,g)	f + g	ite(f,1,g)
1000	NOR(f,g)	(f + g)'	ite(f,0,g')
1001	XOR(f,g)	(f ⊕g)'	ite(f,g,g')
1010	NOT(f,g)	g'	ite(g,0,1)
1011	f ≥ g	f + g'	ite(f,1,g')
1100	NOT(f)	f'	ite(f,0,1)
1101	f ≤ g	f' + g	ite(f,g,1)
1110	NAND(f,g)	(f.g)'	ite(f,g',1)
1111	1	1	1

3.9. Applications directes des OBDDs

Les BDDs sont largement utilisés dans plusieurs domaines. Le fait qu'ils soient canoniques fait qu'ils sont très utiles pour la vérification d'équivalence, de la tautologie, ainsi que des problèmes de satisfiabilité. Parmi les applications des BDDs les plus répandues, on peut citer:

3.9.1. Les tests

Quand on travaille dans une logique, on s'intéresse à l'équivalence sémantique, à la satisfiabilité et à la validité des formules. Voyons comment ces tests peuvent être faits sur les OBDDs. La canonicité des OBDDs est essentielle pour pouvoir effectuer certains tests de façon très efficace : si deux OBDDs représentant des fonctions f1 et f2 qui ne sont pas

isomorphes pour un ordre donné des variables, nous pouvons déduire immédiatement que f1 et f2 ne sont pas sémantiquement équivalentes. Par exemple, si nous voulons savoir si $f1 = x0 \wedge (x1 \vee x2)$ est sémantiquement équivalente à $f2 = (x0 \wedge x1) \vee x2$, nous pouvons nous fixer un ordre des variables et construire l'OBDD représentant chacune de ces fonctions. La figure 3.17-a représente l'OBDD de la fonction f1 et la figure 3.17-b représente l'OBDD de la fonction f2. Puisque ces graphes ne sont pas isomorphes, nous pouvons déduire que la fonction f1 est différente de la fonction f2.

Les tests de satisfiabilité et de validité peuvent aussi être faits efficacement, déterminer si une formule est satisfiable revient seulement à vérifier que l'OBDD correspondant ne consiste pas seulement en la feuille étiquetée 0. Pour le problème de validité, il faut que tous les sommets terminaux (feuilles) de l'arbre de décision binaire vaillent 1 (aucun sommet terminal à 0). Si ce n'est pas le cas, la formule n'est pas valide. Par exemple, la fonction :

$f(x1, x2, x3) = (\neg x1 \wedge x2 \wedge x3) \vee (x1 \wedge x2 \wedge x3) \vee (x1 \wedge \neg x3) \vee \neg x2$ est satisfiable car tous les sommets terminaux de l'arbre de décision binaire sont à 1 (Figure 3.18).

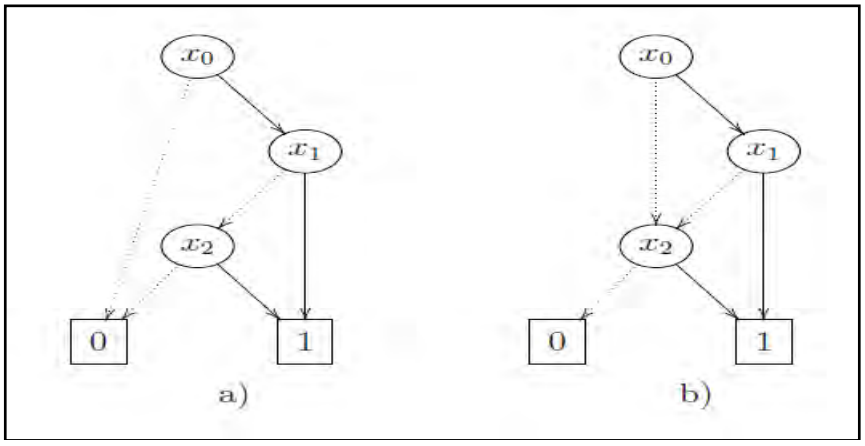


Figure. 3.17– OBDD des fonctions f1 et f2 avec l'ordre x0 < x1 < x2.

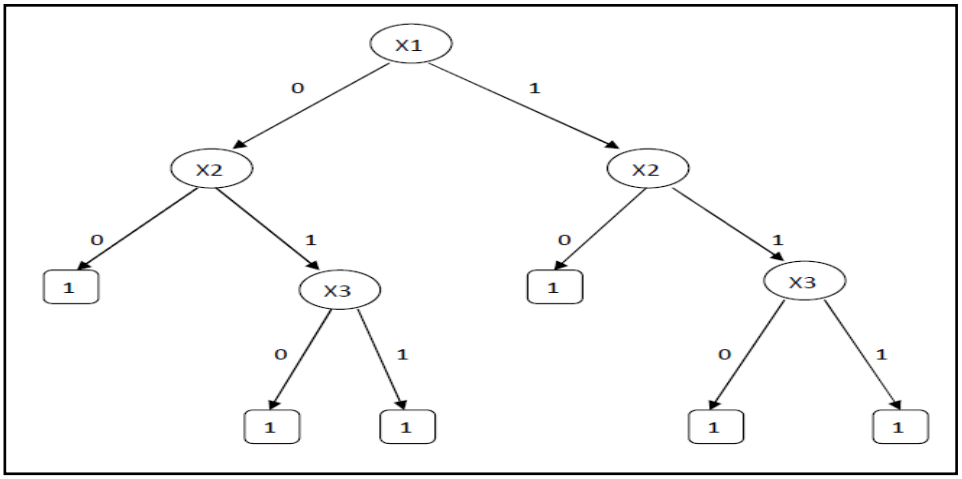


Figure 3.18– Arbre de décision binaire de la fonction pour une formule satisfiable.
$f = (\neg x1 \wedge x2 \wedge x3) \vee (x1 \wedge x2 \wedge x3) \vee (x1 \wedge \neg x3) \vee \neg x2$

On sait que les problèmes de satisfiabilité et de validité d'une formule booléenne sont NP-Complet. Ici les solutions à ces problèmes sont très faciles à obtenir car on suppose qu'on a l'OBDD de la formule. Cependant, obtenir cet OBDD peut être très difficile.

3.9.2. Evaluation d'une fonction représentée sous forme d'OBDD

Etant donnée une assignation des variables d'entrées d'une fonction *f*, la valeur de la fonction *f* peut être trouvée en traversant l'OBDD depuis la racine jusqu'à une feuille, en empruntant la branche gauche ou droite de chaque sommet selon la valeur de la variable d'entrée correspondante. Dans le pire des cas, l'OBDD correspondant à une fonction f de n variables est composé de 2^n-1 sommets. De plus, le nombre de branches entre la racine et une feuille est au plus égal à n. La complexité d'une procédure d'évaluation d'une fonction, c'est à dire de traverser d'un OBDD depuis sa racine jusqu'à une feuille est O(n). La représentation sous forme d'OBDD d'une fonction *f* permet de trouver rapidement une combinaison des entrées telles que *f=0* ou telle que *f=1*. Donc, la procédure permettant de trouver une solution x de f(x) = a (a = 0 ou 1) est en O(n) [Knuth, 2009] [Pravossoudovitch, 2009].

3.9.3. Modèle Checking

Vu leur représentation canonique, les diagrammes de décision binaire ont connu un indéniable succès pour la vérification des systèmes finis. McMillan a utilisé les diagrammes de décision binaire ROBDD pour représenter symboliquement l'espace d'états dans le modèle de la machine d'états finis [McMillan ,1992]. Il a présenté aussi une procédure efficace de calcul de point fixe pour déterminer si le modèle de la machine d'états finis valide des propriétés logicotemporelles. Cette procédure a rendu possible la vérification de circuits de tailles importantes. Nous expliquons maintenant le fonctionnement d'un modele-checking basé sur un ROBDD qui décide si une formule booléenne est satisfaite dans un état *E* du système. Ces techniques sont souvent plus efficaces pour les systèmes présentant un fort degré de concurrence mais font intervenir des mécanismes trop lourds pour des systèmes séquentiels [Malik et al., 1988].

Dans les méthodes symboliques, les ensembles d'états sont représentés par des formules booléennes (i.e leurs fonctions caractéristiques). D'un point de vue technique, ces formules booléennes sont codées par des diagrammes de décision binaires qui permettent une implémentation efficace des opérations ensemblistes, union, intersection, quantification, etc. La représentation des ensembles sous forme de BDD a les propriétés suivantes :

- Pour un ordre total des variables d'état, E est une représentation canonique de *E*.
- Le nombre de nœuds (et d'arcs) de E ne dépend pas de la cardinalité de *E,* et en général, sa représentation en mémoire est de beaucoup plus petite taille que l'énumération de *E*.

- Les opérations ensemblistes sur *E* et *F* sont réalisées par des parcours combinés de E et F de complexité quadratique (opérateur APPLY).
- Les opérations *Post(E,* t) et *Pre(E,* t) (pour t un sous-ensemble de *T* l'ensemble des transitions) sont réalisées par composition d'opérations élémentaires sur les BDD (une substitution de variables, une conjonction (APPLY), une quantification existentielle).

Malheureusement, cette représentation présente une limitation majeure: le problème de l'ordre des variables. Si la taille du BDD ne dépend pas de la cardinalité de l'ensemble qu'il représente, cette taille dépend grandement de l'ordre d'occurrence des variables dans le BDD.

Le fonctionnement d'un model-checker symbolique par BDD consiste en l'obtention de points fixes pour déterminer les états accessibles ainsi que ceux qui satisfont une ou plusieurs propriétés. Ces points fixes sont obtenus à l'aide de deux transformateurs de prédicats associés au franchissement des transitions : Le premier (Post) détermine, pour un ensemble d'états donnés, l'ensemble des états successeurs tandis que le second (Pre) est l'opération réciproque du premier. Le schéma de l'algorithme de calcul des états accessibles est le suivant :

```
bdd A = Empty;
bdd NA = initial ;
while ( A =! NA ) {
A = NA ;
NA =Union (A, Post(A));
}
```

Dans cet algorithme : *initial* est le codage en BDD du singleton contenant l'état initial, *Empty* est la constante qui code l'ensemble vide i.e la fonction identiquement fausse. Les opérations d'affectation et de comparaison entre BDD ont un coût constant négligeable. Cependant, l'opération *Union* a un coût polynomial faible par rapport à la taille de ses opérandes. Toute la complexité de cet algorithme réside donc dans le calcul de la fonction Post.

SMV est le premier vérificateur de modèle qui a prouvé l'efficacité de cette technique sur un espace d'états de l'ordre de 10^{100}. Ce système logiciel, développé par K. McMillan, a illustré les possibilités et la puissance des outils de vérification basées OBDD en ce qui concerne plusieurs aspects. En employant des algorithmes basés sur les OBDDs, SMV permet de vérifier les systèmes séquentiels vis à vis les spécifications exprimées logique temporelle CTL. Le langage d'entrée de SMV offre la possibilité pour définir les systèmes qui doivent être analysés à différents niveaux d'abstraction.

Le deuxième model-ckecking basé ROBDD le plus connu est VIS (Verification Interacting with Synthesis). VIS unifie la vérification, la simulation et la synthèse des machines à états finis. La figure 3.19 montre l'architecture du système de VIS. Il y a quatre composants essentiels: l'interface, le noyau de vérification, le noyau de synthèse, et le paquet fondamental d'OBDD. L'interface fournit des routines pour transformer de divers formats bien connus

d'entrée en description interne de réseau hiérarchisé en particulier, aussi des langages de haut niveau évolués comme le langage commerciale répandu *Verilog* peuvent être transformés en format interne. Afin de représenter les fonctions de commutation d‟occurrence, VIS utilise les OBDDs comme structure de données internes. Tous les accès au niveau de la manipulation booléenne sont effectués par les interfaces uniformes qui correspondent exactement à des opérations et fonctions sur les BDDs. En raison de la séparation claire des algorithmes de vérification du niveau fondamental de la manipulation booléenne il est possible de choisir n‟importe qu‟elle bibliothèque de manipulation d'OBDD (CUDD, BuDDy). Le noyau de vérification fournit des algorithmes pour l'analyse de l‟accessibilité, la vérification d'équivalence, et la vérification basée modèle. En plus de la vérification basée modèle utilisant CTL, des contraintes d'équitabilité, qui ne peuvent pas être exprimées en CTL, peuvent également être tenues en compte. D'une part, il contient plusieurs nouveaux algorithmes pour la synthèse des circuits qui ont été spécifiquement développés pour le système VIS. D'une part, le noyau de synthèse inclut les interfaces bien définies au SIS de progiciel (synthèse interactive séquentielle). Ce progiciel, qui est également basé sur OBDD, a été développé dans Berkeley et fournit de nombreux outils pour la synthèse et l'optimisation des circuits séquentiels.

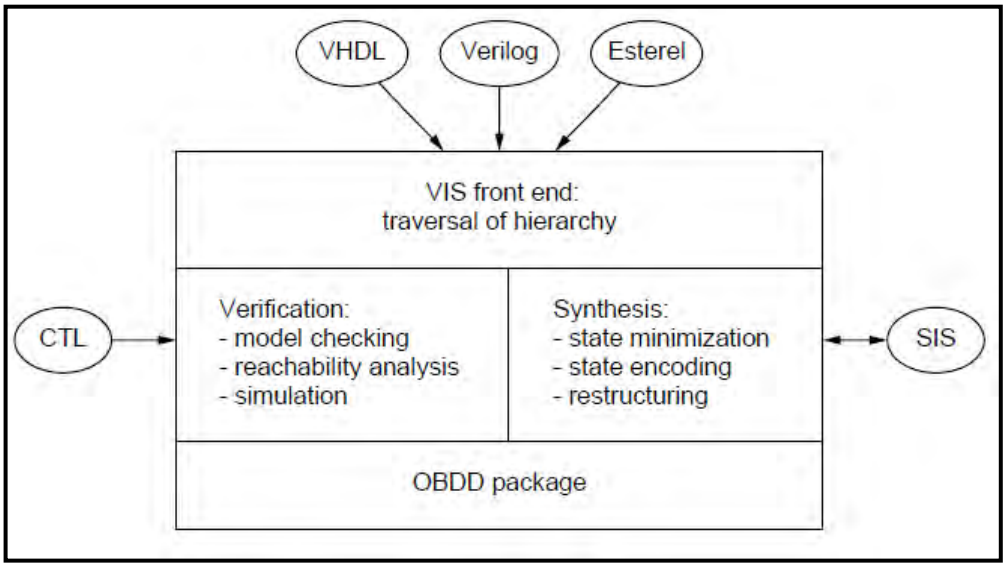


Figure 3.19 – L'architecture du système figure VIS.

3.9.4. Les cofacteurs

Pour obtenir le cofacteur par rapport à x_i, il suffit de supprimer le sommet x_i et de lier les sommets pointant sur x_i au fils droit de x_i. Pour obtenir le cofacteur par rapport à $\neg x_i$ il suffit de supprimer le sommet $\neg x_i$ et de lier les sommets pointant sur x_i au fils gauche de x_i (Figure 3.20) [Knuth, 2009].

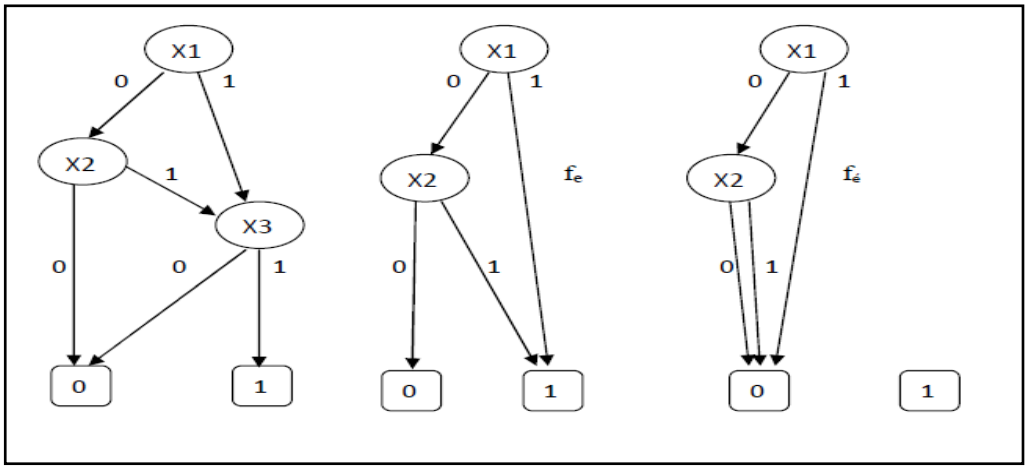


Figure 3.20– OBDD des cofacteurs.

3.10. Implantation informatique des OBDD

La représentation informatique la plus simple d‟un OBDD est un tableau tel que chaque ligne comporte 3 champs : un champ « Identificateur » permettant d‟identifier le sommet dans l‟OBDD, un champ Arc_0 donnant l‟adresse du fils gauche du sommet en question et un champ Arc_1 donnant l‟adresse du fils droit.

Table 3.3: Représentation d'un OBDD

Identificateur	Arc_0	Arc_1
0	-	-
1	-	-
Variable sommet i	Adresse sommet gauche	Adresse sommet droit

Soit une fonction f (x1, x2, x3) = ¬x1 ∨ ¬ x3 ∨ x2 ∧ x3 (Figure 3.21).

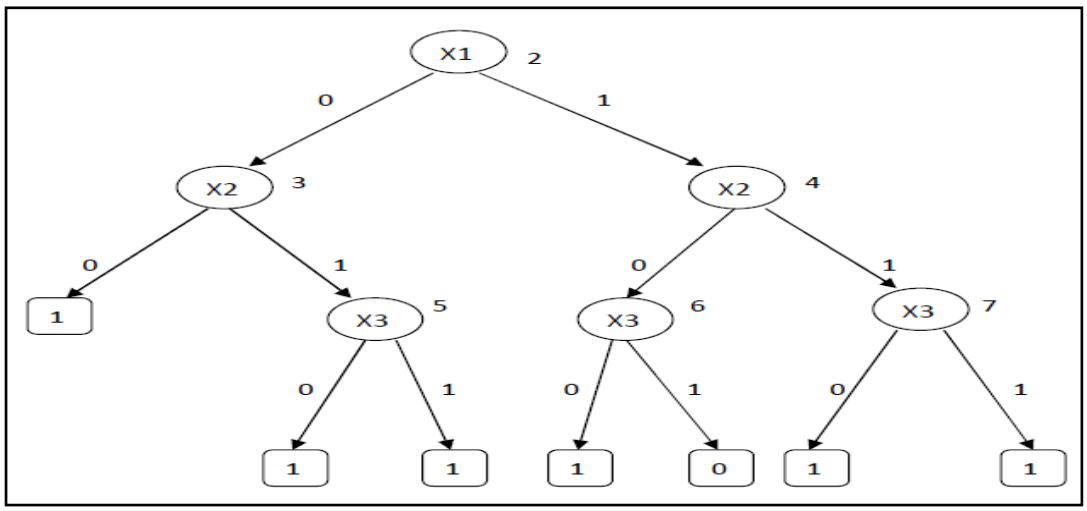


Figure 3.21– BDD de la fonction f =¬x1 ∨ ¬x3 ¬ x2 ¬ x3.

L‟OBDD de la fonction *f* peut se représenter informatiquement par le tableau suivant :

Table 3.4: Représentation informatique de la fonction f

0	0	-	-
1	1	-	-
2		3	4
3		1	5
4		6	7
5		1	1
6		0	1
7		1	1

3.11. Représentation des multi-fonctions

Il existe deux façons pour la représentation des multi-fonctions : soit en réalisant un OBDD pour chaque fonction, soit on partageant des sous-arbres. Cette dernière solution conserve plus de temps et d‟espace [Knuth, 2009]. Le choix d‟utiliser des sous-arbres pour représenter des multi-fonctions peut rendre la vérification de l‟équivalence de deux fonction f et g une tâche facile. La figure suivante représente les fonctions suivantes:

$f1\ (x1, x2) = x1.\ \neg x2,\ f2\ (x1, x2) = x1 \oplus x2,\ f3\ (x1, x2) = \neg x2,$ et $f4\ (x1, x2) = x1 + \neg x2.$

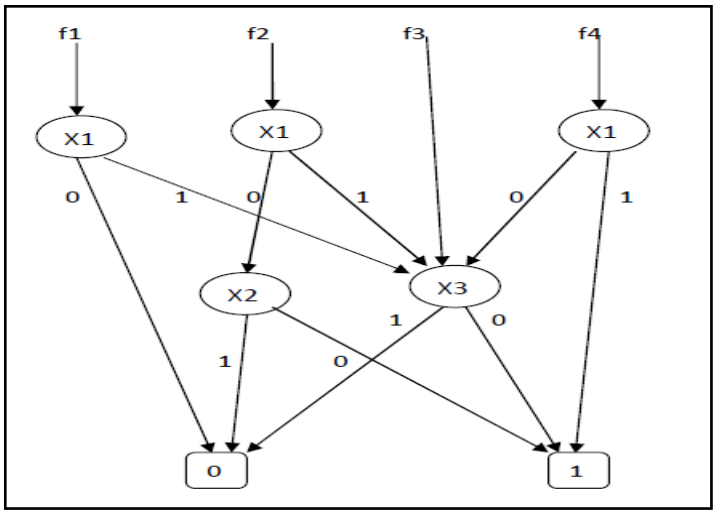


Figure 3.22 – OBDD d'une multi-fonction.

La représentation informatique d‟un tel OBDD peut être :

Table 3.5: Représentation d‟OBDD d'une multi-fonction

	0	-	-	0
	1	-	-	1
2	x1	0	1	
3	x1	1	0	F3
4	x2	0	3	F1
5	x2	2	3	F2
6	x1	3	1	F4

3.12. Variantes des OBDDs

Il existe plusieurs extensions des OBDDs pour des utilisations différentes, nous en présenterons ici quelques-unes parmi les plus populaires. Pour chacune d‟elles, nous expliquerons brièvement la structure et nous donnerons quelques caractéristiques [Paquette, 2005].

3.12.1 Diagrammes de décision sans zéro

Les diagrammes de décision sans zéro (ZBDD, de l‟anglais *Zero-suppressed Binary Decision Diagrams*) [Minato, 1993] sont des diagrammes de décision binaire dont la réduction est différente de celle utilisée pour obtenir un OBDD. Pour réduire un diagramme de décision binaire dans le but d‟obtenir un ZBDD, les nœuds dont l‟arc plein pointe directement sur le nœud terminal 0 sont supprimés, comme le montre la figure suivante. Cette figure est un exemple d‟élimination d‟un nœud dont l‟arc plein pointe sur 0. En a) nous avons le graphe à réduire, en b) le graphe en partie réduit et en c) le graphe totalement réduit. Cette représentation est plus pratique et concise que les OBDDs pour certaines applications, par exemple celles qui utilisent les ensembles de combinaison de bits.

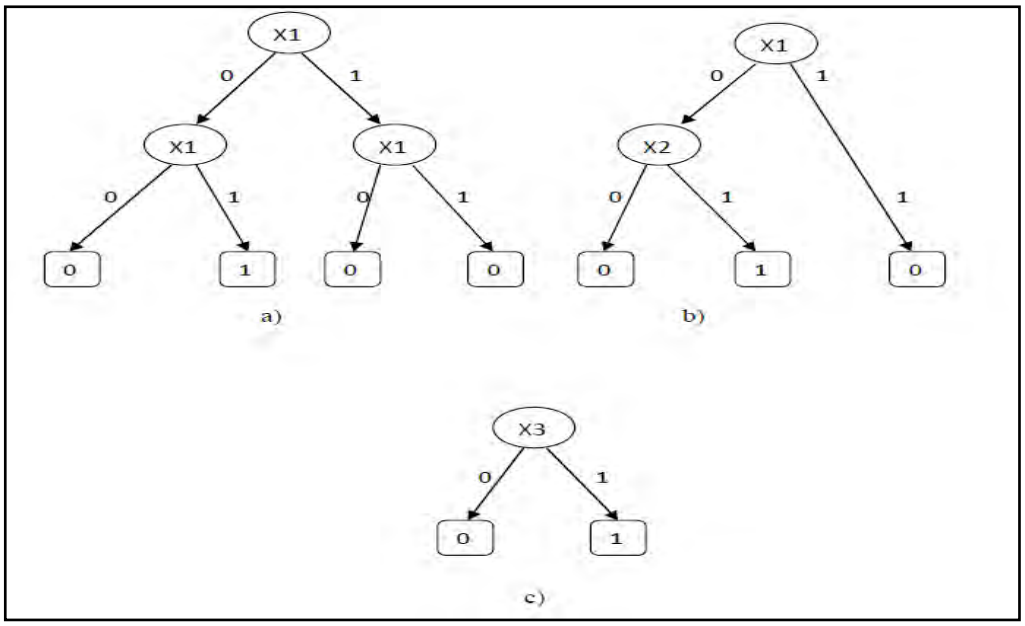


Figure 3.23 – Réduction d'un ZBDD représentant la fonction f = ¬x1 ∧ x2.

3.12.2 Diagrammes de décision binaire ordonnés et partitionnés

Un POBDD (de l‟anglais *Partitioned Ordered Binary Decision Diagrams*) est une structure qui utilise en fait plusieurs OBDDs. Le but est de représenter une formule booléenne avec plusieurs OBDDs qui pourront avoir un ordre des variables différent. Pour ce faire, le domaine de la fonction à représenter est décomposé en sous-ensembles qui ne sont pas nécessairement disjoints [Narayan et al., 1996]. Chaque OBDD utilisé représente une fonction

g_i égale à la fonction f sur le $i^{ème}$ sous-ensemble du domaine et qui est nulle ailleurs. Ainsi, lors de l'évaluation du POBDD, si au moins un des OBDD donne la valeur 1 comme résultat à l'entrée donnée, alors la valeur de la fonction f avec cette même entrée est 1.

Un avantage d'une telle représentation est que chaque OBDD peut avoir un ordre des variables différent. De plus, il se peut qu'il n'existe pas de bon ordre des variables pour une fonction donnée mais qu'il en existe un (le même pour tous les g_i) qui donne de bons résultats pour le POBDDs. Si les sous-ensembles sont fixés, ainsi que l'ordre des variables, alors cette représentation est canonique [Paquette, 2005].

Voyons l'exemple suivant. Soit la fonction f = (¬x1 ∧ ¬x2 ∧ x3) ∨ (x1 ∧ ¬x2 ∧ ¬x3) ∨ (x1 ∧ x2 ∧ x3). Pour illustrer la décomposition du domaine, voyons celle-ci sur la table de vérité de f:

Table 3.6: Décomposition du domaine de f.

Sous-ensembles	X1	X2	X3	F(X1, X2, X3)
A	0	0	0	0
	0	0	1	1
	0	1	0	0
B	0	1	1	0
	1	0	0	1
C	1	0	1	0
	1	1	0	0
	1	1	1	1

3.12.3 Digrammes de différence d'horloge (Clock Difference Diagrams)

Un diagramme de différence d'horloge (CDD) [Larsen et al, 1999] est une nouvelle structure utilisée pour la représentation des contraintes temporelles. Les nœuds de ce diagramme représentent les différences d'horloge y compris une horloge fictive représentant la valeur 0, et les arcs représentent les bornes de ces différences. Les feuilles représentent les valeurs de vérité *true* et *false* (figure 3.24). Mais, seulement on s'intéresse que par les cas qui mènent vers la feuille *true*. Les CDDs sont utilisés dans les model-checking pour vérifier les systèmes temps réel. Des expériences ont montrés que cette représentation diminue considérablement l'espace mémoire et le temp-CPU. À titre d'exemple, l'outil de vérification UPPAAL [Behrmann et al, 2001] utilise un CDD pour la vérification des systèmes temps réel. En effet, UPPAAL a été appliquée avec succès dans un bon nombre d'études de cas académiques et industriels.

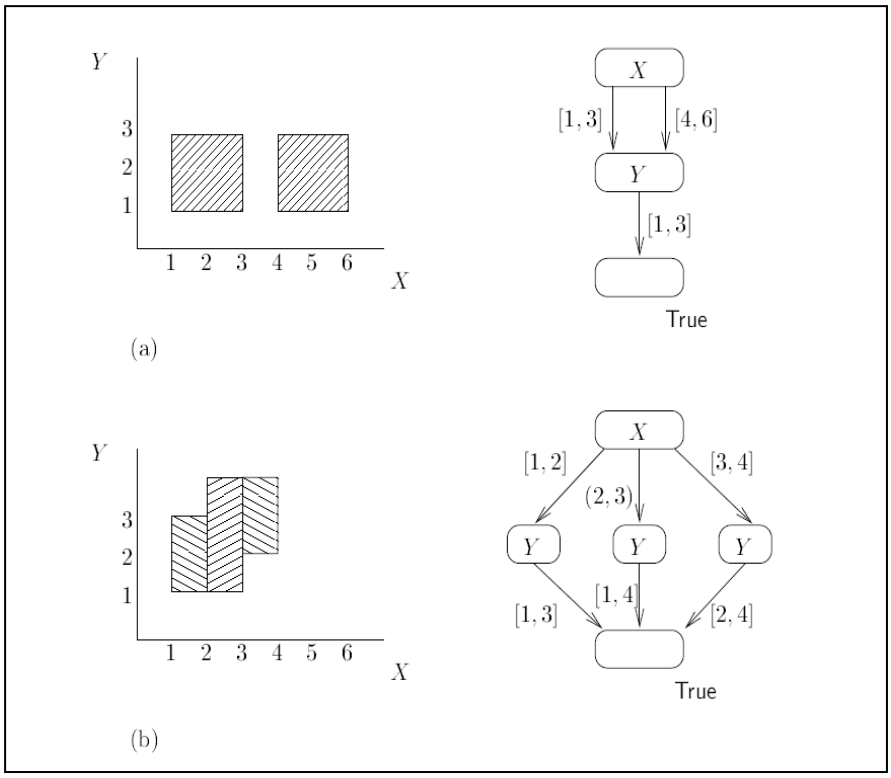


Figure 3.24 – Deux exemples de CDD.

3.12.4 Diagrammes de décision binaire à arcs valués

Les diagrammes de décision binaire à arcs valués (EVBDD, de l'anglais Edge Valued Binary Decision Diagrams) sont des diagrammes de décision binaire qui ont une seule feuille (étiquetée 0) et dont les arcs sont munis d'une valeur (figure 3.25). Ils sont utilisés pour représenter les fonctions à valeurs entières et à variables binaires. Les EVBDD et les OBDD sont trés identique à l'exception de la sémantique d'un nœud d'un EVBDD et celle d'un OBDD.

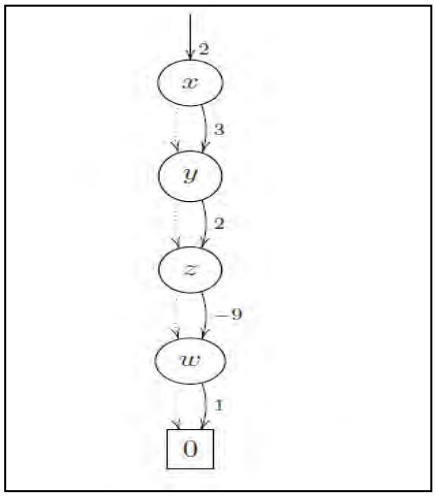


Figure 3.25 – L'EVBDD associé à la fonction g= g(x, y, z,w) = 3x+2y−9z+w+2.

3.13. Conclusion

Dans ce chapitre, nous avons présenté les diagrammes de décision binaire. Ces derniers servent à représenter de manière symbolique des ensembles d'états. Cette méthode de représentation a été utilisée avec succès dans la vérification de systèmes. En effet, le pouvoir d‟expression des BDDs est suffisant pour manipuler une grande classe de systèmes finis. D‟un autre côté, l‟un des problèmes rencontrés en utilisant les BDDs est le choix de l‟ordre de variables utilisées. En outre, un bon ordre de variables réduit considérablement la taille du BDD ce qui facilite donc sa manipulation. De ce fait, trouver le bon ordre de variables constitue une étape très importante dans le processus d‟optimisation des BDDs. Malheureusement, cette tâche n‟est pas facile et ce problème fait partie de la classe de problèmes NP-complet.

Pour cela, nous avons proposé dans le chapitre 5 deux nouvelles approches évolutionnaires hybrides pour résoudre ledit problème.

CHAPITRE 4

Problèmes de satisfiabilité booléenne

"La liberté, c'est la liberté de dire que deux et deux font quatre. Lorsque cela est accordé, le reste suit." - George Orwell, 1984

Certains problèmes combinatoires occupent une place importante chez les chercheurs qui sont soucieux pour trouver des méthodes de résolution efficaces pour ces problèmes. C'est notamment le cas du problème de satisfiabilité en logique propositionnelle (SAT). Le problème SAT consiste à déterminer si une formule propositionnelle mise sous forme normale conjonctive est satisfaisable. Ce problème a été le premier à être démontré comme étant NP–complet. De nombreuses études, théoriques et pratiques, ont porté sur le problème SAT. Elles ont permis de créer des algorithmes, souvent basés sur un algorithme complet et incomplet.

Dans ce chapitre, nous allons présenter un état de l'art de problème de satisfiabilité booléenne SAT. Nous rappelons tout d'abord les éléments de base de la logique propositionnelle ainsi que certains problèmes qui lui sont associés tels que le problème MAX-SAT. Ainsi, nous rappelons les différentes méthodes de résolutions existantes (méthodes exactes et méthodes approchées, hybrides).

Sommaire

4.1. Introduction

L'optimisation combinatoire occupe une place très importante en recherche opérationnelle, en mathématiques discrètes et en informatique. Il existe plusieurs problèmes d‟optimisation combinatoire qui sont difficile à résoudre. Parmi ces problèmes, il y a le problème de la satisfiabilité propositionnelle (SAT). Le problème SAT consiste, pour une formule booléenne donnée, à trouver une interprétation des variables qui satisfasse l'ensemble des clauses de cette formule. Il existe plusieurs variantes de ce problème. L‟une des variantes la plus populaire est le problème de MAXimum SATisfiabilité MAX-SAT qui consiste à déterminer l'interprétation qui rende un maximum de clauses de la formule vraies. Ce problème constitue une généralisation du problème SAT lorsque la formule n'admet pas de solution, en permettant de déterminer une solution approximative.

Le problème SAT est très important, de nombreux problèmes sont facilement représentables par la logique propositionnelle comme dans les cas suivants :

- Software Engineering : vérification de logiciels
- Hardware Engineering : vérification de circuits
- Réseaux : routage, ordonnancement, spécification de protocoles
- IA : diagnostic, planification, raisonnement, apprentissage, etc.
- Web : e-commerce.

La résolution de ce problème est généralement considérée comme un problème de décision. Malheureusement, le problème SAT et un grand nombre de ses variantes appartiennent à la classe des problèmes NP-complet [Cook, 1971]. En effet, la complexité de résolution augmente de façon exponentielle par rapport à la taille du problème.

Etant donné l'importance de ce problème, de nombreuses méthodes de résolution ont été proposées et fournissent des résultats satisfaisants dans la pratique. Elles peuvent se départager en deux grandes classes, les méthodes complètes dites exactes et les méthodes incomplètes dites approchées [Lardeux, 2005]. Les méthodes complètes examinent la totalité de l‟espace de recherche, elles répondent entièrement au problème de satisfaisabilité d‟une instance. Dans le pire des cas, le temps de calcul nécessaire à l‟exécution de telles méthodes augmente exponentiellement avec la taille de l‟instance à résoudre. C‟est le cas notamment des méthodes DP et DPLL [Davis et al., 1962], ou de variantes de méthodes plus génériques comme Branch & Bound [Alsinet et al, 2003]. Cependant, les méthodes incomplètes n‟explorent que certaines zones de l‟espace de recherche et s‟avèrent particulièrement utiles pour des instances de grande taille [Hoos et al, 2000]. Les méthodes incomplètes, bien que très efficaces, ne répondent que partiellement à la question de satisfaisabilité d‟une instance SAT. Lorsqu‟elles ne parviennent pas à prouver la consistance d‟une instance, seules des conjectures concernant la satisfaisabilité peuvent être avancées, ce qui est insuffisant.

4.2. Logique Propositionnelle

La logique propositionnelle est la base de la logique des prédicats et des relations. Elle représente le plus simple et le premier des calculs logiques. Malgré son apparente simplicité, la logique propositionnelle permet de représenter une grande quantité de connaissances. Elle a pour objet l'étude des relations logiques entre propositions (ses éléments de base) en fonction de leur propriétés d'être vrais ou fausses en fournissant des règles d'inférence permettant de produire des raisonnement valides. Une logique est une syntaxe ou une façon de construire l'ensemble des formules. Une sémantique est une description de ce que ces formules signifient, et un système de preuve qui nous permet de calculer la signification des formules en construisant des preuves [Lassaigne et al, 1996]. Pour bien situer les problèmes de satisfiabilité, nous commencerons par donner des définitions qui seront utiles pour mieux comprendre le contexte du problème. La plus part des définitions suivantes sont prises de [Huth et al ,2000] et de [Lardeux, 2005].

4.2.1 Syntaxe

Nous commençons par définir les principaux éléments syntaxiques de la logique propositionnelle. La syntaxe de la logique propositionnelle est fondée sur des variables de proposition ou atomes que nous notons avec des lettres majuscules prises au début de l'alphabet, A, B, C, etc., possiblement avec des indices ou des primes. Ces lettres sont supposées représenter des propriétés de base, qui sont soit vraies soit fausses. Ces variables sont combinées aux moyens de connecteurs logiques pour former des formules ou propositions, que nous notons par des lettres grecques majuscules comme F, Y.

Définition 4.1 (Langage) : Le langage de la logique propositionnelle est construit à partir de l'alphabet constitué de :

– Un ensemble V de variables propositionnelles

– Les parenthèses «(» et «)»

– Les opérateurs, appelés également connecteurs, non «¬», ou «∨», et «∧», implique «⟶», équivaut «⟷».

– les 2 constantes «Vrai» et «Faux», que l'on peut noter également par «True» et «False», «⊤» et «⊥» ou encore «0» et «1».

Définition 4.2 (Variable propositionnelle) : Une variable propositionnelle est une formule atomique pouvant prendre les valeurs de vérité *vrai ou faux*. Elle représente la valeur logique d'une proposition. L'ensemble des variables propositionnelles est noté X .

Définition 4.3 (Connecteur logique) : L'ensemble des connecteurs logiques pour la logique propositionnelle est l'ensemble P= {¬, ou ∨, et ∧, ⟶, ⟷} Le connecteur ¬ est un connecteur unaire appelé négation. Les autres connecteurs sont binaires et sont appelés conjonction (∧), disjonction (∨), implication (→) et équivalence (↔).

Définition 4.4 (Littéral) : Un littéral est une variable propositionnelle $x \in X$ ou sa négation ¬x. L'ensemble des littéraux est noté L.

Définition 4.5 (**Formule propositionnelle)** : Soit un alphabet constitué d'un ensemble de variables propositionnelles et de deux symboles particuliers ⊤ et ⊥. L'ensemble des formules propositionnelles ou propositions Prop est défini récursivement comme suit:

- ⊤ et ⊥ ∈Prop,
- X ⊂ Prop
- Si A∈ Prop, Alors ¬ A est une formule,
- Si A∈ Prop, Alors (A) est une formule
- Si A, B ∈ Prop Alors A∧B, A∨B, A→B, A↔B ∈ Prop.

Définition 4.6 (**Clause)** : Une clause est une disjonction de littéraux aussi manipulée comme l'ensemble de ses littéraux. L'ensemble des clauses d'une formule ф est noté Cф.

Définition 4.7 (Clause positive): Une clause est dite positive si elle ne contient que des littéraux positifs.

Définition 4.8 (Littéral pur): Un littéral pur est un littéral (un élément de Lit( )) qui apparaît dans au moins une clause de la formule et dont la négation est absente de toute clause.

Définition 4.9 (Clause unitaire) : Une clause unitaire est une clause ne contenant qu'un seul littéral.

Définition 4.10 (Clause vide) : Une clause vide ne contient aucun littéral.

Définition 4.11 (Clause de Horn) : Une clause de Horn ne contient qu'un seul littéral positif.

4.2.2 Sémantique

La sémantique associée à la logique propositionnelle se fonde sur le principe suivant : La valeur de vérité d"une variable est soit vrai, soit faux. Une affectation sur l"ensemble des

variables est une assignation de valeurs de vérité qui permet de déterminer la valeur de vérité correspondante d‟une formule.

Définition 4.12 (Variable instanciée) : Une variable instanciée est une variable ayant une valeur de vérité.

Définition 4.13 (Formule satisfiable) : Une formule propositionnelle ϕ est dite satisfiable s'il existe au moins un modèle ($\exists A \in S, A \vDash \phi$).

Définition 4.14 (Formule insatisfiable): Une formule propositionnelle ϕ est dite insatisfiable si pour toute affectation il n'existe pas de modèle ($\forall A \in S, A \nvDash \phi$).

Définition 4.16 (Interprétation) : Une interprétation I en calcul propositionnel est une application associant à toute formule propositionnelle ϕ une valeur I(F) dans {faux, vrai}. L'interprétation I(F) d'une formule F est définie par la valeur de vérité donnée à chacun des atomes de F. I(F) se calcule par l'intermédiaire des règles suivantes : Où 1 désigne le vrai et 0 le faux.

- I(⊤)=vrai ;
- I(⊥)=faux
- I(¬F) = vrai ssi I(F) = faux ;
- I(F ∧ G) = vrai ssi I(F) = I(G) = vrai ;
- I(F ∨ G) = faux ssi I(F) = I(G) = faux ;
- I(F → G) = faux ssi I(F) = vrai et I(G) = faux ;
- I(F ↔ G) = vrai ssi I(F) = I(G).

Définition 4.17 (Interprétation partielle, incomplète et complète)

Soit ϕ une formule propositionnelle dont le nombre de variables est égale à n. On dit que :

– I est une **interprétation partielle** de ϕ si et seulement si I est une interprétation et $|I| \leq n$;

– I est une **interprétation incomplète** de ϕ si et seulement si I est une interprétation et $|I| < n$;

– I est une **interprétation complète** de ϕ si et seulement si I est une interprétation et $|I| = n$.

Définition 4.18 (Tautologie) : Une formule propositionnelle ϕ est une tautologie si toutes affectations sont vraies ($\forall A \in S, A \vDash \phi$).

Définition 4.19 (Modèle) : On appelle **modèle** d'une une formule propositionnelle ϕ, une interprétation I qui satisfait ϕ.

Définition 4.20 (Contre-Modèle) : On appelle **Contre-Modèle** d'une une formule propositionnelle ϕ, une interprétation I qui falsifie ϕ.

Définition 4.21 (Consistance, Inconsistance): Une formule est dite **consistante** ou satisfiable ou satisfaisable si et seulement si elle admet au moins un modèle. Une formule est dite **inconsistante** ou **insatisfaisable** si et seulement si elle n'admet pas de modèle.

Définition 4.22 (Conséquence sémantique ou conséquence logique) : Une formule ϕ **implique sémantiquement** une formule Ψ notée, $\phi \models \Psi$ si et seulement si tout modèle de ϕ est un modèle de Ψ. On dit alors que ϕ a pour **conséquence logique** Ψ ou encore que Ψ est une **conséquence logique de** ϕ.

Définition 4.23 (équivalence sémantique) : On dit que deux formules ϕ et Ψ sont **sémantiquement équivalentes**, noté $\phi \equiv \Psi$ si et seulement si $\phi \models \Psi$ et $\Psi \models \phi$, c'est à dire si ces deux formules admettent exactement les mêmes modèles.

Définition 4.24 (Forme Normale Conjonctive (CNF)) : Une disjonction est une proposition de la forme A∨B et une conjonction est une proposition de la forme A∧B. Une proposition est en forme normale conjonctive (FNC) si elle est composée de conjonctions de disjonctions. Une proposition est en forme normale disjonctive (FND) si elle est composée de disjonctions de conjonctions**.** Seuls les connecteurs ¬, ∨ et ∧ ainsi que les variables propositionnelles composent une formule CNF ou FND.

Définition 4.25 (Forme Normale Conjonctive) : Une formule ϕ est en forme normale conjonctive si et seulement si elle s'écrit :

$\phi = C1 \wedge C2 \wedge \ldots \wedge Cm$ Où chaque C_i est une clause.

Propriété : Une formule propositionnelle en CNF est vraie si et seulement si toutes ses clauses sont vraies. Une formule en CNF peut donc être vue comme un ensemble de clauses devant toutes être vraies pour que la formule soit vraie.

4.2.3 Notions propres au contexte de résolution

On va présenter quelques notions qui sont utilisées pour la résolution des problèmes de satisfiabilité.

Définition 4.26 (Résolvante) : Deux clauses c1 et c2 se résolvent si et seulement si il existe un littéral l tel que $l \in c1$ et $\sim l \in c2$. La clause $((c1) \setminus \{l\} \cup (c2) \setminus \{\sim l\})$ est

appelée résolvante de c1 et c2 en l. Il est évident que cette résolvante est logiquement impliquée par les clauses c1 et c2.

Définition 4.27 (Backbone) : Le backbone d'une formule ϕ est l'ensemble des variables une valeur de vérité constante dans toutes les affectations satisfaisantes ϕ.

$$\text{Backbone: Prop} \rightarrow 2^K$$

$$\phi \rightarrow \{v \mid \forall A, A' \in S \text{ tq } A \models \phi \wedge A' \models \phi \Rightarrow A(v) = A'(v)\}$$

Définition 4.28 (propagation unitaire) On note ϕ* la formule obtenue à partir ϕ par application de la propagation unitaire. ϕ* est définie récursivement comme suit :

- $\phi^* = \phi$ si ϕ ne contient aucune clause unitaire.
- $\phi = \bot$ si ϕ contient deux clauses unitaires {x} et {¬x},
- autrement $\phi^* = (\phi \mid x)^*$ tel que x est le littéral qui apparaît dans une clause unitaire de ϕ.

Définition 4.29 (Flip) : Le flip d'une variable, aussi appelé complémentation, est le fait d'inverser sa valeur de vérité de F à V ou de V à F. Seules les variables ayant déjà une valeur de vérité peuvent entre flippées. La fonction flip peut être définie comme suit :

$$\text{Flip: } \{F, V\} \rightarrow \{F, V\}$$

$$a \rightarrow \neg a$$

4.3. Problèmes de satisfiabilité booléenne SAT et MAX SAT

Le problème de satisfaction d'une formule de la logique propositionnelle mise sous forme normale conjonctive s'énonce comme suit :

Soit:

- X un ensemble de n variables propositionnelle tels que :
- $X=\{x_1, x_2 \ldots x_n\}$ avec $x_i \in \{1,0\}$, i=1…, n.
- C un ensemble de m clauses : $\phi = \{C1, C2, \ldots, Cm\}$ avec Cj une disjonction de littéraux x_i ou leurs négation $\neg x_i$ $(Cj = \vee l_{jk})$ Où j=1..m, k=1..n.
- ϕ est une formule sous forme normale conjonctive : $\phi = C1 \wedge C2 \wedge \ldots \wedge Cm$.

Le problème de satisfiabilité booléenne ou tout court SAT consiste à vérifier si la formule ϕ énoncée comme ci-dessus est satisfaite ou non [Cook, 1971. La résolution de ce problème consiste à trouver une assignation ou une interprétation I de valeurs pour les littéraux x_i, qui satisfait toutes les clauses Cj en même temps : ∀ Cj (j=1..m) : telle que Cj ≡vrai pour I. Cela signifie qu'au moins un littéral x_i i=1..n, est à vrai pour chaque clause Cj (j=1..m). Dans le cas

contraire prouver son inconsistance c'est-à-dire que pour au moins une clause, cette assignation de valeurs donne faux : ∃ Cj (j=1..m) : telle que Cj ≡Faux pour I. C'est-à-dire qu‟il existe au moins une clause Cj pour laquelle tous les littéraux x_i sont à faux. Cette définition démontre clairement le caractère NP-Complet du problème SAT.

Une instance du problème SAT est le k–SAT, où k est un entier naturel. K–SAT est un problème SAT où toutes les clauses contiennent exactement K littéraux. 3–SAT est la forme la plus simple de k–SAT qui reste NP–complet. Hormis, 2–SAT, il existe diverses classes de SAT qui sont polynomiales. Citons par exemple, le problème Horn-SAT est semblable au problème SAT, mais les clauses qui constituent la formule booléenne sont des clauses de Horn. Ce type de problème appartient à la classe P. Le problème (s, r–SAT) où les clauses possèdent s littéraux et où chaque variable apparaît au plus r fois.

Dans certain cas, on veut savoir combien de clauses sont vraies. Ceci est une autre instance du problème père SAT appelé MAX-SAT pour MAXimum SATisfiabilité. Ce problème suscite un intérêt croissant ces dernières années vu ces applications dans plusieurs domaines. Pour une formule propositionnelle en CNF, le problème consiste à déterminer le nombre maximal de clauses de cette formule qui peuvent être satisfaites (i.e. minimisant le nombre de clauses fausses). Une instance de ce problème est le problème *MAX k-SAT***.** Ce dernier concerne les instances ayant des clauses d‟au maximum k littéraux. Comme un cas particulier du problème K-SAT on a le problème 3-SAT *(K=3)* c‟est-à-dire la taille des clauses est inférieure ou égale à 3. Le problème MAX k-SAT a été démontré NP-complet pour tout $k>2$.

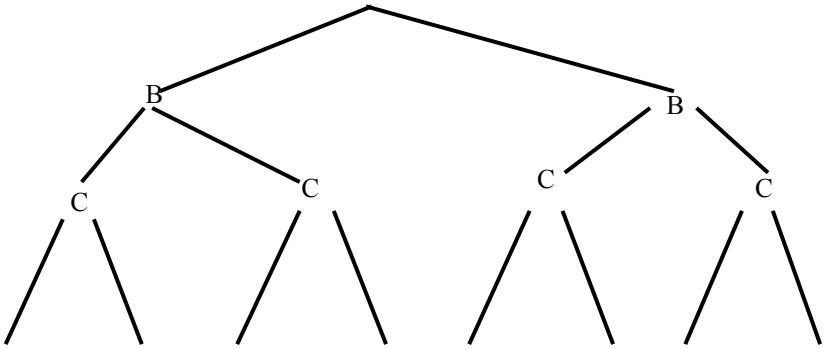
Il existe plusieurs variantes du problème MAX-SAT. Par exemple, le problème MAX-W-SAT est similaire au problème MAX-SAT sauf que les clauses sont pondérées où chaque clause est associée à un poids w_i. La résolution du problème consiste à maximiser la somme des poids des clauses satisfaites ou de minimiser la somme des poids des clauses insatisfaites.

4.4. Méthodes de Résolutions pour les problèmes SAT et MAX-SAT

Bien que le problème SAT appartienne à la classe des problèmes NP–complets, de nombreuses méthodes de résolution ont été proposées et fournissent des résultats satisfaisants dans la pratique. Au cours des vingt dernières années, l‟avènement de nouvelles méthodes a permis de réaliser des progrès significatifs en terme de performance, que ce soit au niveau des temps de calcul ou de la taille des instances traitées. On peut départager en deux grandes classes, les méthodes incomplètes et les méthodes complètes.

Plusieurs approches de résolution exacte pour le problème MAX-SAT ont été élaborées. La plupart de ces méthodes sont basées sur les techniques énumératives s'appuyant, généralement, sur une méthode par séparation et évaluation (Branch & Bound) [Alsinet et al., 2003]. Il s'agit d'énumérer d'une manière implicite toutes les solutions et d'en choisir la meilleure parmi toutes (figure 5.1). Elles peuvent donc théoriquement répondre au problème

SAT et la plupart de ces instances. Actuellement, les méthodes exactes les plus efficaces s'appuient sur la procédure de Davis, Putnam et Loveland [Davis et al, 1962] [Bruni, 2004]. La seule différence entre les méthodes exactes réside essentiellement dans leurs heuristiques de branchement. Des techniques de détection de symétries, de détection de backbones ou d'élimination d'équivalences sont également utilisées pour renforcer ces méthodes. Malgré que les méthodes exactes sont plus complètes en théorie que les méthodes approchées, la complexité de leur mise en œuvre, les rend moins appropriées à ce jour, de résoudre des instances difficiles de grandes tailles.



A	V	V	V	V	F	F	F	F
B	V	V	F	F	V	V	F	F
C	V	F	V	F	V	F	V	F

Figure. 4.1– Exemple d'arbre binaire (complet) représentant une formule propositionnelle avec 3 variables. Le tableau donne les 8 (2^3) affectations possibles (lecture en colonne).

Afin de remédier à cette complexité, des méthodes dites approchées sont généralement appliquées. En effet, les méthodes approchées n'explorent pas d'une manière exhaustive tout l'espace de recherche [Bailleux, 1996], donc elles donnent des solutions approchées. Les méthodes approchées sont essentiellement fondées sur les métaheuristiques. Elles ont pour objectif non pas de trouver la ou les meilleure(s) solution(s) mais de rechercher rapidement une bonne solution vis à vis du coût et des contraintes du problème. Ces méthodes sont particulièrement appropriées pour résoudre des instances MAX-SAT de grandes tailles. On distingue dans la littérature, plusieurs algorithmes approchés basés sur la recherche locale ou les algorithmes évolutionnaires. Parmi les algorithmes populaires de cette catégorie d'algorithme, nous pouvons citer le très simple algorithme GSAT [Selman et al., 1992] et sa variante améliorée Walksat [Selman et al., 1994] basées sur la recherche locale. Les méthodes incomplètes, bien que très efficaces et elles constituent une approche complémentaire indispensable et une alternative intéressante aux méthodes exacte, ne répondent que partiellement à la question de satisfaisabilité d'une instance SAT.

4.4.1 Méthodes exactes pour les problèmes SAT et MAX-SAT

Il existe de nombreux algorithmes exacts dans la littérature pour la résolution des problèmes SAT et MAX-SAT qui se basent sur différente approches. Mais la plupart des méthodes complètes sont des améliorations de la procédure de la méthode Davis Putnam (DP) et la méthode Davis Logemann Loveland (DLL).

4.5.1.1 Algorithme de Davis Putnam

La méthode de Davis et Putnam (DP) [Davis el al, 1960] est l'une des premières méthodes de résolution dédiées au problème SAT. Son principe de base est l'application de la résolution comme principe de l'élimination des variables [Lardeux, 2005]. Le schéma général de cet algorithme est le suivant :

```
Algorithme 4.1 : un algorithme DP
Données : une formule ϕ en CNF
Résultat : satisfiabilité de ϕ
début
    pour chaque variable x de Xϕ faire
        Résolvant = ∅;
        pour chaque couple de clauses (c1, c2) de Cϕ faire
            si (x ∈ c1 et ¬x ∈ c2) alors
                c=c1 ∪ c2 - {x, ¬x};
                si c n'est pas une tautologie alors
                    Résolvant = Résolvant ∪ c;
                fin
            fin
        fin
        Clauses Avec x = {γ ∈ Cϕ| x ∈ γ} ;
        ϕ = ϕ − Clauses Avec x;
        ϕ = ϕ ∪ Résolvant;
    fin
    si ϕ contient une clause vide alors
        retourner "insatisfiable";
    sinon
        retourner "satisfiable";
    fin
fin
```

Dans chaque itération de DP, un certain nombre de variables est supprimé de la formule. Ainsi, si x est la variable à considérer, on insère dans la formule toutes les résolvantes possibles obtenues à partir des clauses contenant x et ¬x. Dans l'étape suivante, les clauses dont les clauses résolvantes ajoutées sont des tautologies sont supprimées. La procédure s'arrête soit par la génération d'une clause résolvante vide, dans ce cas la formule et non

satisfiable, ou bien il n'y a aucune clause résolvante à générer, dans ce cas le problème est satisfiable et chacune des clauses restantes représente une solution possible.

4.4.1.2 Algorithme de Davis Logemann Loveland (DPLL)

La procédure de Davis, Putnam, Logemann et Loveland (communément appelée DPLL) [DAVIS et al. 1962] est une amélioration de l'algorithme DP. Cette méthode remplace le mécanisme de résolution par celui de la séparation du problème en deux sous problèmes. L'idée consiste à construire un arbre binaire dans lequel chaque nœud représente un appel récursif à la procédure DPLL. Toutes les feuilles représentent l'arrêt du processus de recherche sur une clause vide sauf une si la formule est satisfiable. Cette recherche utilise le backtrack afin de remonter dans l'arbre de recherche et ainsi de tester de nouvelles branches. La procédure est détaillée dans l'algorithme 4.2.

Algorithme 4.2: un algorithme DPLL [Lardeux, 2005]:
Données : une formule ϕ en CNF
Résultat : satisfiabilité de ϕ

```
début
  si (ϕ= ∅) alors
    Retourner "satisfiable";
  fin
  ϕ = Propagation Unitaire(ϕ);
  si (ϕ contient une clause vide) alors
   Retourner "non satisfiable";
  Fin
  Sélection d'une variable x dans Cϕ grâce à heuristique H;
 si (DLL (ϕ ∪ {x}) retourne "satisfiable") alors
  Retourner "satisfiable";
 sinon
 Retourner le résultat de DLL (ϕ ∪ ¬{x}) ;
 fin
fin
```

Etant donné une formule CNF ϕ et une variable l de ϕ, la procédure DPLL est basée sur l'idée que ϕ est consistante si et seulement si $\phi \wedge l$ ou $\phi \wedge \neg l$ est satisfaisable. Cette procédure consiste donc à choisir une variable et à tester récursivement les deux sous-formules en interprétant successivement cette variable à *VRAI* puis à *FAUX* si nécessaire.

Cette procédure utilise également le mécanisme de la propagation unitaire, aussi appelée propagation de contraintes booléennes (algorithme 4.3). Son principe est de supprimer dans la

formule CNF toutes les clauses où apparaît un littéral unitaire et de supprimer toutes les occurrences du littéral opposé à ce littéral dans les clauses où il apparait. Ce processus est répété tant qu"il reste un littéral unitaire où bien jusqu"à ce que la production de la clause vide. Pour le problème SAT, l"unique littéral de cette clause doit être rendu vrai (sinon une clause fausse apparaît).

Algorithme 4.3 : Propagation Unitaire

Données : une formule ϕ en CNF

Résultat : ϕ simplifiée

```
début
    tant que il n'y a pas de clause vide et qu'une clause
    unitaire cl existe dans φ faire
      Affecter la valeur de vérité valeur à la variable v de cl
     afin de satisfaire cl;
      φ = Simplifier(φ, v, valeur);
    fin tant que
    Retourner φ;
fin
```

Algorithme 4.4 : Simplifier

Données : une formule ϕ en CNF, une variable v et sa valeur de verite valeur

Résultat : ϕ simplifiée

```
Début
  pour chaque clause c ∈ Cφ faire
     si (v ∈ c) = valeur alors
      φ = φ  − c;
     sinon
     c = c − v;
     fin si
  fin pour
  Retourner φ;
fin
```

4.4.1.3 Méthode Branch-and-bound pour les problèmes SAT

La méthode Branch-and-Bound (B&B) pour MAX-SAT [Alsinet et al., 2003] peut être vu comme une extension de l"algorithme précédent (DPLL). Pour une formule CNF donnée, B&B explore l"espace de recherche de toutes les affectations possibles dans une recherche en

profondeur d'abord. Le schéma de l'algorithme Branch&Bound pour le problème MAX 3-SAT est décrit par l'algorithme suivant :

```
Algorithme 4.5 : un algorithme  B&B
Données: une formule ϕ en CNF, une borne supérieure ub
Résultat: le nombre minimum de clauses fausses.
début
  si (ϕ = Ø ou ϕ  ne contient que des clauses vides) Alors
     Retourner le nombre de clauses vides de ϕ ;
  fin
  si (le nombre de clauses vides de ub ≤ ϕ)  alors
      Retourner  ∞ ;
  fin
  v= variable sélectionnée par une heuristique;
  ub= min(ub,  B&B ( Simplifier(ϕ,¬x),ub));
  Retourner min(ub,  B&B ( Simplifier(ϕ,x), ub));
Fin
```

L'algorithme B&B commence par choisir une variable x non valuée grâce à une heuristique de branchement donnée. Cette variable est ensuite instanciée et la formule est simplifiée en fonction de cette valuation : quand x est mise à vrai (resp. faux), les clauses contenant le littéral x (resp. ¬x) sont supprimées. En outre, les occurrences du littéral ¬x (resp. x) sont effacées des clauses les contenant. Ce processus est connu sous le nom de règle du littéral unique. Une clause dont tous les littéraux ont été supprimés (clause vide) correspond à une clause fausse. Si le nombre de clauses fausses obtenues avec cette valuation est supérieur à la borne supérieure (ub) représentant le nombre de clauses fausses obtenues par la meilleure affectation trouvée depuis le début de la recherche, alors aucune affectation construite à partir de ce nœud ne peut fournir un nombre de clauses fausses plus petit que ub. Selon le principe de la séparation de B&B, la branche correspondante est donc coupée. À cet instant, soit la variable courante est flippée (vrai à faux ou faux à vrai), soit, si l'algorithme a testé les deux valeurs de vérité, il effectue un backtrack. La valeur de vérité de la variable courante est effacée et la variable précédente est inspectée [Lardeux et al., 2006].

Quand une affectation complète dont le nombre de clauses vides correspondant est inférieur à ub est trouvée, alors ub est mise à jour avec cette nouvelle valeur. Le processus ne s'arrête que quand tous les backtracks possibles ont été effectués. Le nombre minimum de clauses fausses pour la formule initialement donnée est la valeur courante de ub. Au début de la recherche, ub doit être initialisée en utilisant deux possibilités. La première consiste à initialiser ub avec une valeur égale au nombre de clauses de la formule mais une telle initialisation mène à une large exploration de l'arbre de recherche. Pour réduire cette exploration, la seconde technique consiste à utiliser un algorithme approché comme

l'algorithme glouton qui retourne une affectation engendrant peu de clauses fausses ce qui réduit donc l'espace de recherche.

4.4.2 Méthodes approchées pour les problèmes SAT et MAX-SAT

Le fait que les algorithmes complets requièrent encore énormément de temps de calcul, même à des instances de problèmes relativement petits, les méthodes incomplètes ont été développés pour tenter de résoudre les problèmes SAT ou MAX-SAT à taille importante dans des délais raisonnables. Les méthodes incomplètes ne garantissent pas de trouver la solution optimale, car elles ne recherchent pas tout l'espace des solutions possibles. Elles ont, toutefois, la potentialité de converger plus rapidement aux optima locaux ou globaux. Une solution globale est une affectation qui soit satisfait totalement un problème SAT ou maximise le nombre de clauses satisfaites dans un problème MAX-SAT. Cependant, les optima locaux sont des solutions que la méthode incomplète n'a pas pu aller au-delà pour trouver une meilleure solution.

Les méthodes incomplètes sont compilées à partir des algorithmes de recherche stochastiques tels que GSAT, WalkSAT, Basic Hill-Climbing, recuit simulé, algorithmes génétiques, et la liste s'allonge encore et encore. Ils ont tous un élément aléatoire dans leur stratégie de recherche. L'idée de base de ces algorithmes est de trouver le coût des assignements aléatoires, puis de régler progressivement ces assignements jusqu'à ce qu'un optimum global soit atteint dans le cas de SAT, ou le temps s'écoule sans trouver une solution à laquelle l'algorithme s'arrête prématurément. Avec le problème de MAX-SAT, on ne sait pas si une solution est atteinte si la formule n'est pas entièrement satisfaisable.

Dans la section suivante, on va parler avec un peu de détails sur des algorithmes incomplets les plus célèbres à savoir GSAT et WalkSAT. Tout comme dans le cas de DPLL, une grande partie des algorithmes stochastiques sont construits autour de ces deux-là.

4.4.2.1 Méthodes approchées basées sur la recherche locale

Les méthodes de recherche locale, pour lesquelles l'espace de recherche est exploré de façon non systématique s'avérèrent très efficaces sur un large ensemble d'instances satisfaisables laborieuses, aussi bien générées aléatoirement qu'issues d'applications réelles codées en SAT. Leur grand atout par rapport aux méthodes exactes est l'exploration incomplète de l'espace de recherche ce qui aide à réduire le temps de résolution [Aarts et al, 1997] [Michalewicz et al, 2000]. Ces méthodes sont basées sur un processus itératif, du fait qu'elles raffinent itérativement une solution initiale (interprétation), et tentent de la raffiner en appliquant des opérations de flips sur certaines variables jusqu'à ce que l'interprétation devienne un modèle. La méthode de choix de la variable à flipper caractérise la méthode de recherche locale. L'objectif de ces méthodes est de s'échapper des minima locaux. Il s'agit de situations où quel que soit la variable que l'on flippe, le nombre de clauses insatisfaites ne diminue pas. Il est nécessaire de trouver des heuristiques dites d'échappement qui permettent

de sortir de ce genre de piège. Les méthodes de recherche locales utilisent principalement les deux fonctions suivantes:

1) Fonction d'évaluation: Pour estimer la qualité d'une affectation A de l'espace de recherche S, une fonction d'évaluation est utilisée. Elle consiste à retourner le nombre de clauses vraies générées par une affectation A pour la formule ϕ.

2) Fonction de voisinage: la fonction de voisinage consiste à chercher les voisins d'une solution. Un voisin d'une affectation A est une affectation A' ne différant de A que sur une seule variable. Chaque voisin A′ correspond au flip d'une variable de A. Pour une formule ayant n variables, chaque affectation a donc n voisins possibles. Afin d'éviter de se bloquer dans un voisinage restreint, des techniques d'échappement sont à utiliser comme:

- Flipper les variables qui ne changent pas la valeur de la fonction d'évaluation.
- Eviter de toujours flipper les mêmes variables afin d'éviter les boucles.

Il est à noter qu'il existe plusieurs heuristiques pour choisir la ou les interprétations suivantes parmi le voisinage [Le Berre, 2000] :

- ***Choix du meilleur voisin :*** dans le cadre général, le choix des interprétations est restreint à l'ensemble des interprétations du voisinage dont l'évaluation est meilleure que l'évaluation de l'interprétation courante. Dans le cadre d'un algorithme glouton (Greedy Local Search), seul le meilleur de ces voisins est choisi (pas de retour arrière).
- ***Choix aléatoire :*** cette technique consiste à flipper aléatoirement une variable, cela revient à choisir une interprétation du voisinage au hasard. Une amélioration consiste à flipper seulement les variables des clauses fausses.

La figure 4.2 nous montre le comportement possible d'un algorithme de type recherche locale. La courbe représente l'évaluation de chaque interprétation par la fonction f (le nombre de clauses falsifiées par l'interprétation).

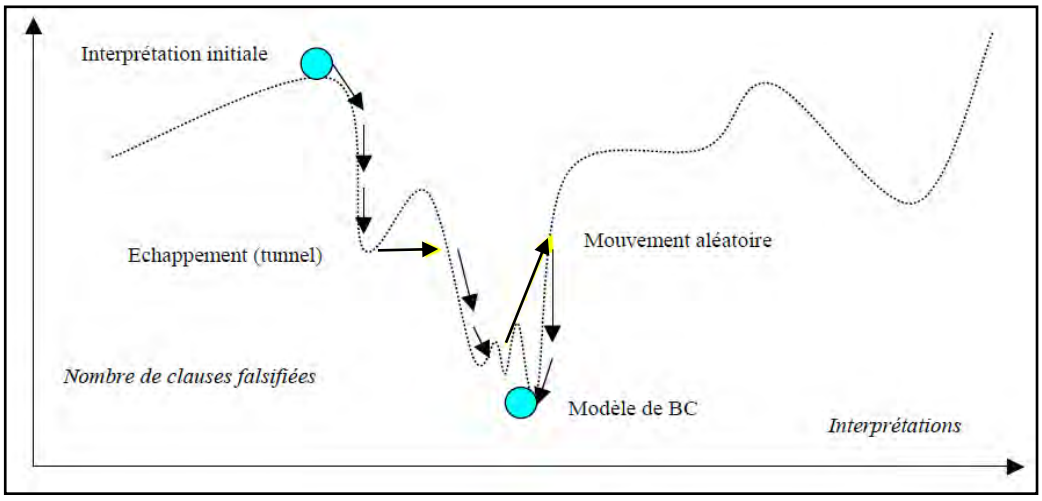


Figure 4.2 – Exemple de parcours d'un algorithme de recherche locale.

On peut distinguer les différentes phases de la recherche locale dans ce schéma :

- *La descente :* on choisit le meilleur voisin.
- *L'échappement de type tunnel :* quand on se trouve dans un minimum local, on «saute» sur une interprétation qui n'est pas dans le voisinage et qui ne change pas la valeur de la fonction d'évaluation.
- *Les sauts aléatoires :* on effectue un choix aléatoire. Cela permet d'échapper à certains minima locaux.

D'une manière plus formelle, la structure détaillée d'un simple algorithme de recherche locale pour le problème MAX SAT est la suivante :

Algorithme 4.6 : procédure de recherche locale pour MAX-SAT
Donnée: formule *MAX-SAT* ϕ *en CNF,* maxSteps
Résultat: *meilleure solution trouvée*

```
Début
  A:= initAssign(ϕ)
  A_best := A;
  pour j:=0  jusqu'à maxSteps faire
     si eval(A_best) < eval(A) alors b A_best := A;
     si A satisfies ϕ alros retourner A;
     sinon
      x:= chooseVariable(ϕ ,A);
      A := A with truth value of x flipped;
     fin si
  fin pour
  retourner A_best;
end
```

la méthode populaire dans cette classe de méthodes est sans aucun doute GSAT proposée par Selman [Selman et al., 1992]. GSAT se base sur l'idée de minimiser la valeur de la fonction objectif qui est le nombre de clauses insatisfaites, en balayant le voisinage de la solution courante par le basculement de la valeur d'une seule variable de la solution. Plusieurs variantes de GSAT ont été développées dans le but d'améliorer l'algorithme d'origine tel que WSAT, GWSAT et HSAT, etc. Nous allons présenter ci-après la méthode GSAT et sa variante la méthode Walksat [Selman et al., 1994].

- ***GSAT***

GSAT est l'une des premières méthodes de recherche locale à avoir été développée pour le problème SAT. Elle a été présentée en 1992 par Selman [Selman et al., 1992] . La méthode

GSAT permet d'atteindre un état satisfaisant l'ensemble ou un sous-ensemble des contraintes d'un problème en appliquant des réparations successives à une instanciation initiale des variables. Cette méthode a démontré son efficacité par le traitement des problèmes de grandes tailles qui restent généralement inaccessibles aux méthodes classiques de type exploration arborescente. De plus, confronté à d'autres procédures de recherche locale, GSAT a souvent obtenu les meilleurs résultats.

L'algorithme commence par une valuation aléatoire de chaque variable de la formule étudiée. Ensuite, GSAT sélectionne, grâce à une heuristique de branchement H, une variable x et la flippe. Cette heuristique choisit la variable induisant la plus grande valeur pour la fonction d'amélioration. Cette opération est réitérée jusqu'à ce que la formule soit satisfaite ou jusqu'à ce que MAX-FLIPS soient exécutés. Ce processus est répété (MAX-TRIES fois au maximum) jusqu'à obtenir une affectation correcte. L'intensification correspond ici aux flips et la diversification s'opère à chaque génération aléatoire de valeurs des variables. Il est à noter que l'amélioration du nombre de clauses vraies engendrée par le flip de la variable sélectionnée x peut être négative lorsqu'aucune affectation ne peut améliorer l'affectation courante (l'algorithme 4.7). Actuellement GSAT est moins compétitif dans le monde des solveurs Max Sat.

Algorithme 4.7 : un algorithme GSAT

Données: une formule ϕ en CNF, MAX- FLIPS, MAX -TRIES

Résultat: satisfiabilité de ϕ si elle est trouvée

```
début
  Tantque MAX-TRIES ≠ 0 faire
    A =une affectation aléatoire des variables de ϕ
    tantque MAX-FLIPS ≠ 0 faire
        si (A⊨ ϕ ) alors
          Retourner "satisfiable";
        fin
        x= H(ϕ ,A);
        A = A[x ← flip(A|x)];
        MAX FLIPS = MAX FLIPS – 1;
    Fintanque
    MAX TRIES = MAX TRIES – 1;
  Fintanque
  Retourner "Pas d'affectation trouvée";
fin
```

– ***Walksat***

Walksat [Selman et al., 1994] est longtemps resté l'algorithme de référence pour les méthodes approchées. Walksat est une évolution de GSAT dans laquelle l'heuristique utilisée pour sélectionner la variable à flipper est améliorée. La principale différence se situe au niveau de la taille du voisinage évalué : Walksat réduit énormément le nombre de voisins car il sélectionne une clause fausse, et seule une des variables de cette clause pourra être flippée. Une des améliorations les plus importantes pour cette méthode consiste à adopter la stratégie de «marche aléatoire » pendant la résolution du problème. Cette stratégie permet d'accélérer la convergence tout en permettant de s'échapper de certains minima locaux. L'idée de cette stratégie, consiste à choisir une variable à flipper parmi les variables impliquées dans une seule clause falsifiée choisie aléatoirement. Cela permet donc de réduire considérablement la taille du voisinage par rapport à la méthode GSAT tout en permettant d'augmenter significativement le nombre de flips possibles dans un temps fixé. D'autre part, le mécanisme probabiliste dans cette stratégie permet aussi, de s'affranchir de l'évaluation d'un quelconque voisinage en choisissant aléatoirement une variable impliquée dans une clause falsifiée comme variable à flipper, ce dernier procédé augmente également les capacités de diversification de la méthode. L'algorithme détaillé de WalkSat est le suivant :

Algorithme 4.8 : un algorithme Walksat
Données: une formule Φ en CNF, une affectation A
Résultat: une variable

```
Début
   Pour i = 1 à MAXTRIES faire
     Pour i allant de 1 à max essais faire
       Générer une configuration initiale I ;
       Pour j allant de 1 à maxFLIPS faire
        Si I satisfait Φ alors
        retourner (I)
        Fin si
        Choisir aléatoirement une clause c falsifiée par I
        Choisir aléatoirement une variable v de c
        Inverser la valeur de v dans I
       Fin pour
     Fin pour
   Fin pour
 retourner (I est modèle)
Fin
```

De nouvelles heuristiques pour choisir une variable de la clause sont régulièrement proposées dont les principales comme : Random, Best, Novelty, Rnovelty et Tabu. L'heuristique proposée par défaut est Best, comme son nom l'indique, elle retourne, avec une probabilité p (appelée bruit), la variable x dont le flip génère le moins de clauses fausses parmi toutes les variables de la clause. Avec une probabilité (1 – p), elle retourne aléatoirement une des variables de la clause. Quand le nombre de clauses devenant fausses par le flip de la variable x est nul alors x est toujours retournée. D'autre part, l'heuristique Novelty [Selman et al., 1994], qui fournit de meilleurs résultats, est basée sur le même principe que best mais prend aussi en compte un critère d'ancienneté. Une variable qui a été flippée très récemment aura moins de chance d'être sélectionnée par rapport aux autres variables.

4.4.2.2 Méthodes approchées basées sur des algorithmes évolutionnaires

Les algorithmes évolutionnaires ont été appliqués aux problèmes SAT. Néanmoins, l'utilisation d'un pur algorithme évolutionnaire s'est avérée infructueuse dans un problème aussi compliqué que le MAX SAT. En effet, Rana et Whitley [Rana et al, 1998] ont démontré qu'un algorithme génétique classique est inadapté pour le problème MAX 3-SAT, car il nécessite plus d'intensification de recherche que d'exploration et de diversification de l'espace de recherche. Pour cela des hybridations entre les algorithmes évolutionnaires est les méthodes de recherche locales sont nécessaires afin de trouver de bons résultats. Ce type d'hybridation est appelé Algorithme Mémétique [Moscato et al, 2005] [Bontoux et al., 2008].

L'algorithme mémétique le plus connu qui traite le problème MAX 3-SAT est *FlipGA* de [Marchiori et al, 1999]. FlipGA travaille sur une population de 10 individus. À chaque génération, la population est renouvelée en gardant les deux meilleurs individus qui sont conservés à chaque itération. Les parents sont sélectionnés aléatoirement dans la population puis sont croisés en utilisant le croisement uniforme. L'opérateur de mutation est appliqué sur chaque fils avec une probabilité de 0.9 et il flippe la valeur de chacune de ses variables avec une probabilité de 0.5. La nouveauté de cet algorithme réside dans sa méthode de recherche locale qui utilise l'heuristique Flip (algorithme 4.9).

Récemment, un autre algorithme mémétique nommé *GASAT* [Lardeux et al., 2005a] [Lardeux , 2005] semble donner de bons résultats. Ce dernier hybride une méthode génétique et une méthode de recherche locale. Les opérateurs de base de l'algorithme génétique de *GASAT* sont un opérateur de sélection, une condition d'insertion et un opérateur de croisement. GASAT contient deux nouveaux croisements spécifiquement développés pour les problèmes de satisfiabilité fournissent de bons résultats, basés sur la structure des clauses de l'instance étudiée. Les deux nouvelles opérations de croisement permettent de diversifier la recherche en fournissant des individus avec de bonnes propriétés se trouvant dans des zones prometteuses de l'espace de recherche. Ces zones nécessitent une intensification de la recherche qui est effectuée par une méthode de recherche locale. La méthode de recherche

locale utilisée dans *GASAT* est une méthode tabou. À cette méthode de base, deux mécanismes spécifiquement développés pour les problèmes de satisfiabilité ont été ajoutés et permettent de mieux guider la recherche. Le premier mécanisme intensifie la recherche en réduisant le voisinage et le second permet de légèrement diversifier la recherche en identifiant des clause-butoirs et en s'assurant qu'elles soient satisfaites [Lardeux, 2005].

Algorithme 4.9 : l'heuristique flip pour MAX SAT :
Données: une formule ϕ en CNF, une affectation A
Résultat: une affectation

```
début
    improve:=1;
    tant que (improve>0) faire
        improve:=0;
        pour j:=1 jusqu'à nVar faire
            flip le j^ème variable de A;
            si (Amélioration(A[xj← flip(A| xj)])>0) alors
                A = A[x ← flip(A|x)];
                improve:=improve + Amélioration(A[xj←flip(A| xj)]);
            fin si
        fin pour
    fin tant que
Fin
```

Les algorithmes développés jusqu'ici pour le problème SAT fournissent des résultats intéressants mais ne s'avèrent pas encore aussi efficaces que les algorithmes de recherche locale pure. Toutefois, ce type de combinaison ayant déjà fait ses preuves pour la résolution d'autres problèmes combinatoires, il nous a semblé pertinent de pousser plus avant des investigations dans ce sens afin de prouver que de bons résultats pouvaient être obtenus avec de tels algorithmes hybrides pour les problèmes de satisfiabilité. Nos méthodes proposées pour le problème MAX-SAT dans le chapitre 6 sont basées sur ce mécanisme d'hybridation.

4.4.2.3 Méthodes hybrides entre méthodes complètes et incomplètes

Alors que les méthodes exactes sont à même de fournir une solution optimale pour les instances de petite taille, les méthodes approchées sont en général capables de trouver des solutions proches de l'optimum pour des instances de grandes tailles. C'est donc tout naturellement que, partant de cette observation, de nombreuses hybridations de ces deux paradigmes de résolution ont été envisagées. L'hybridation revient d'un coté à utiliser des heuristiques inspirées de DPLL pour la recherche locale et réciproquement des heuristiques inspirées de la recherche locale pour DPLL. Dans ces combinaisons, les méthodes approchées

sont chargées d'identifier rapidement des zones prometteuses de l'espace de recherche dans lesquelles les méthodes exactes pratiquent une exploration exhaustive. On trouve dans la littérature plusieurs méthodes de ce genre d'hybridation. Mazure dans [Mazure et al., 1998] utilise la recherche tabou comme méthode de branchement dans l'algorithme exact de Davis et Putnam. Lardeux dans [Lardeux et al., 2005b] utilise une procédure Branch and Bound comme une procédure d'intensification. Cependant, Boughaci dans [Boughaci et al, 2007] combine un algorithme génétique avec l'algorithme exact de propagation unitaire pour résoudre le problème MAX SAT.

Une des principales difficultés liées à ces hybridations réside dans le fait que les différentes méthodes n'utilisent pas nécessairement la même représentation de l'espace de recherche, rendant ainsi leur utilisation conjointe délicate.

4.5. Utilisation des solveurs SAT dans le model-checking

Récemment, plusieurs suggestions ont été proposées pour remplacer les ROBDDs dans les modèle-checking par des solveurs SAT afin d'augmenter leurs rendements. Des outils qui utilisent plutôt des moteurs de test de satisfaction booléenne à la place de BDD sont apparus [McMillan, 2003]. Les procédures de SAT ne souffrent pas du problème d'explosion de l'espace des méthodes basées BDD. En effet, les solveurs modernes de SAT peuvent traiter des problèmes propositionnels avec des centaines de milliers de variables ou de plus. Le premier model-cheking basé sur un solveur SAT est connu sous le nom de *méthode de vérification par modèle borné* (BMC pour Bounded Model Checking), proposée par Biere en [Biere et al, 1999]. Ils effectuent une vérification de modèle bornée en donnant un état initial et un nombre de pas fixes. L'objectif est de chercher rapidement un assignement pour les variables qui rend la formule booléenne fausse (figure 4.3).

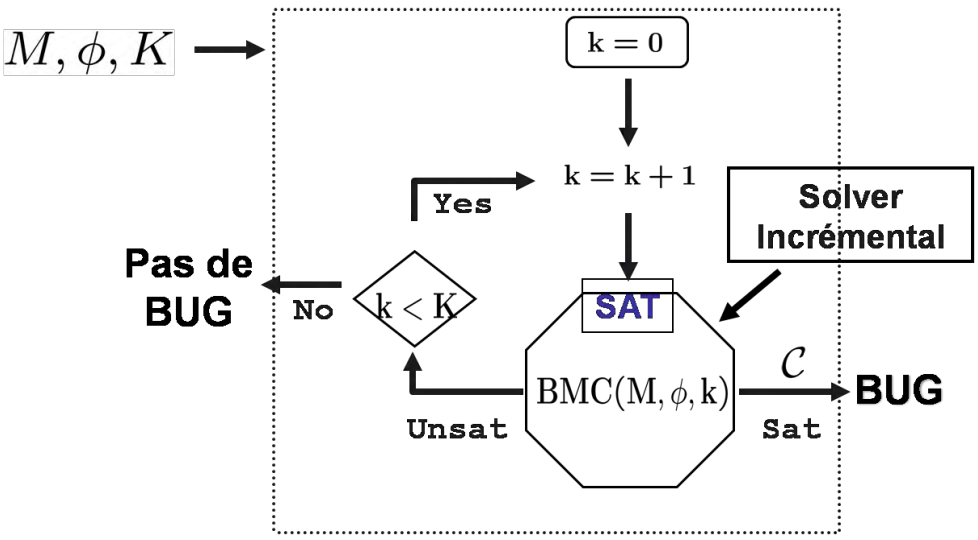


Figure 4.3 – Un exemple d'utilisation de Bounded Model Checking.

L'idée fondamentale dans BMC est de chercher un contre-exemple dans les exécutions dont la longueur est liée à un certain nombre entier K. Si aucun bogue n'est pas trouvé alors on augmente k jusqu'à ce qu'un bogue soit trouvé, ou le problème devient insurmontable, ou une certaine limite supérieure pré-connue est atteinte (cette limite s'appelle le seuil de perfection de la conception). En comparant cette technique avec les modele-checking basés sur les BDDs, on remarque qu‟elle est une méthode incomplète. En effet, cette technique ne garantit pas que la propriété soit valide dans tous les cas possibles, mais elle reste utile car elle est applicable sur des circuits de taille plus grande que ceux traités par la vérification de modèle classique. Il existe plusieurs vérificateurs de modèles industriels basés sur les mêmes principes comme *FormalCheck.* D‟autre part, il existe des softwares qui combinent les deux types de modèle-checking (SAT et BDD) comme l‟application nuSMV (figure 4.4).

Par ailleurs, BMC ne résout pas complètement le problème de complexité de la vérification de modèle, puisqu'il se fonde toujours sur un procédé exponentiel et par conséquent il est limité dans sa capacité, bien que des expériences aient prouvé qu'il peut résoudre beaucoup de cas qui ne peuvent pas être résolus par des techniques basées BDD. L'inverse est également vraie : il y a des problèmes qui sont mieux résolus par des techniques basées BDD que celle basées SAT. En outre, dans la plupart des cas réalistes, BMC a également l'inconvénient de ne pas pouvoir prouver l'absence des erreurs. Par conséquent, BMC joint l'arsenal des outils automatiques de vérification mais ne remplace pas n'importe lequel outil dans cet arsenal.

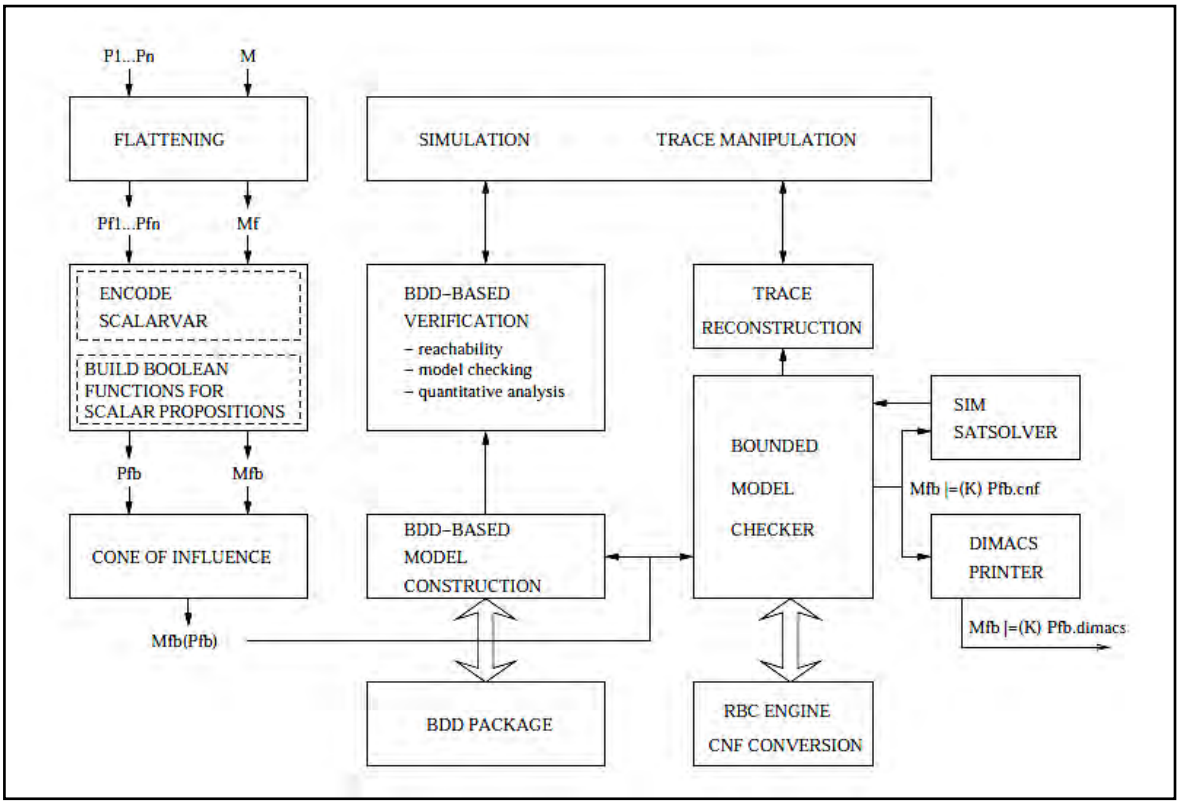


Figure 4.4 – L'architecture interne de nuSMV.

4.6. Conclusion

Dans ce chapitre nous avons présenté le problème SAT et ses variantes comme le problème MAX-SAT, ainsi que les deux principales classes des méthodes de résolution utilisées : méthodes exactes et les méthodes approchées. Ces méthodes ont chacune leurs caractéristiques, avantages et inconvénients. Les méthodes exactes permettent de répondre au problème de décision (SAT) et l'obtention de la meilleure borne pour le problème d'optimisation (MAX-SAT) en parcourant un arbre de recherche jusqu'à obtenir une solution. L'inconvénient de ces méthodes est qu'elles s'avèrent peu efficaces sur des instances de grande taille. Au contraire, les méthodes approchées ne sont en général pas limitées par la taille des instances mais ne permettent pas de déterminer le non satisfiabilité ou d'obtenir une borne optimale. Elles travaillent sur l'ensemble des affectations possibles et ne peuvent donc en explorer que certaines parties. Chacune des méthodes possédant leurs propres qualités, leur hybridation a pour but d'obtenir des algorithmes plus performants.

Le chapitre 6 est consacré à la présentation des approches réalisées au cours de cette thèse pour résoudre le problème de satisfiabilité maximale.

CHAPITRE 5

Approches quantiques évolutionnaires pour la minimisation d'un BDD

Si n'il ya pas de solution,
c'est qu'il n'y a pas de problème

Devise Shadok

Ce chapitre constitue une mise en application des concepts et méthodes détaillés dans les chapitres précédents. Le but de chapitre est de montrer nos contributions dans le domaine de l'ordonnancement des variables d'un BDD. Nous allons présenter deux approches hybrides développées au cours de ce travail de doctorat. Des tests entre ces approches et d'autres algorithmes de référence permettent de mettre en avant les qualités de ces approches.

Une partie des travaux présentés dans ce chapitre ont fait l'objet de plusieurs publications.

Sommaire

5.1. Introduction

Nous avons vu dans le chapitre 3 que la recherche d'un ordre des variables qui diminue au maximum la taille d'un diagramme de décision binaire est une tâche laborieuse. En effet, la recherche du meilleur ordre de variables pour lequel la taille du BDD correspondant est minimale est un problème d'optimisation combinatoire caractérisé par une grande complexité temporelle et spatiale. Dans le cas où l'on souhaite résoudre d'une manière exacte le problème considéré, des techniques comme la programmation dynamique ou la méthode de branch-and-bound peuvent être utilisées. Néanmoins, il est utile de rappeler que les temps d'exécution de ces méthodes deviennent de plus en plus prohibitifs à mesure que les données deviennent de plus en plus grandes. Pour cela, plusieurs méthodes basées sur des heuristiques ont été proposées pour trouver le bon ordre de variables d'un BDD. Malheureusement, aucune d'elles n'a résolu le problème efficacement.

Dans ce travail de doctorat, on s'est intéressé à la résolution de ce problème en investiguant l'apport des approches dites quantiques évolutionnaires. Pour cela, un noyau quantique évolutionnaire a été d'abord défini en première étape. Le développement de ce noyau a nécessité la définition d'une représentation quantique appropriée ainsi qu'une dynamique évolutionnaire opérant sur cette représentation (figure 5.1). Sur la base de ce noyau deux approches ont été proposées : l'approche QGABDD [layeb et al, 2007] et l'approche QDEBDD [Layeb et al, 2008a] [Layeb et al, 2008d]. La première approche est un Algorithme Quantique Génétique (AQG) simple pour résoudre le problème d'ordonnancement des variables d'un BDD, et la deuxième approche est basée sur un autre algorithme évolutionnaire différent des AQG qui est *l'Algorithme Quantique à Evolution Différentielle (AQED)*. Ce dernier a été couplé avec une procédure de recherche locale afin d'augmenter ses performances d'optimisation. Les résultats obtenus sont très encourageants et comparables voir meilleurs que ceux obtenus avec des approches populaires utilisées à ce jour.

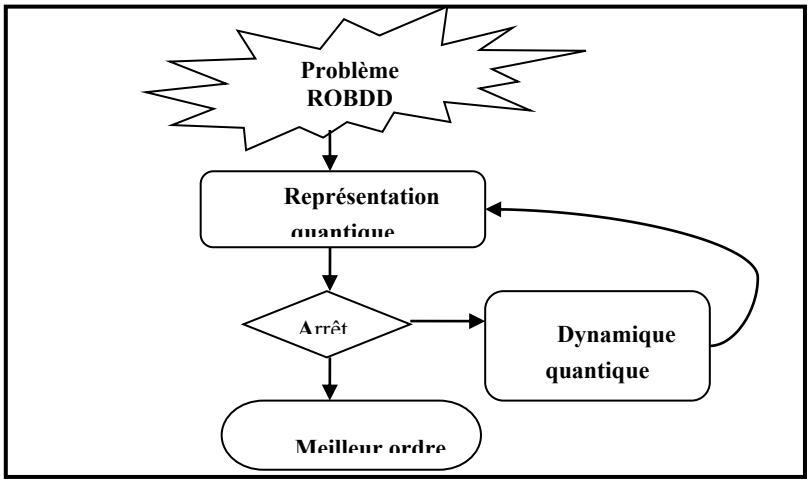


Figure 5.1 – Noyau quantique évolutionnaire pour le problème de la recherche d'ordre de variables d'un ROBDD.

5.2. Formulation du problème

Le problème d'ordonnancement des variables des BDDs peut être formulé comme suit : Etant donné une formule booléenne de ***n*** variables $f(x_1, x_2, x_3, ..., x_n)$. Le nombre de nœuds nécessaires pour représenter cette formule à l'aide des BDDs dépend étroitement de l'ordre des variables choisi. Le problème consiste à trouver le meilleur ordre des variables (x_1, x_2, x_3, …, x_n), pour lequel la taille du BDD représentant cette formule soit minimale. Ce problème est connu comme étant NP-Complet [Bollig et al., 1996]. En effet, lorsque le nombre de variable augmente le nombre de solutions faisables augmente de manière exponentielle. Ainsi, il est impossible de trouver une solution optimale à ce problème en un temps raisonnable. Si ***n*** est le nombre de variables, le nombre d'ordre possible est de ***(n-1) !/2***. En conséquence, Il est préférable de faire appel aux méthodes approchées afin de résoudre ce problème. Mathématiquement on peut formuler le problème comme suit [Layeb et al, 2007]:

Ayant un ensemble de variables $V=(x_1, x_2, x_3, ..., x_n)$. Le problème est défini par la spécification du couple ***(Ord, SC)*** où ***Ord*** est l'ensemble de tous les ordres possibles et ***SC*** est une fonction de **Ord** vers l'ensemble des entiers ***N*** qui donne la taille de BDD correspondant à un ordre donné. Chaque ordre peut être vu comme une permutation de l'ensemble des variables V=(x_1, x_2, x_3, …, x_n). En conséquence, le problème consiste à trouver la meilleure permutation qui correspond à une taille minimale du BDD.

5.3. Définition d'un noyau quantique évolutionnaire

Les deux méthodes développées pour la résolution du problème d'ordre de variables d'un BDD se basent sur un noyau d'optimisation commun. En effet, nous avons développé un noyau basé sur les principes des algorithmes évolutionnaires quantiques pour résoudre le problème sous-jacent. Pour cela, une représentation quantique de l'espace de recherche associé au problème a été développée. En outre, nous avons proposé un ensemble d'opérations quantiques constituant la dynamique quantique utilisée pour explorer l'espace de recherche en opérant sur la représentation quantique des solutions du problème de l'ordonnancement des variables des BDDs.

5.3.1. Représentation binaire d'un ordre des variables

Le codage des individus est une étape très importante dans les algorithmes évolutionnaires. Il existe plusieurs façons pour coder un ordre des variables d'un BDD, comme le codage entier, le codage binaire, etc. Néanmoins, le codage le plus adapté pour les algorithmes à base du quantique est le codage binaire parce que les solutions binaires sont faciles à manipuler par les opérations quantiques telle l'opération de mesure et l'interférence. De ce fait, l'ordre de variables est représenté sous forme d'une matrice binaire (figure 5.2) satisfaisant les contraintes suivantes [layeb et al, 2007] [Layeb et al, 2008a]:

- Pour ***N*** variables, la taille de la matrice binaire est ***N*N***,
- Les colonnes représentent les variables et les lignes représentent leurs rangs dans la solution,
- Dans chaque ligne et colonne, on trouve un seul 1, une variable n'a pas deux rangs et deux variables n'ont pas le même rang.

La figure suivante montre le codage binaire de l'ordre des variables *x2, x1, x4, x3* pour l'ensemble de variables *V={x1, x2, x3, x4}*.

$$\begin{pmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

Figure. 5.2 – La représentation binaire matricielle de l'ordre {x2, x1, x4, x3}.

5.3.2. La représentation quantique d'un ordre de variables d'un BDD

Comme les algorithmes évolutionnaires classiques, on a utilisé des chromosomes quantiques pour encoder les ordres de variables potentiels. Chaque chromosome quantique contient un ensemble de registres quantiques (figure 5.3). Chaque registre représente le rang d'une variable dans un ordre, comme il est montré sur la figure 5.4. Chaque colonne $\begin{pmatrix} a_i & b_i \end{pmatrix}^t$ représente un seul qubit et correspond à un 0 ou 1. Les a_i et b_i sont des valeurs réelles satisfaisant la relation d'unicité $|a_i|^2 + |b_i|^2 = 1$. Pour chaque qubit une valeur binaire est extraite selon les probabilités $|a_i|^2$ et $|b_i|^2$ et le nombre de 1 dans chaque colonne (ligne). Par conséquent, tous les ordres possibles peuvent être représentés par une matrice quantique (figure 5.3) contenant une superposition de toutes les configurations possibles. Cette matrice quantique peut être vue comme une représentation probabiliste de tous les ordres possibles des variables. Cette représentation est efficace quand une approche évolutionnaire est adaptée car elle contribue à la réduction de la taille de la population des chromosomes.

$$\begin{bmatrix} \begin{pmatrix} a_{11} \\ b_{11} \end{pmatrix} & \begin{pmatrix} a_{12} \\ b_{12} \end{pmatrix} & \cdots & \begin{pmatrix} a_{1m} \\ b_{1m} \end{pmatrix} \\ \begin{pmatrix} a_{21} \\ b_{21} \end{pmatrix} & \begin{pmatrix} a_{22} \\ b_{22} \end{pmatrix} & \cdots & \begin{pmatrix} a_{2m} \\ b_{2m} \end{pmatrix} \\ & \vdots & & \\ \begin{pmatrix} a_{n1} \\ b_{n1} \end{pmatrix} & \begin{pmatrix} a_{n2} \\ b_{n2} \end{pmatrix} & \cdots & \begin{pmatrix} a_{nm} \\ b_{nm} \end{pmatrix} \end{bmatrix}$$

Figure 5.3 – Représentation quantique d'un ordre de variables.

$$\left(\begin{array}{c|c|c|c} a_1 & a_2 & \cdots & a_m \\ b_1 & b_2 & \cdots & b_m \end{array}\right)$$

Figure 5.4 – Un registre quantique représentant le rang d'une variable.

5.3.3. Les opérations quantiques de base

Nous avons adapté un ensemble d'opérations quantiques complémentaires pour le problème d'ordonnancement de variables des BDDs. Ces opérations qui sont à la base de la dynamique quantique évolutionnaire proposée sont les suivantes :

5.3.3.1 L'opération de mesure

Cette opération est très importante car elle nous permet d'obtenir des chromosomes binaires à partir des chromosomes quantiques. Cette opération transforme par projection la matrice quantique en une matrice binaire (figure 5.5). Donc, on aura une solution parmi toutes les solutions présentes dans la superposition. Mais contrairement à la théorie quantique pure, cette mesure ne détruit pas la superposition. Cela a l'avantage de préserver la superposition pour les itérations suivantes sachant qu'on opère sur des machines classiques. La valeur d'un qubit est calculée selon ses probabilités $|a_i|^2$, $|b_i|^2$ et le nombre de 1 dans chaque colonne (ligne). La matrice binaire est ensuite traduite en une suite de nombres numériques. Donc, on aura une solution possible à notre problème parmi toutes les solutions codées dans la matrice quantique. En outre, l'opération de mesure joue également un autre rôle, celui d'un opérateur de diversification. En effet, deux mesures successives ne donnent pas forcément la même solution ce qui augmente les capacités de diversification de notre approche.

$$\begin{pmatrix} \begin{pmatrix}0.99\\0.14\end{pmatrix} & \begin{pmatrix}0.3778\\-0.9259\end{pmatrix} & \begin{pmatrix}0.14\\0.99\end{pmatrix} & \begin{pmatrix}0.8884\\-0.459\end{pmatrix} \\ \begin{pmatrix}0.14\\0.99\end{pmatrix} & \begin{pmatrix}0.3778\\-0.9259\end{pmatrix} & \begin{pmatrix}0.99\\0.14\end{pmatrix} & \begin{pmatrix}0.14\\0.99\end{pmatrix} \\ \begin{pmatrix}0.14\\0.99\end{pmatrix} & \begin{pmatrix}0.14\\0.99\end{pmatrix} & \begin{pmatrix}0.3778\\-0.9259\end{pmatrix} & \begin{pmatrix}0.8884\\-0.459\end{pmatrix} \\ \begin{pmatrix}0.8884\\-0.459\end{pmatrix} & \begin{pmatrix}0.8884\\-0.459\end{pmatrix} & \begin{pmatrix}0.14\\0.99\end{pmatrix} & \begin{pmatrix}0.99\\0.14\end{pmatrix} \end{pmatrix} \xrightarrow{\text{Mésure}} \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

Figure 5.5 – Projection d'une matrice quantique.

5.3.3.2 L'interférence quantique

Cette opération amplifie l'amplitude de la meilleure solution et diminue les amplitudes des mauvaises solutions. Elle consiste essentiellement à déplacer l'état de chaque qubit dans la direction de la valeur du bit correspondant dans la meilleure solution en cours. L'opération d'interférence permet donc d'intensifier la recherche autour de la meilleure solution. Elle peut être accomplie en utilisant une transformation unitaire qui effectue une rotation dont les

angles sont fonction des amplitudes a_i, b_i (Figure 5.6) et de la valeur du bit correspondant dans la solution référence. Les valeurs de l'angle δθ sont choisies empiriquement de telle manière à éviter une convergence prématurée. La direction de la rotation est déterminée suivant la table 5.1. De point de vue de l'algorithme évolutionnaire, cette opération peut être vue comme une opération de mutation spécifique.

5.3.3.3 La mutation quantique

Cette opération permet l'exploration de nouvelles solutions et l'augmentation des capacités de diversification du processus de recherche. Ainsi, le processus de mutation permet d'explorer davantage l'espace de recherche et peut aider l'algorithme à éviter une stagnation autour d'optima locaux. Elle consiste en l'application d'une perturbation sur un ensemble de qubits de la matrice quantique (Figure 5.7). Cette mutation peut être une mutation simple de qubits, comme elle peut être remplacée par une méthode de recherche locale

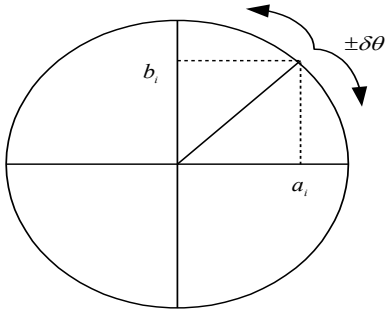


Figure 5.6 – Interférence quantique.

Table 5.1. Valeurs prises par l'angle de rotation.

Angle	Reference bit value	a_i	b_i
+δθ	1	> 0	> 0
-δθ	0	> 0	> 0
-δθ	1	> 0	< 0
+δθ	0	> 0	< 0
-δθ	1	< 0	> 0
+δθ	0	< 0	> 0
+δθ	1	< 0	< 0
-δθ	0	< 0	< 0

$$\begin{pmatrix} \begin{pmatrix} 0.99 \\ 0.14 \end{pmatrix} & \begin{pmatrix} 0.37 \\ -0.92 \end{pmatrix} & \begin{pmatrix} 0.14 \\ 0.99 \end{pmatrix} & \begin{pmatrix} 0.88 \\ -0.45 \end{pmatrix} \\ \begin{pmatrix} 0.14 \\ 0.99 \end{pmatrix} & \begin{pmatrix} 0.37 \\ -0.92 \end{pmatrix} & \begin{pmatrix} 0.99 \\ 0.14 \end{pmatrix} & \begin{pmatrix} 0.14 \\ 0.99 \end{pmatrix} \\ \begin{pmatrix} 0.14 \\ 0.99 \end{pmatrix} & \begin{pmatrix} 0.99 \\ 0.14 \end{pmatrix} & \begin{pmatrix} 0.37 \\ -0.92 \end{pmatrix} & \begin{pmatrix} 0.88 \\ -0.45 \end{pmatrix} \\ \begin{pmatrix} 0.88 \\ -0.45 \end{pmatrix} & \begin{pmatrix} 0.88 \\ -0.45 \end{pmatrix} & \begin{pmatrix} 0.14 \\ 0.99 \end{pmatrix} & \begin{pmatrix} 0.99 \\ 0.14 \end{pmatrix} \end{pmatrix} \xrightarrow{\text{Mutation}} \begin{pmatrix} \begin{pmatrix} \mathbf{0.14} \\ \mathbf{0.99} \end{pmatrix} & \begin{pmatrix} 0.37 \\ -0.92 \end{pmatrix} & \begin{pmatrix} \mathbf{0.99} \\ \mathbf{0.14} \end{pmatrix} & \begin{pmatrix} 0.88 \\ -0.45 \end{pmatrix} \\ \begin{pmatrix} 0.14 \\ 0.99 \end{pmatrix} & \begin{pmatrix} 0.37 \\ -0.92 \end{pmatrix} & \begin{pmatrix} 0.99 \\ 0.14 \end{pmatrix} & \begin{pmatrix} 0.14 \\ 0.99 \end{pmatrix} \\ \begin{pmatrix} 0.14 \\ 0.99 \end{pmatrix} & \begin{pmatrix} 0.99 \\ 0.14 \end{pmatrix} & \begin{pmatrix} 0.37 \\ -0.92 \end{pmatrix} & \begin{pmatrix} 0.88 \\ -0.45 \end{pmatrix} \\ \begin{pmatrix} 0.88 \\ -0.45 \end{pmatrix} & \begin{pmatrix} \mathbf{0.14} \\ \mathbf{0.99} \end{pmatrix} & \begin{pmatrix} \mathbf{0.88} \\ \mathbf{-0.45} \end{pmatrix} & \begin{pmatrix} 0.99 \\ 0.14 \end{pmatrix} \end{pmatrix}$$

Figure. 5.7 – Opération de mutation simple.

5.4. Un Algorithme Génétique Quantique pour le problème ROBDD

Nous allons présenter dans cette section notre premier algorithme développé pour le problème d'ordonnancement des variables d'un BDD. L'approche QGABDD a été proposée à cet effet [Layeb et al, 2007]. Cette nouvelle approche est basée sur le noyau quantique défini auparavant. QGABDD repose sur une hybridation de ce noyau quantique et un algorithme génétique comme il est montré par la figure suivante.

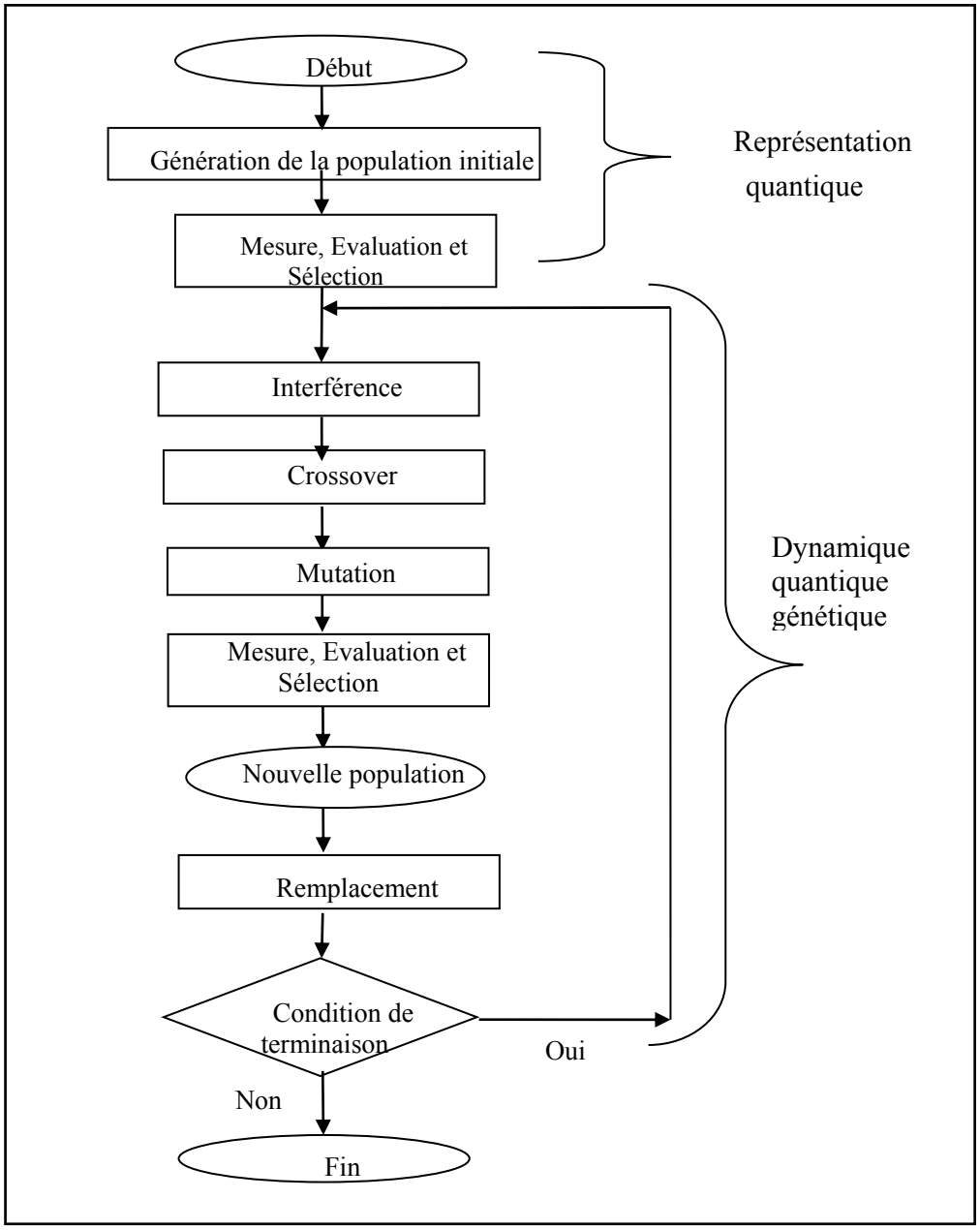


Figure 5.8 – La structure générale de QGABDD.

Nous décrivons, dans cette partie, les étapes principales de l'algorithme pour résoudre le problème sous-jacent. L'idée de cette approche consiste à partir d'une population initiale de solutions et de tenter de l'améliorer en appliquant itérativement une série de raffinements

afin de trouver une solution optimale. D'abord dans l'étape initialisation, une population de chromosomes quantiques identiques est construite. Pour cela, tous les chromosomes quantiques sont initialisés avec la même constante $\frac{1}{\sqrt{2}}$. D'une autre façon, les chromosomes quantiques représentent des superpositions linéaires de tous les états possibles avec la même probabilité (équation 5.1). Dans le cas des BDDs, un chromosome quantique représente une superposition de tous les ordonnancements possibles.

$$\left|\psi_{q_j^0}\right\rangle = \sum_{k=1}^{2^m} \frac{1}{\sqrt{2^m}} \left|X_k\right\rangle \tag{5.1}$$

Où X_k est le k^{ieme} état représenté par la chaîne binaire $(x_1\, x_2 \ldots x_m)$, où x_i, i = 1,2,…, m.

Il faut noter, qu'afin de réduire le temps d'exécution, il est recommandé de démarrer avec une population diverse contenant des solutions de bonnes et de mauvaises qualités. Pour cela, on peut par exemple utiliser la méthode de recherche en profondeur (Depth-First Search) [Malik et al, 1988], ou bien utiliser l'une des heuristiques implémentés dans le package de construction et de manipulation des BDDs *Buddy* [Lind-Nielsen, 1999] afin d'injecter dans la population des solutions de bonnes qualités.

L'étape suivante consiste à évaluer la population initiale. L'évaluation de chaque chromosome est précédée par l'opération de mesure. Cette dernière extrait une matrice binaire *BM(0)* à partir de la matrice quantique *QM*(0) (figure 5.5). L'opération de mesure doit prendre en compte les probabilités $|a_i|^2$, $|b_i|^2$ et les contraintes de la représentation binaire d'un ordre de variables citées précédemment. Ensuite, la matrice binaire est traduite en un ordre de variables selon les règles citées dans la section 5.3.1 (figure 5.9). Cet ordre est utilisé pour la construction du BDD correspondant à la formule booléenne en entrée. Après cette étape, on calcule la fitness du BDD résultant. La fitness dans notre cas est le nombre de nœuds dans le BDD. Le meilleur individu est celui qui donne un nombre minimum des nœuds, plus la taille du BDD est minimale plus la solution obtenue est optimale.

$$\begin{matrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{matrix} \xrightarrow{Transformation} (2,1,4,3)$$

Figure 5.9 – Transformation d'une matrice en une suite d'entiers.

Il faut noter que la construction de la fonction qui fait l'évaluation des individus doit être particulièrement optimisée, car elle sera exécutée un grand nombre de fois, ce qui fait que la

rapidité de l'algorithme dépend essentiellement d'elle. Ils ont démontré que la fonction objectif consomme près de 90% de temps CPU. Dans le cas du problème de la recherche d'ordre des variables de BDD, la fonction d'évaluation est l'étape la plus gourmande en espace mémoire et temps d'exécution parce que pour chaque ordre trouvé, il faut construire le BDD correspondant. Pour remédier ce problème, une parallélisation de cette fonction est indispensable.

Dans chaque étape de la boucle de QGABDD, on applique d'abord l'opération d'interférence, elle permet l'amplification des amplitudes des qubits de la bonne solution et donc sa probabilité d'être mesurée. Deuxièmement, une opération de croisement quantique est appliquée, le croisement a pour but de produire une nouvelle génération d'individus en recombinant les individus sélectionnés par l'opérateur de sélection. Ainsi, dans un premier temps, les individus sélectionnés sont répartis aléatoirement en couples de parents. Puis, chaque couple de parents subit une opération de croisement afin de générer un ou deux enfants. Le rôle principal de l'opérateur de croisement est de produire des individus diversifiés et potentiellement prometteurs. Dans le cas du croisement quantique, on échange des blocs de qubits comme il est montré dans la figure 5.10.

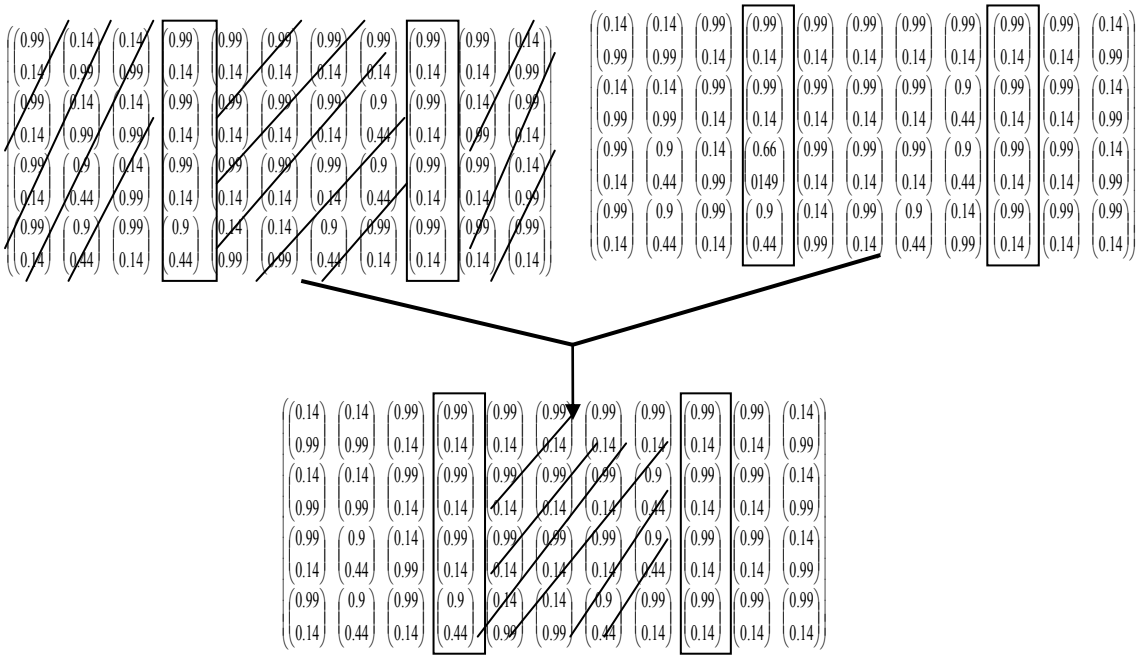


Figure 5.10 – Croisement quantique uniforme.

Dans l'étape suivante, on applique une perturbation sur les chromosomes afin de générer d'autres solutions. QGABDD utilise une opération de mutation simple, c'est une opération de permutation des qubits choisis aléatoirement (figure 5.11). L'opération de mutation est suivie de l'opération de mesure quantique, cette dernière permet d'avoir une autre population de matrice binaire BM(t+1). Ensuite, on procède à l'évaluation de chaque matrice binaire

BM(t+1). Finalement, on calcule la fitness de la nouvelle population. Si on trouve une solution meilleure que celle enregistrée alors on garde cette solution comme la meilleure solution courante sinon elle est rejetée. La phase de remplacement consiste à construire la nouvelle population de la génération suivante. Le critère de remplacement est le respect de la taille maximale de la population. Par conséquent, la nouvelle population est construite avec une partie des meilleurs éléments et une partie des éléments les moins performants afin de garder la diversité de la population et éviter une convergence rapide vers un optimum local. Le processus est réitéré jusqu'à la satisfaction des conditions finales.

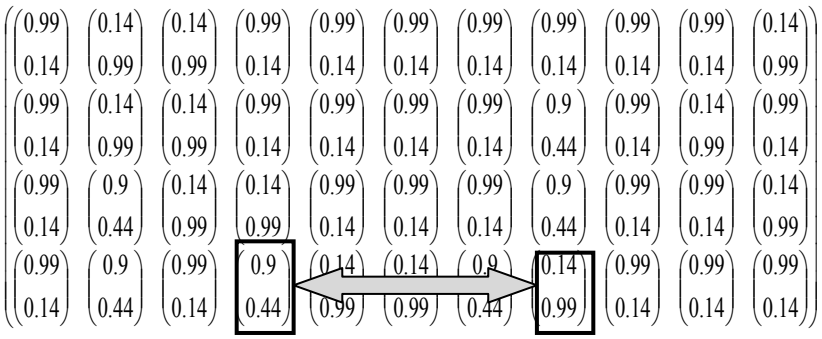


Figure 5.11– Opération de permutation.

5.5. Un Algorithme Quantique à Evolution Différentielle pour la Minimisation d'un ROBDD

Notre deuxième contribution dans le cadre de nos travaux sur la résolution du problème d'ordonnancement des variables des BDDs est basée sur un nouvel Algorithme Quantique à Evolution Différentielle (AQED) hybridé avec une méthode de recherche locale. L'Algorithme Quantique à Evolution Différentielle est un nouvel algorithme qui combine entre les idées de l'informatique quantique et l'algorithme à évolution différentielle. L'Algorithme Quantique à Evolution Différentielle a été utilisé pour traiter plusieurs problèmes d'optimisation combinatoire comme le traitement d'images [Draa et al, 10], les systèmes chaotiques, optimisation binaire [Haijun et al, 2008], etc. D'autre part, l'hybridation est choisie afin d'augmenter les performances de l'algorithme quantique à évolution différentielle pour la résolution du problème en question. Le principe général de cet algorithme hybride appartenant à classe des algorithmes memetiques (figure 5.12) [Bontoux et al., 2008] est le même que pour les algorithmes génétiques mis à part qu'un opérateur de recherche locale est ajouté après celui de la mutation. L'hybridation entre un algorithme d'optimisation global et une recherche locale est maintenant reconnue comme une approche compétitive pour traiter des problèmes combinatoires difficiles. Ces deux méthodes sont complémentaires, car l'une permet de se mouvoir rapidement dans l'espace de recherche et ainsi de diversifier la recherche alors que l'autre explore de manière intensive ces zones de l'espace de recherche. Leur combinaison vise à balayer rapidement les zones intéressantes de

l'espace de recherche. En effet, la partie évolutionnaire de ces algorithmes peut être vue comme une forte diversification alors que la partie recherche locale correspondrait à une forte intensification. Les résultats expérimentaux montrent que l'intégration de la méthode de recherche locale dans le noyau quantique a amélioré considérablement les performances de ces algorithmes évolutionnaires quantiques.

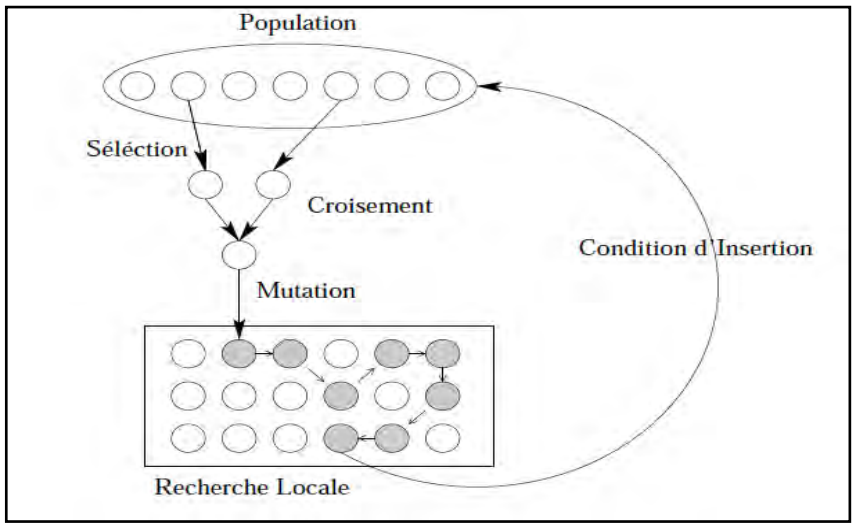


Figure 5.12 – Schéma des algorithmes mémétiques.

Dans ce contexte, nous proposons dans cette thèse, une nouvelle approche itérative nommée QDEBDD en se basant sur un Algorithme Quantique à Evolution Différentielle [Layeb et al, 2008a] [Layeb et al, 2008d]. Afin d'optimiser la résolution du problème sous jacent, l'algorithme résultant a été hybridé avec une méthode de recherche locale. Une manière utilisée pour hybrider un algorithme d'optimisation global et une méthode de recherche locale consiste à remplacer ou compléter l'opérateur de mutation de l'algorithme d'optimisation global par un opérateur de recherche locale. En partant de cette idée, nous avons intégré la méthode de la descente (Hill-Climbing) en tant que mutation dans la dynamique évolutionnaire quantique de notre approche. Bien qu'il existe d'autres méthodes de recherche locale plus performantes que le hill-climbing telle que la méthode tabou ou le tout récent algorithme de recherche local: l'algorithme de *recherche à voisinage variable*, notre choix se justifie par la simplicité de l'algorithme hill-climbing et ses bonnes performances. D'un autre côté, nous n'avons pas utilisé dans notre méthode l'opération de croisement différentiel, car les opérations quantiques telles que l'interférence, offrent plus de diversité à la population, et elle peut donc substituer le croisement classique. Finalement, des expériences ont été effectuées pour montrer la faisabilité et l'efficacité de notre approche. La description générale de notre algorithme est donnée par l'algorithme suivant :

Algorithme 5.2 : Schéma de QDEBDD
Données: un ensemble de variables, une formule booléenne.
Sorties: ord_{best} et $SC(ord_{best})$.

```
Début
    construire la population initiale.
    évaluer la population initiale.
    mettre ord_best = ord' et SC_best = SC ( ord').
    Répéter (pour chaque chromosome)
     Appliquer l'opération d'interférence.
     Appliquer la mutation quantique différentielle.
     Appliquer la procédure Hill Climbing.
     Evaluer l'ordre de variables courant ord'.
     Mise à jour  de la meilleure solution.
    jusqu'à la satisfaction des critères d'arrêt.
Fin
```

Nous expliquons maintenant les grandes étapes de l'algorithme QDEBDD :

5.5.1 La population initiale

Ayant un ensemble S de variables d'un BDD à ordonner. Tout d'abord, des solutions initiales sont encodées en N matrices quantiques représentant la population initiale PQM. La population initiale est la première manière avec laquelle un algorithme quantique à évolution différentielle démarre. Le processus de génération des individus de cette population doit être bien choisi afin de pouvoir fournir à l'algorithme un ensemble d'individus potentiels et diversifiés dans le but de se converger rapidement vers la meilleure solution. L'objectif est donc de trouver un nombre d'ordres de variables différents pour la même formule afin de pouvoir faire évoluer l'algorithme. Nous pouvons utiliser le schéma de population initiale proposé par l'algorithme à évolution différentielle de base ou engendrer une population initiale complètement au hasard. Dans notre cas, on a choisi la création de la population initiale de manière aléatoire. Néanmoins, il est important d'utiliser des heuristiques pour construire des solutions initiales de bonnes qualités et réduire donc le temps de convergence.

5.5.2. L'évaluation des individus

Afin de pouvoir sélectionner les meilleurs individus de la population courante, il faut tout d'abord évaluer ces individus afin d'établir un rangement d'efficacité ou d'adaptation. Pour notre problème l'évaluation des individus se fait selon la taille des BDDs correspondants aux

ordres des variables. Pour l'évaluation chaque solution, on doit se procéder comme dans la méthode QGABDD vue dans la section précédente.

5.5.3 La sélection

Cette opération permet de sélectionner les individus aptes à la reproduction et donc participer aux opérations évolutionnaires. Les individus aptes sont souvent les mieux classés lors de l'évaluation. Dans cette approche, la sélection consiste à combiner entre les meilleures solutions et les mauvaises solutions, afin de garder la diversité dans la population et éviter donc les risques des minima locaux.

5.5.4 Opérateur de mutation différentielle quantique

C'est l'idée primordiale derrière un AQED, et qui lui différencie d'un algorithme génétique. Dans la première étape de cette mutation, nous faisons la différence entre deux vecteurs choisis aléatoirement. Ensuite, cette différence est pondérée par un poids w ∈ R. Cette différence est ajoutée ensuite à un troisième individu (figure 5.13). Afin de maintenir la caractéristique de l'unité d'un qubit ($|a_i|^2 + |b_i|^2 = 1$), nous avons utilisé les angles pour représenter chaque qubit. En fait, nous pouvons donner à chaque qubit $Q(a_i, b_i)$ une représentation géométrique : Q1(cos(Φ1),sins(Φ1)).

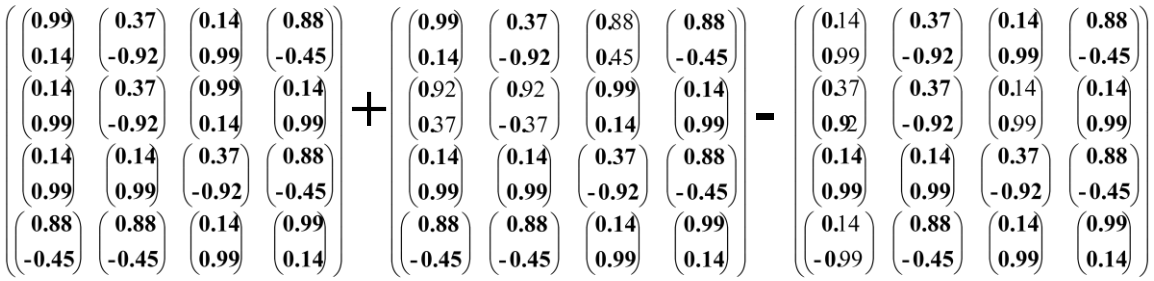


Figure 5.13 – Opérateur de mutation différentielle quantique.

5.5.5 Méthode de recherche locale

Afin d'accélérer la convergence de l'algorithme quantique à évolution différentielle, nous l'avons couplé avec une méthode de recherche locale. Nous avons utilisé une méthode simple qui est la descente ou le Hill Climbing [Aarts et al, 1997]. L'approche de la descente dépend essentiellement de la relation de voisinage à laquelle elle est associée. Il faut toutefois noter que la complexité d'une approche de résolution basée sur la recherche hill climbing dépend essentiellement de la taille du voisinage de la solution courante et de la méthode d'évaluation de chacun de ces voisins afin de déterminer celui qui minimise la fonction coût. Il a été démontré que près de 90% du temps de cette méthode est pris par l'évaluation des voisinages. Par conséquent, il est intéressant d'utiliser des voisinages aussi restreints que possible afin de diminuer au maximum la complexité de résolution. Dans notre méthode, le mécanisme pour

générer des voisins de solution est basé sur la sélection des fenêtres dynamiques sur chaque ordre (figure 5.14). Premièrement, on choisit un premier point à déplacer, puis d'un deuxième point qui va être la cible du déplacement. La taille de la fenêtre de déplacement est de taille variable afin de garantir plus de performance. Finalement, le segment entre les deux est inversée. Cette procédure s'appelle permutation de fenêtres (Windows permutation). L'efficacité d'une telle procédure réside alors dans sa capacité à correctement explorer des zones variées de l'espace de recherche mais également dans sa propension à converger vers un optimum local. Le schéma général de cette procédure est donné par l'algorithme 5.3.

En outre, la méthode développée est très générale qui reste simplifiée par rapport à la manipulation des variables et qui garde l'esprit des algorithmes quantiques évolutionnaires. Idéalement, cette méthode peut être combinée à d'autres algorithmes de recherche locale plus robustes comme la recherche tabou, ou utiliser des heuristiques liées à la résolution du problème d'ordre des variables des BDDs.

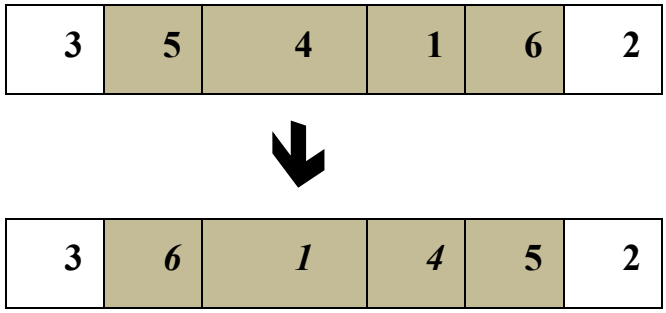


Figure 5.14 – Le mécanisme de la recherche locale *permutation de fenêtres.*

Algorithme 5.3 : La procédure Hill climbing pour ROBDD:
Données: un ordre ord.
Sorties: ord_{best} and $SC(ord_{best})$.

```
début
 Evaluer l'ordre ord
 Mettre ord_best=ord  et SC(ord_best)=sc(ord);
 Répéter
   Choisir dynamiquement un segment dans ord ;
    Inverser le segment choisi ;
    Evaluer le nouvel ordre ord' ;
    If sc(ord') > SC(ord_best) then
      ord_best = ord
 jusqu'à (nombre essais est atteint)
fin
```

5.5.6 Critère d'arrêt.

Les algorithmes quantiques à évolution différentielle sont des méthodes itératives qui ont besoin d'un grand nombre d'itérations pour pouvoir converger vers les solutions optimales. Le choix d'un critère d'arrêt peut se révéler une tâche très difficile car on ne sait pas si l'objectif de l'algorithme est atteint ou non. Pour notre algorithme, il semble que le nombre d'itération est le seul critère utilisable.

5.6. Implémentation et évaluation

Les méthodes proposées QGABDD et QDEBDD sont implémentées en java 1.5 et testés sur un micro-ordinateur avec 3 GHz et 1 GB sous Windows XP. On a utilisé le package *JAVABDD* [Whaley, 2005] qui contient un ensemble d'outils pour la création et la manipulation des BDDs en Java. Pour évaluer les performances de ces méthodes, des expériences ont été entreprises sur des ensembles de tests crées aléatoirement par un programme avec les portes logiques AND, XOR et NOT (table 5.1). Ces tests sont choisis de telle sorte à montrer les performances de chaque programme. Les tests sont classés selon le nombre de variables dans chaque test. En outre, les tests de la même classe peuvent ne pas avoir la même taille (le nombre de portes logiques).

Nous avons comparé les résultats des approches avec ceux de deux programmes connus dans ce domaine WIN2ITE et WIN3ITE. Ces deux programmes sont basés sur une méthode de recherche locale itérative intégrés dans la bibliothèque populaire de construction et manipulation des BDDs *Buddy* [Lind-Nielsen, 1999]. La différence entre WIN2ITE et WIN3ITE est la taille de la fenêtre de déplacement, dans WIN2ITE, elle est égale à deux tandis que dans WIN3ITE elle est trois. Finalement, pour la validation statistique de nos résultats, on a utilisé les tests statistiques de Friedman et de Wilcoxon "Wilcoxon signed rank" [Hollander, et al, 1999].

Le tableau 5.2 résume les résultats trouvés au cours de cette expérience. Les résultats montrent que l'approche QDEBDD est le meilleur programme dans cette expérience en termes de performance. En effet, les tests de Friedman et de Wilcoscon sur la globalité des tests confirment que les performances de QDEBDD sont totalement différentes que celles des autres programmes (Figure 5.15). Par ailleurs, le test de Friedman montre également qu'il n'existe pas de grandes différences entre les programmes QGABDD, WIN2ITE, et WIN3ITE.

D'un autre côté, La comparaison entre les méthodes QDEBDD et QGABDD montre l'utilité de la recherche locale. En effet, QDEBDD est meilleur que QGABDD parce que ce dernier utilise une simple opération de mutation, alors que QDEBDD est basée sur une méthode de recherche locale. Il est ainsi primordial d'intégrer une méthode de recherche locale pour trouver de bons résultats. Nous avons remarqué également que dans cette expérience que le programme WIN2ITE et meilleur que WIN3ITE.

Table 5.2. Les résultats obtenus.
(Les tests contiennent les opérateurs logiques AND, XOR et NOT)

Test	# variables	QDEBDD	WIN2ITE	QGABDD	WIN3ITE
Test40_1	40	36	31	47	53
Test40_2	40	232	342	242	138
Test40_3	40	26	30	21	27
Test40_4	40	405	175	437	319
Test40_5	40	360	563	385	443
Test60_1	60	1016	6240	1339	2216
Test60_2	60	739	1189	926	753
Test60_3	60	427	108	467	437
Test60_4	60	114	272	118	211
Test60_5	60	873	2191	2402	1451
Test80_1	80	305	259	443	447
Test80_2	80	169	60	362	392
Test80_3	80	311	239	603	785
Test80_4	80	359	1964	720	1720
Test80_5	80	904	606	1499	1990
Test100_1	100	6719	998	8275	10777
Test100_2	100	2507	5343	4315	3491
Test100_3	100	483	1167	798	1061
Test100_4	100	566	181	1003	1151
Test100_5	100	6952	7003	8574	8112
Test150_1	150	3301	857	15984	18330
Test150_2	150	6611	509	8699	11980
Test150_3	150	1428	7303	5442	6919
Test150_4	150	1909	4740	3516	4414
Test150_5	150	2130	8115	4726	7629
Test200_1	200	8359	11794	20273	21966
Test200_2	200	7114	14484	11295	13310
Test200_3	200	878	2027	1718	1995
Test200_4	200	4754	13673	27730	36593
Test200_5	200	112585	367620	209380	256879
Test250_1	250	447097	586488	443280	483304
Test250_2	250	1554	8505	6965	7885
Test250_3	250	12311	26923	21803	25043
Test250_4	250	259	649	510	589
Test250_5	250	54251	81961	60451	61518

D"autre part, si on compare les performances de ces programmes pour chaque classe de tests. On constate que dans les tests dont le nombre de variables est 40, le programme QDEBDD est légèrement supérieur sur les autres méthodes quoi que cette supériorité ne soit pas statistiquement significative selon le test de Friedman (Figure 5.16). Idem, pour les tests contenant 60 (Figure 5.17), 100 (Figure 5.19) et 150 variables (Figure 5.20). Néanmoins, dans les tests de 80 variables, on a noté que le programme WIN2ITE est le meilleur dans cet ensemble de tests, alors que WIN3ITE est moins bon que WIN2ITE dans ces tests (Figure 5.18). Dans les tests contenant 200 ou 250 variables, on distingue clairement les performances de nos programmes QGABDD et QDEBDD par rapport aux autres programmes. En effet dans les tests de 200 variables les programmes WIN3ITE et WIN2ITE ne sont pas réussis (Figure 5.21), alors dans le les tests 250 variables, seulement QGABDD et WIN3ITE sont les plus proches au programme QDEBDD (figure 5.22).

En termes de temps d"exécution, on remarque que les deux programmes WIN2ITE et WIN3ITE sont plus rapides que QGABDD et QDEBDD pour différentes raisons. Mais la lenteur de nos approches peut être surmontée en effectuant plusieurs modifications. La première, c"est l"utilisation du langage C++ comme le langage d"implémentation parce que le C++ est plus rapide que Java dans telles applications. Deuxièmement, le temps de calcul peut être considérablement réduit en démarrant avec une bonne solution initiale. Pour cela, l"intégration d"une heuristique dans ces approches pour calculer la solution initiale peut être bénéfique. Troisièmement, afin de réduire le temps de calcul, la parallélisation de ces approches peut apporter plus de rapidité. En effet, la parallélisation permet d'accélérer la recherche de l'optimum et d'élargir l'espace de recherche exploré.

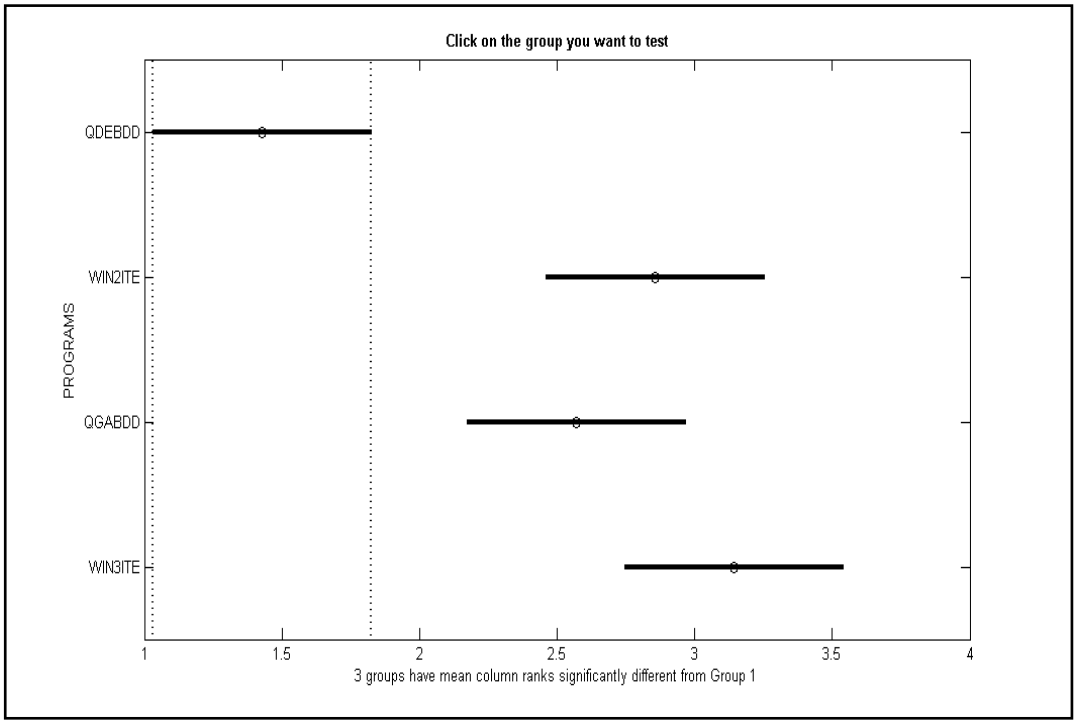


Figure 5.15 – Test de Friedman (a=0.05) pour l'ensemble de tests.

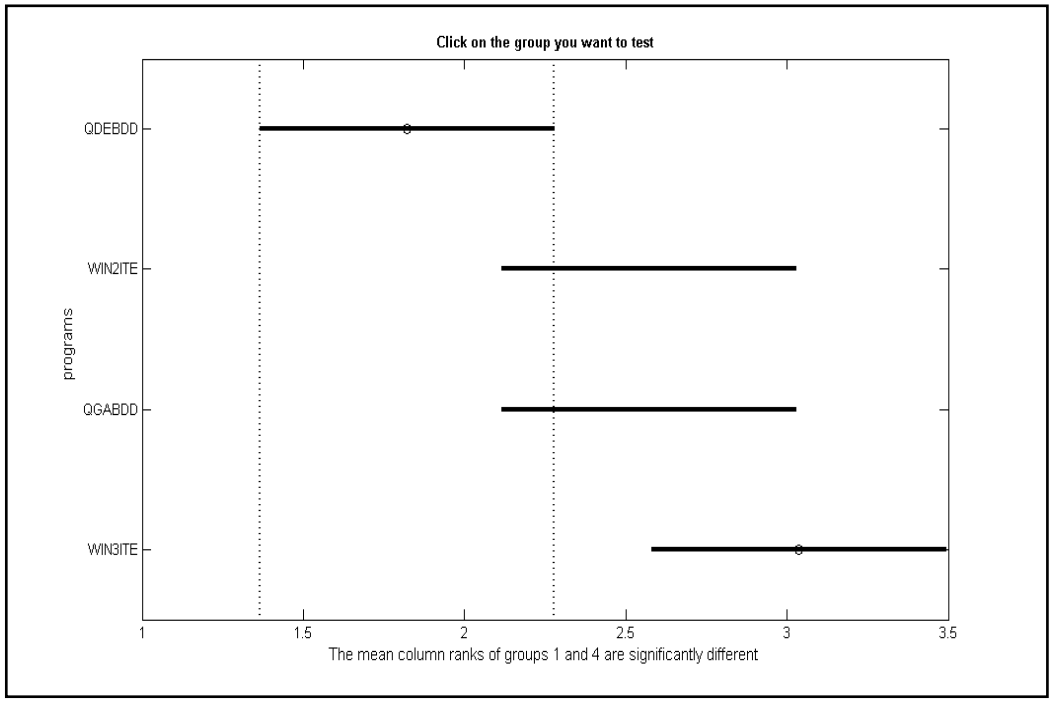


Figure 5.16 – Test de Friedman (a=0.01) pour l'ensemble de tests de 40 variables.

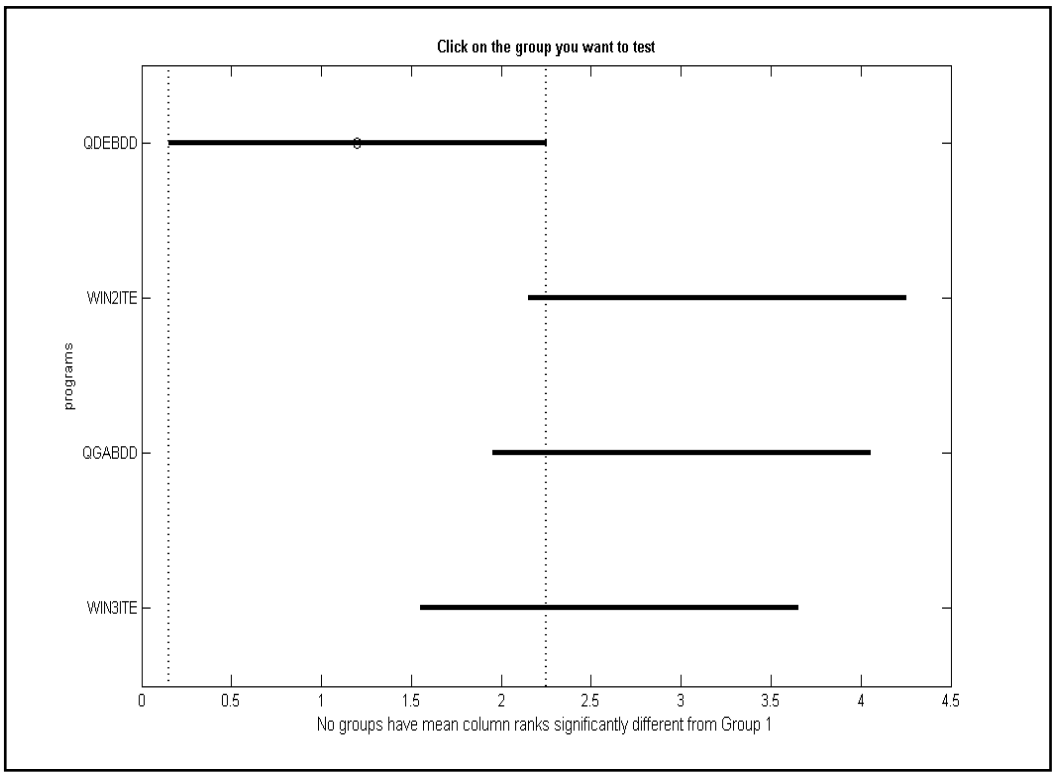


Figure 5.17 – Test de Friedman (a=0.01) pour l'ensemble de tests de 60 variables.

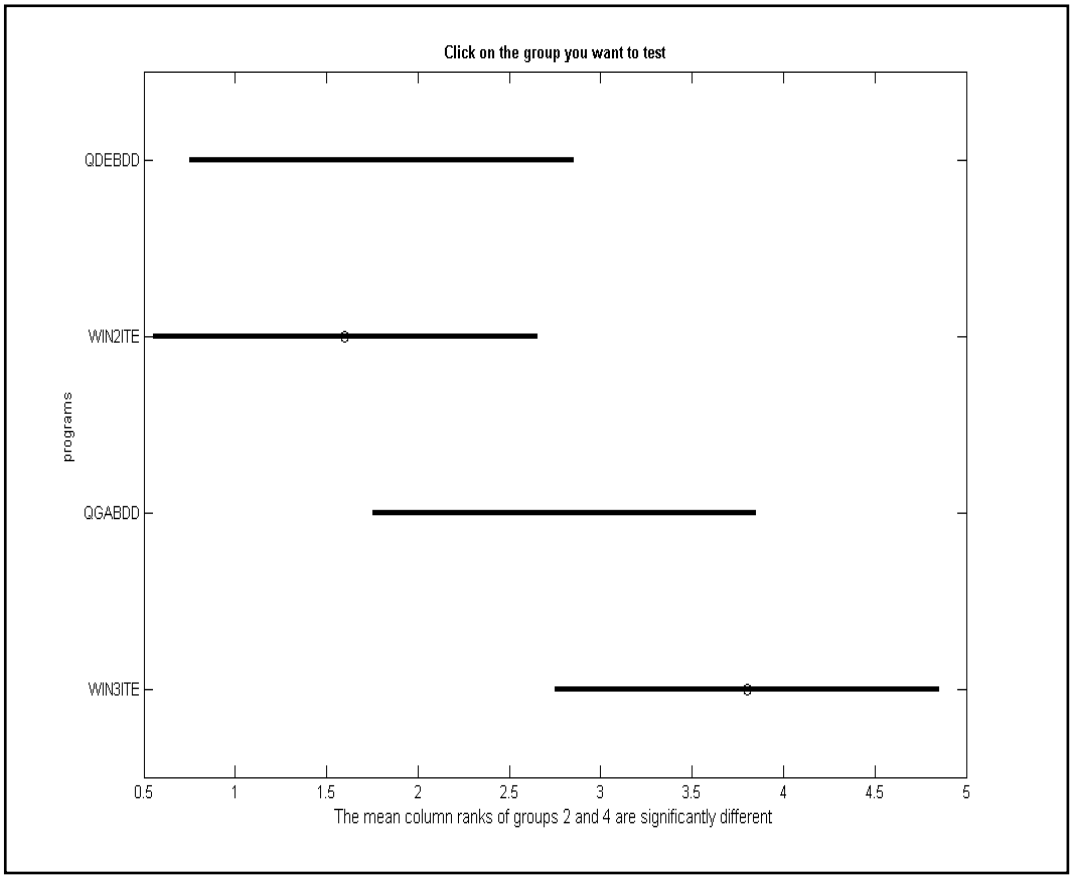


Figure 5.18 – Test de Friedman (a=0.01) pour l'ensemble de tests de 80 variables.

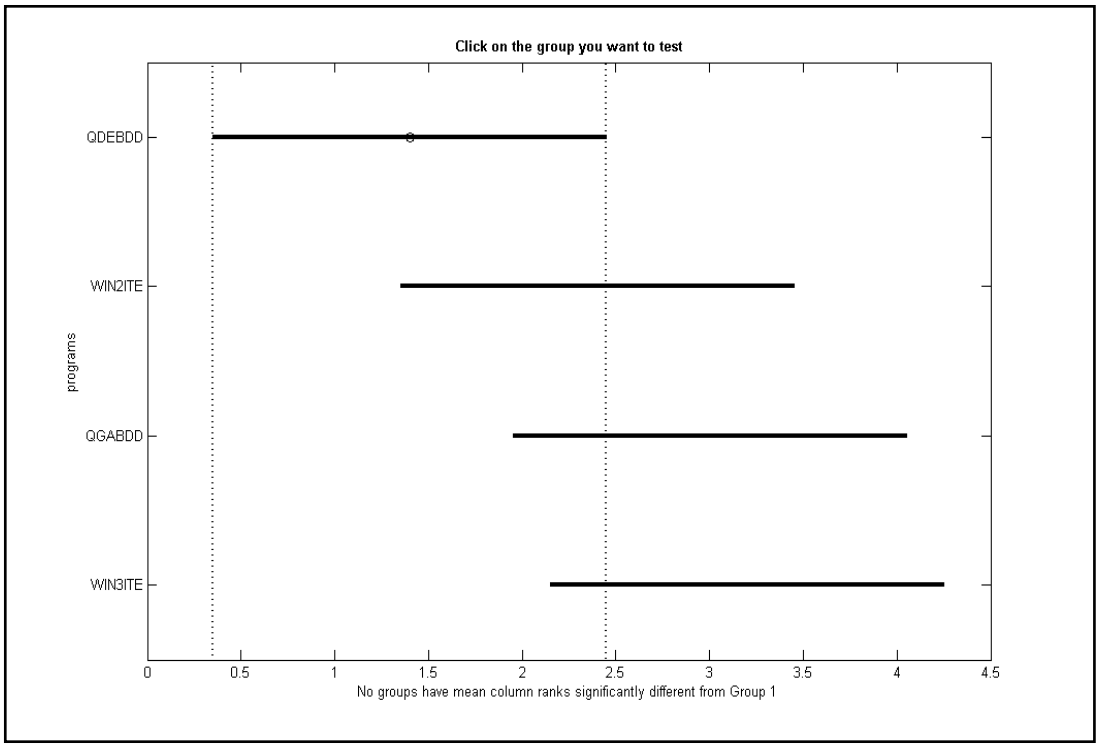


Figure 5.19 – Test de Friedman (a=0.01) pour l'ensemble de tests de 100 variables.

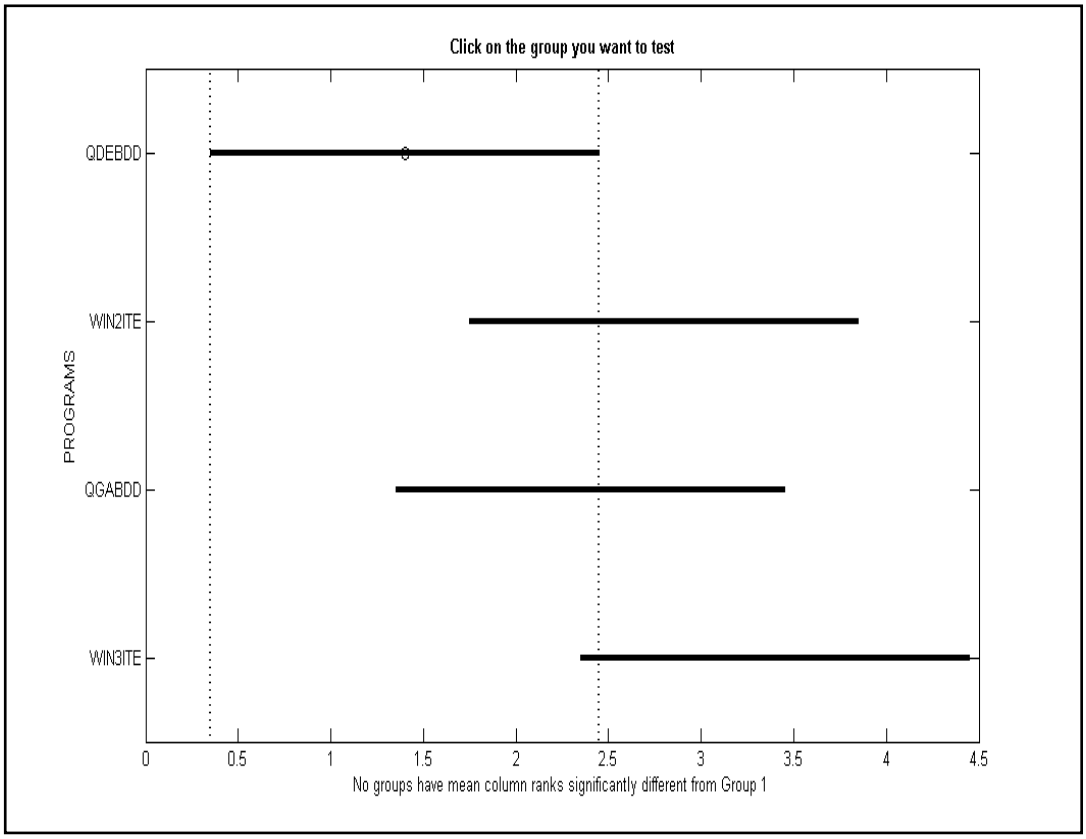


Figure 5.20 – Test de Friedman (a=0.01) pour l'ensemble de tests de 150 variables.

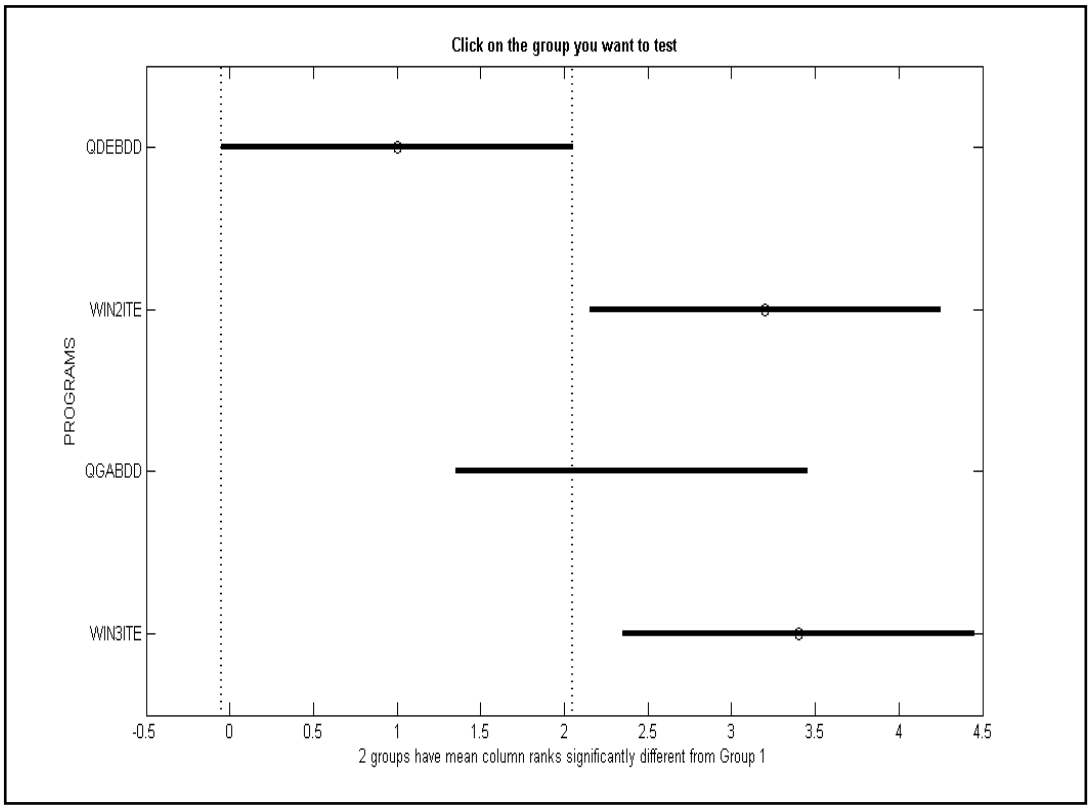


Figure 5.21. Test de Friedman (a=0.01) pour l'ensemble de tests de 200 variables.

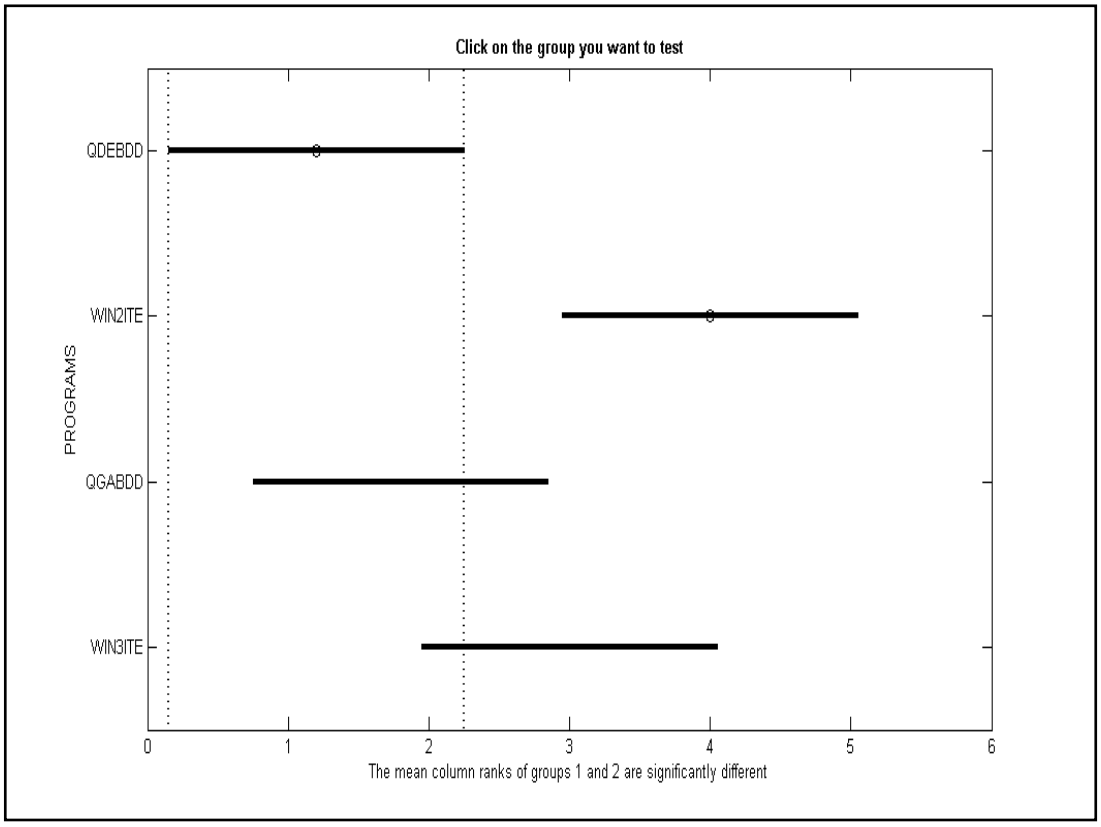


Figure 5.22 –Test de Friedman (a=0.01) pour l'ensemble de tests de 250 variables.

Concernant le comportement d'optimisation de l'approche QDEBDD, on constate que la recherche de la solution optimale n'est pas effectuée d'une manière aléatoire. Mais elle se déroule itérativement en améliorant progressivement la solution jusqu'à la solution optimale. La figure 5.23 montre la variation de la meilleure taille de BDD trouvée au cours du processus d'optimisation.

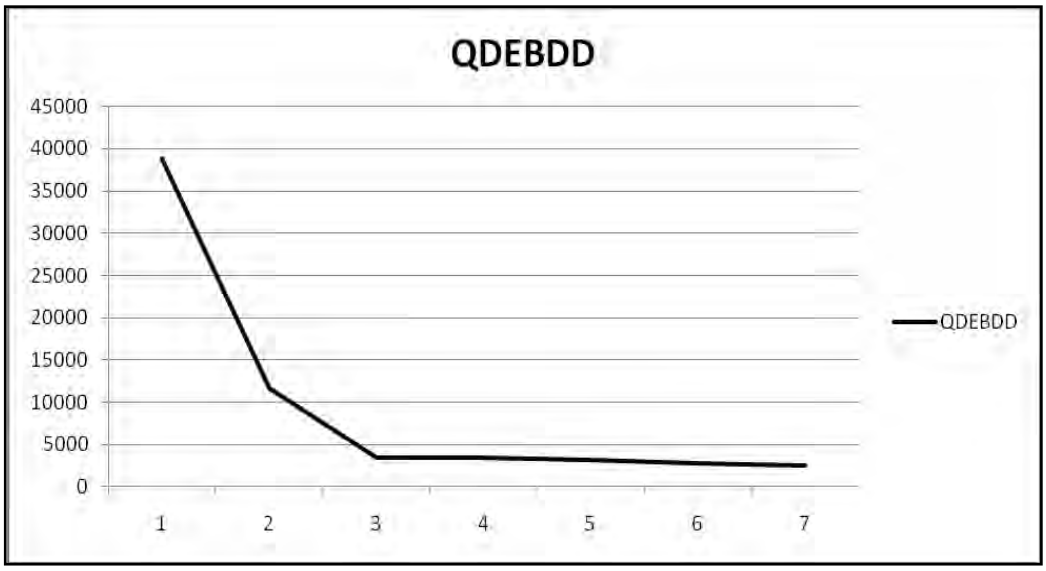


Figure 5.23 – La recherche progressive de la solution optimale pour 200 variables.

Pour voir graphiquement le BDD construit lors de la phase d'évaluation nous avons enregistré le BDD sous le format *dot,* et on a utilisé l'outil Graphviz qui permet de tracer un BDD. La figure 5.24 montre le BDD initial pour un test de 40 variables, après l'application de la méthode QDEBDD sur le même test on est arrivé à un ordre optimal des variables où la taille du BDD trouvé est considérablement réduite comme le montre la figure 5.25.

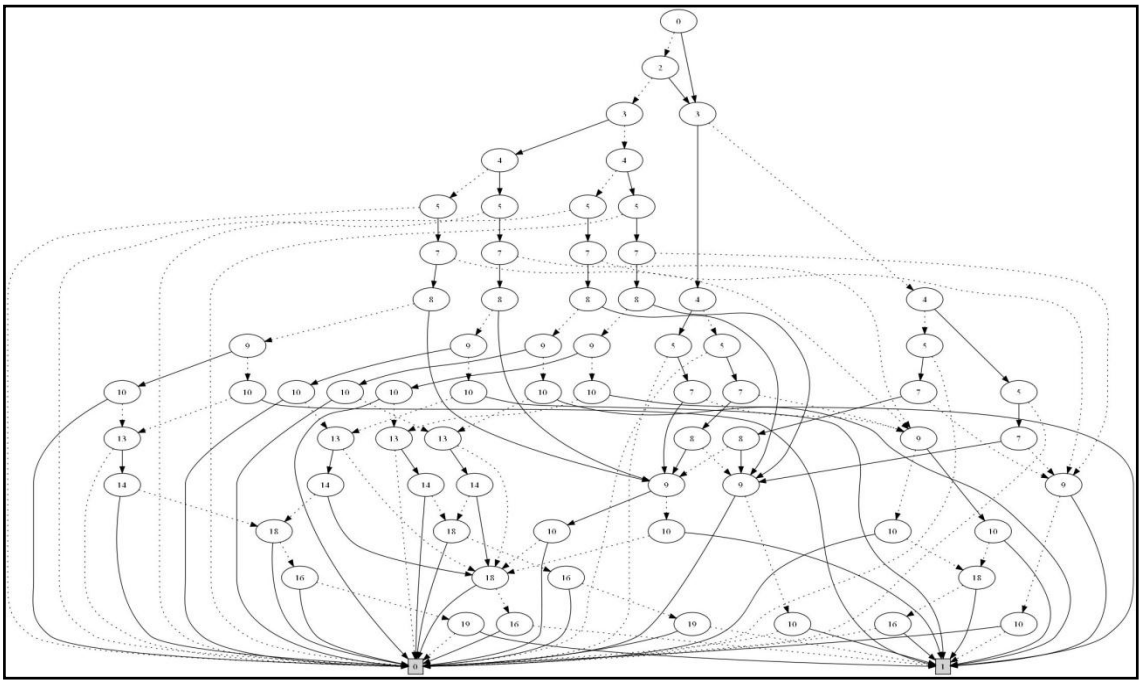


Figure 5.24 – Le BDD initial pour un test de 40 variables.

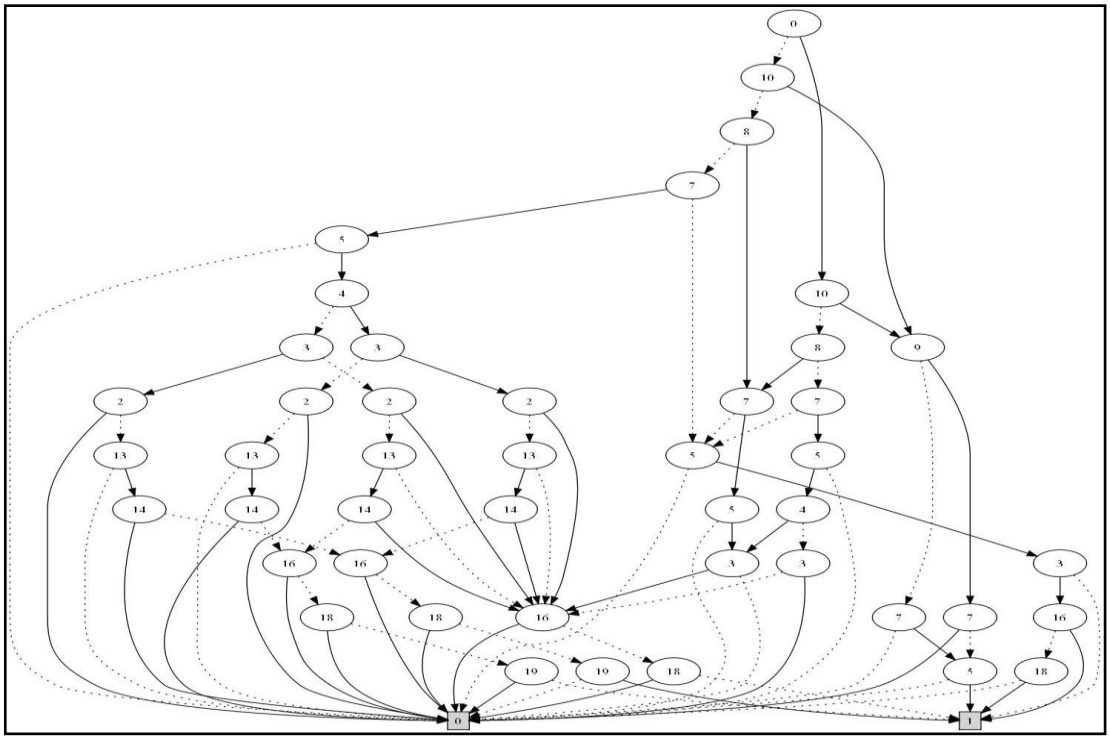


Figure 5.25 – Le BDD final pour un test de 40 variables.

5.7. Étude de l'effet de la solution initiale

Comme nous avons évoqué précédemment, le bon choix de la solution initiale contribue efficacement dans l'amélioration des performances de l'approche proposée. En effet, une population initiale avec de bonnes solutions initiales peut aider grandement à accélérer la convergence de l'algorithme et trouver de bons résultats. Néanmoins, il faut éviter la convergence trop rapide, parce qu'on risque de stagner dans un minimum local. Pour éviter ça, il faut que la population soit toujours diversifiée (contient de bons et de mauvais individus).

Le choix d'un ordre initial se fait habituellement par la définition d'heuristiques qui sont liées à la classe de fonctions à représenter. Il existe plusieurs heuristiques pour la construction d'un bon ordre. Ces heuristiques utilisent généralement des informations extraites à partir des fonctions booléennes comme par exemple la signification sémantique de certaines entrées de la fonction booléenne. Cependant, aucune de ces heuristiques n'est meilleure que l'autre : une heuristique peut donner de bons résultats pour une fonction mais de très mauvais pour une autre. En outre, Il existe des fonctions pour lesquelles il est impossible de trouver un ordre donnant un BDD de taille polynomiale [Bryant, 1988].

La création d'ordre initial que nous avons fait est basé sur l'heuristique générale suivante, qui consiste à essayer de rapprocher au maximum dans l'ordre choisi deux variables booléennes en relation l'une avec l'autre. Cette heuristique se justifie par les arguments suivants :

Considérons deux variables x_i et x_j avec i < j dans l'ordre des variables d'un BDD, et x_i et x_j sont en relation l'une avec l'autre (par exemple, elle vérifie $x_i = \neg x_j$). Dans ce cas, il faudra forcement introduire le nœud x_j sur tous les chemins du graphe de décision partant du nœud x_i. Le nombre de ces chemins va dépendre en partie du nombre de nœuds et donc du nombre de variables qui sont intercalées entre x_i et x_j. Si $j \Leftrightarrow i = 1$, il n'y aura que deux nœuds x_j (ce qui est le minimum). Sinon, si $j \Leftrightarrow i = d$, alors nous pouvons avoir jusqu'à 2^{d+1} nœuds x_j. La figure 5.26 montre l'effet de cette heuristique sur la taille du BDD résultant. Chacun des deux BDDs représente la fonction booléenne $(x1 \vee x3) \wedge (x2 \Rightarrow x4)$. Le BDD à droite est construit sur un ordre entrelacé des variables : x1<x3<x2<x4. Sa taille en nombre de nœuds est considérablement inférieure à celle du BDD à gauche qui utilise l'ordre: x1, x2, x3, x4.

Nous avons implémenté cette heuristique dans la méthode QDEBDD. D'après les résultats trouvés, cette heuristique a considérablement amélioré les performances de l'algorithme QDEBDD (figure 5.27). En effet, cette heuristique a permis d'accélérer la convergence de QDEBDD vers des solutions meilleures (table 5.3) que celles obtenues avec des solutions initiales aléatoires et pour un nombre inferieur de génération.

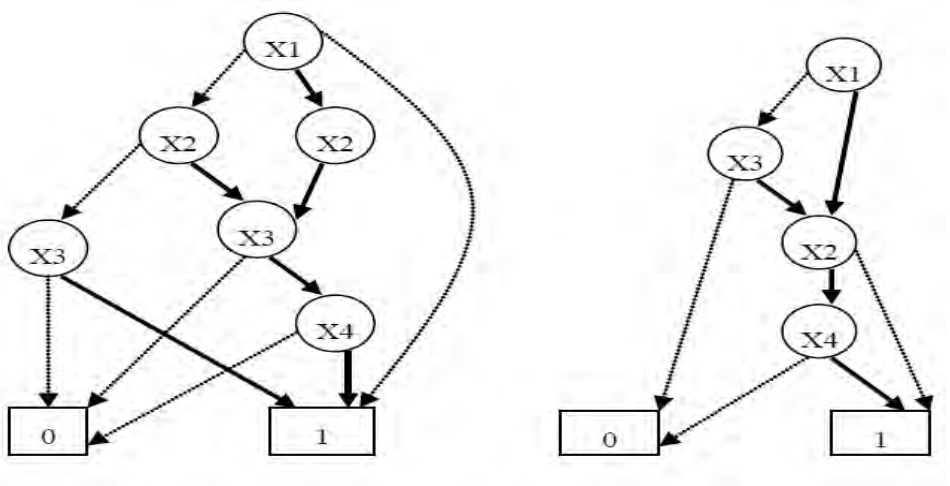


Figure 5.26 – L'effet de l'ordre entrelacé des variables.

Table 5.3 – Les résultats de l'impact de la solution initiale**.**

Test	Nombre de variables	QDEBDD	QDEBDD avec une bonne solution initiale
Test40_2	40	232	205
Test40_4	40	405	374
Test60_1	60	1016	376
Test60_2	60	739	534
Test60_3	60	427	312
Test80_4	80	566	392
Test80_5	80	904	886
Test100_2	100	2507	2011
Test100_5	100	6952	4497
Test150_2	150	6611	6278
Test150_4	150	1909	1909
Test150_5	150	2130	2007
Test200_1	200	8359	3546
Test200_2	200	7114	2775
Test200_4	200	4754	2180
Test200_5	200	112585	70718
Test250_1	250	447097	140076
Test250_2	250	1554	1224
Test250_5	250	54251	20759

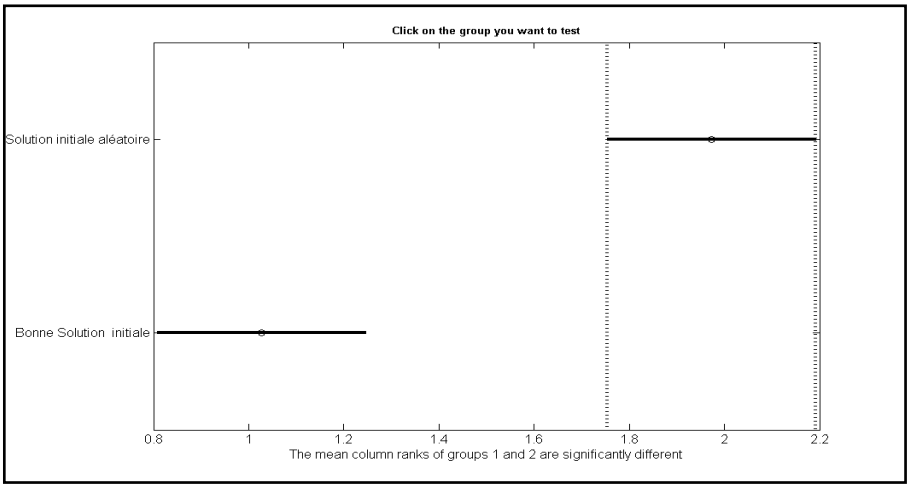


Figure 5.27 –Test de Friedman (a=0.05) : l'impact de la solution initiale.

5.8. Étude de l'effet de la recherche locale

Nous allons étudier dans cette section l'impact de la recherche locale sur les performances de l'algorithme QDEBDD. Pour cela, nous avons étudié deux méthodes de recherche locale. La première est une méthode simple de permutation de variable. L'idée de base consiste à changer l'ordre de deux variables en les permutants (figure 5.28). En appliquant cette technique sur plusieurs paires de variables, on arrive à choisir un ordre qui diminue la taille de l'OBDD (figure 5.29). Cette technique connue dans la littérature des diagrammes de décision binaire sous le nom du *SWAP*.

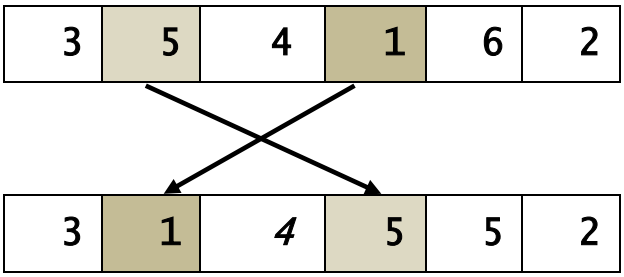


Figure 5.28 – Le mécanisme de la recherche locale *SWAP*.

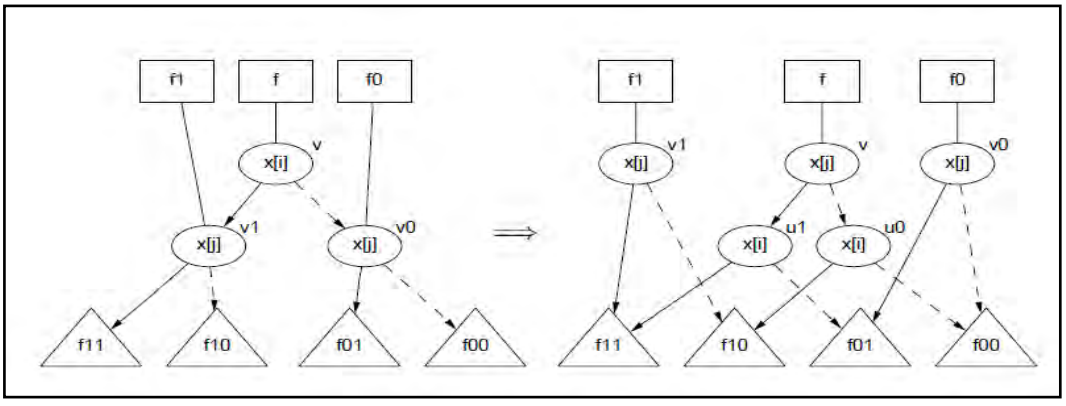


Figure 5.29 – L'effet de la recherche locale *SWAP*.

La deuxième heuristique utilisée consiste à chercher la meilleure position d'une variable sans changer l'emplacement des autres variables. Cet algorithme est connu sous le nom *Sifting*. Tout d'abord, l'algorithme évalue le coût du déplacement d'une variable le long de l'ordre et mémorise le bon emplacement qui réduit la taille du BDD. Dans la seconde étape, l'algorithme *Sifting* déplace la variable vers le meilleur rang trouvé. Ce déplacement peut être effectué soit de droite à gauche, ou bien l'inverse comme il est montré par la figure. Cette technique permet bien d'échapper aux minima locaux. En effet, cet algorithme donne généralement de bons résultats par rapport aux autres méthodes de recherche locale dans l'ordonnancement des variables d'un BDD. Néanmoins l'inconvénient majeur de cette méthode et la lenteur, parce qu'il effectue plusieurs évaluation de la fonction objectif.

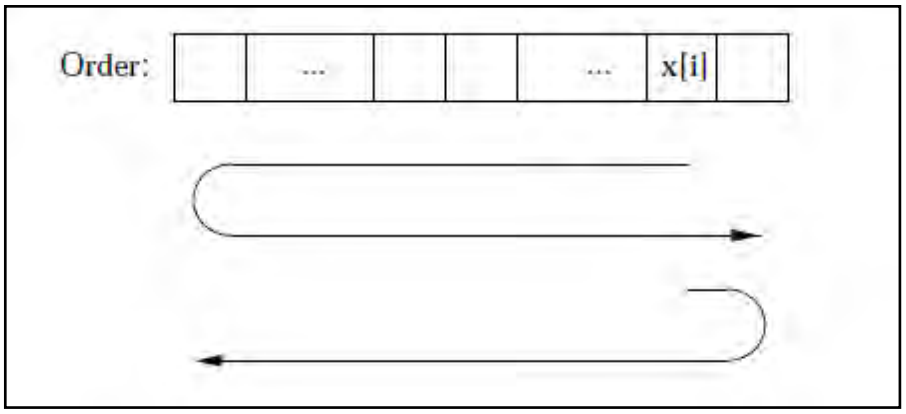


Figure 5.30 –L'idée du de l'algorithme Sifting.

Nous avons implémenté les deux algorithmes de recherche locale mentionnés en haut au sein de la méthode QDEBDD. Nous avons utilisé les mêmes conditions pour exécuter les trois programmes suivants : QDEBDD avec la procédure Permutation de fenêtres vu dans la section 5.5, QDEBDD avec la procédure Swap, et QDEBDD avec la procédure Sifting. D'après les résultats trouvés (table 5.4), la procédure Sifting est la méthode de recherche locale la plus performante. En effet, elle donne des résultats nettement supérieurs que les autres méthodes (figure 5.29). Malheureusement, on a constaté que QDEBDD devient très lent avec la procédure Sifting. D'un autre côté, QDEBDD avec la méthode de permutation de fenêtres donne des résultats moins bons que les autres programmes dans cette expérience ce qui confirme ses limites pour le traitement du problème d'ordonnancement de variables de BDD.

Pour tirer profit des caractéristiques des trois méthodes, nous pouvons intégrer les trois méthodes dans une seule méthode où l'exécution de chaque méthode est conditionnée avec une probabilité d'exécution. Nous pouvons également utiliser les techniques d'hybridations, de distribution et de parallélisation afin de construire un bon algorithme englobant les trois méthodes.

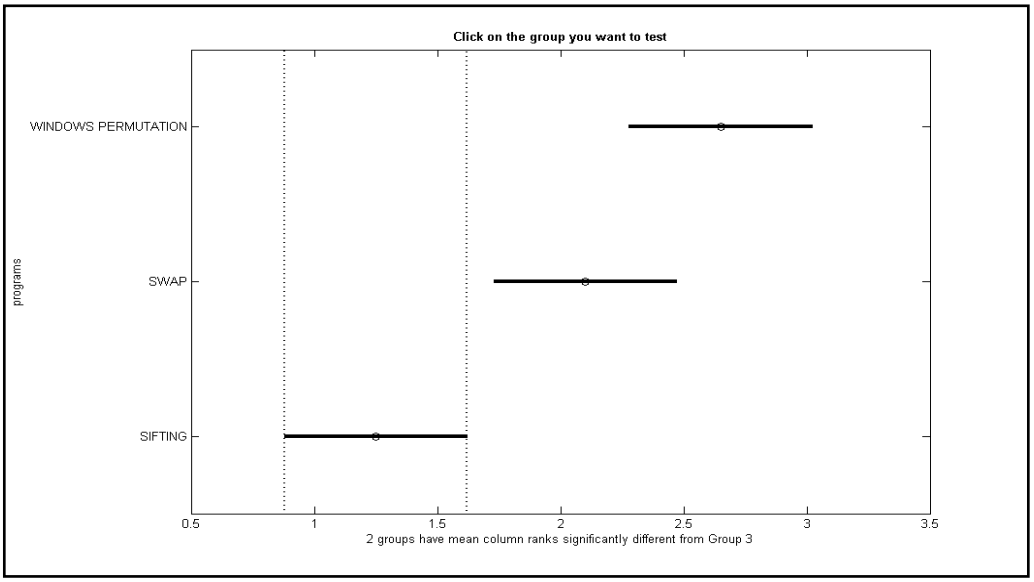


Figure 5.31 – Test de Friedman (a=0.05): comparaison entre QDEBDD basé *permutation de fenêtres (Windows-permutation),* QDEBDD basé Swap et QDEBDD basé *Sifting*.

Table 5.4. Les résultats de l'utilisation de différentes méthodes de recherche locale.

Test	# variables	Windows-permutation	SWAP	SIFTING
Test40_2	40	232	172	108
Test40_4	40	405	335	268
Test60_1	60	1016	874	437
Test60_5	60	873	781	883
Test80_4	80	359	453	320
Test80_5	80	904	713	579
Test100_1	100	6719	2540	1733
Test100_2	100	2507	1973	1407
Test100_3	100	483	435	399
Test100_5	100	6952	2451	2325
Test150_1	150	3301	3128	3385
Test150_2	150	6611	6394	4447
Test150_4	150	1909	1764	1290
Test200_1	200	8359	6059	4021
Test200_2	200	7114	4620	4574
Test200_4	200	4754	5348	4857
Test200_5	200	112585	124781	109760
Test250_1	250	447097	291626	259763
Test250_3	250	12311	13727	9107
Test250_5	250	54251	51535	48359

5.9. La complexité des approches QGABDD et QDEBDD

Maintenant, nous allons étudier la complexité algorithmique des approches développées. Pour trouver l'ordre optimal de *n* variables, cette complexité dépend de plusieurs facteurs. Concernant l'approche QGABDD, les étapes de l'approche les plus gourmandes en temps CPU sont les suivantes:

1. La complexité de l'évaluation d'un individu est égale à la complexité de construction d'un BDD plus la complexité de calculer le nombre de nœuds : c_Eval_BDD. Cette dernière est de l'ordre de O(n).
2. La complexité des opérations d'interférence, de mutation, de croisement, et de mesure est de l'ordre $O(n^2)$.
3. Donc, la complexité d'optimisation d'un individu pour un nombre d'itérations *m* est : $O(m*(n^2+\mathrm{n}))$.
4. Par conséquent, la complexité d'optimisation d'une population de *p* individus est: $complexité(QGABDD) = O(p*m*(n^2+\mathrm{n})) \approx O(p*m*n^2)$.

Où *n* : le nombre de variables, *p* : la taille de la population, *m* : le nombre d'itérations.

Pour le cas de l'approche QDEBDD, il faut ajouter à la complexité précédente la complexité de la recherche locale (*complexité_hill*) qui est de l'ordre de $O(n^2)$.

$$complexité(QDEBDD) = O(p*m*(n^2+\mathrm{n})) \approx O(2p*m*n^2)$$

Donc le temps CPU de nos résultats est dépendant de nombre de variables, la taille de la population et le nombre d'itérations au niveau de la recherche locale et globale.

Comme nous l'avons fait sous-entendre précédemment, nous nous intéressons quasi-exclusivement à la complexité en temps des algorithmes. Il est parfois intéressant de s'intéresser à d'autres de leurs caractéristiques, comme la complexité en espace (taille de l'espace mémoire utilisée). Pour notre problème le nombre de variables ainsi que le la taille de la formule (nombre de portes logiques) ont une grande influence sur les performances des approches proposées. Tous les résultats précédents sont obtenus sur une machine avec 1 GB de mémoire. Dans toutes les expériences dont le nombre de variables varie entre 40 et 250 le problème d'espace mémoire ne s'est pas posé, mais, on a rencontré ce problème pour un nombre de variables supérieur à 250 où l'exécution du programme a été interrompu parce que l'espace de la mémoire est insuffisant pour construire le BDD.

5.10. Conclusion

Les travaux réalisés durant cette partie de notre thèse s'articulent autour de la résolution du problème d'ordonnancement des variables des diagrammes de décision binaire. Pour résoudre ce problème, on a proposé deux approches basées sur un noyau quantique évolutionnaire. La première méthode développée nommée QGABDD est basée sur un algorithme génétique quantique. La représentation quantique des solutions permet le codage de tous les ordres potentiels avec une certaine probabilité. Le processus d'optimisation consiste en l'application d'une dynamique quantique constituée d'un ensemble d'opérations quantiques telles que l'interférence, la mutation quantique et la mesure. La taille de la population est considérablement réduite grâce au principe de la superposition. Bien que les études expérimentales soient encourageantes, QGABDD nécessite plus d'améliorations afin de trouver des solutions meilleures. Pour cela, une deuxième méthode plus robuste a été proposée. La deuxième approche nommée QDEBDD est basée sur une hybridation entre un algorithme quantique à évolution différentielle AQED et une méthode de recherche locale. La combinaison des deux paradigmes de résolution peut permettre de bénéficier des atouts respectifs de chacun d'entre eux pour construire des méthodes de résolution plus efficaces et plus robustes. Les capacités de diversification des AQEDs permettent de bien couvrir l'espace des solutions et de déterminer les zones prometteuses. Tandis que les capacités d'intensification des méthodes de recherche locales permettent d'approfondir la recherche dans chacune des zones prometteuses localisées. L'étude expérimentale a montré que QDEBDD est plus performant que QGABDD et d'autres programmes itératifs de la littérature. Cependant, le temps d'exécution reste encore élevé par rapport aux méthodes heuristiques. Ce problème peut être surmonté en implémentant notre approche sur des machines parallèles. Le parallélisme peut se faire au niveau de la recherche locale cela nous permet d'accélérer l'évaluation des individus et trouver les solutions dans un temps raisonnable.

CHAPITRE 6

Approches Hybrides pour le Problème de Satisfiabilité Propositionnelle

"Anything that can go wrong will go wrong".

Murphy's law

Dans ce chapitre, nous allons présenter nos contributions pour la résolution du problème de la satisfiabilité maximale MAX-SAT. Les approches développées sont basées sur des algorithmes hybrides entre plusieurs paradigmes.

Ce chapitre est subdivisé de la façon suivante : après une formulation mathématique du problème traité. Nous présentons ensuite le cadre de résolution commun des approches développés basé sur un noyau évolutionnaire quantique : QHILLSAT, QGASAT et QGADPLL. Ensuite, on présente la méthode ClonSat basée sur un Système Immunitaire Artificiel. Finalement, une conclusion et quelques perspectives sont données.

Une partie des travaux présentés dans ce chapitre ont fait l'objet de plusieurs publications.

Sommaire

6.1 Introduction

Comme on a mentionné dans le chapitre 4, le problème de la satisfiabilité maximale en logique propositionnelle (MAX-SAT) vise à trouver la meilleure affectation d'un ensemble de variables booléennes qui rende le maximum de clauses vraies dans une formule booléenne. Ce problème complexe a plusieurs applications dans divers domaines tels que les modèles de contrôle, les graphes colorés, la planification de tâches, etc. Une instance de ce problème est MAX 3-SAT, où le problème impose un nombre limite à 3 de variables logiques inclues dans chaque clause C de la formule booléenne sous forme FNC. La formule suivante est une formule Booléenne en 3-CNF :

f= (x1 OU x2 OU x3) ET (non (x1) OU x2 OUx3) ET (x1 OU non (x2) OU x3)

Malheureusement, il est démontré que le problème MAX-SAT est NP-complet si le nombre de variables par clause est supérieur à 3. Il est question donc de proposer de nouvelles approches pour appréhender ce problème. Pour cela, nous avons adopté une approche évolutionnaire hybride dont le développement a conduit à la proposition de quatre approches. Les travaux réalisés durant cette thèse apportent différentes contributions autour des problèmes de satisfiabilité.

Tout d'abord, nous avons adopté une approche quantique évolutionnaire dont le développement a conduit à la proposition de trois méthodes: QHILLSAT [Layeb et al, 2008e], QGASAT [Layeb et al, 2008c] et QGADPLL [Layeb et al, 2010a]. Ces dernières reposent sur un noyau quantique génétique de base défini par une représentation quantique appropriée et une dynamique quantique évolutionnaire adaptée similaire à celui présenté dans le chapitre précédent avec quelques différences liées à la nature du problème traité. Les deux méthodes QHILLSAT et QGASAT utilisent deux méthodes de recherche locale différentes afin de renforcer leurs capacités d'intensification. Cependant, la méthode QGADPLL est une hybridation entre un algorithme quantique génétique et la méthode de résolution exacte DPLL [Davis et al., 1962]. Les méthodes développées se distinguent des autres méthodes évolutionnaires par une taille de population réduite et un nombre raisonnable d'itérations pour trouver de bonnes solutions grâce aux principes de l'informatique quantique telle que la superposition d'états, l'interférence et la mutation. Les expériences menées sur un ensemble de tests ont montré l'efficacité des approches proposées et leurs capacités à réaliser des solutions de bonnes qualités. Les résultats trouvés sont comparables voir meilleures que d'autres méthodes populaires dans le domaine du MAX 3-SAT.

Indépendamment, des algorithmes inspirés du quantique, nous avons proposé une nouvelle approche nommée ClonSat [Layeb et al, 2010b] basée sur les systèmes immunitaires artificiels pour résoudre le problème sous-jacent. L'objectif de ce travail est de valider la faisabilité et l'efficacité des systèmes immunitaires artificiels pour traité un dur problème d'optimisation qui est le MAX-SAT. Les résultats obtenus sont encourageants et montre la

faisabilité de notre méthode. En outre, ce cadre de résolution constitue une plateforme pour la résolution d'autres problèmes de satisfiabilité booléenne.

Dans la suite de ce chapitre nous allons tout d'abord présenter un cadre de résolution de problème MAX SAT basé sur des algorithmes quantiques génétiques hybridés avec des méthodes de recherche locales. Le développement de ce cadre est basé sur une représentation quantique adéquate ainsi qu'une dynamique quantique génétique opérant sur cette représentation (Figure 6.1).

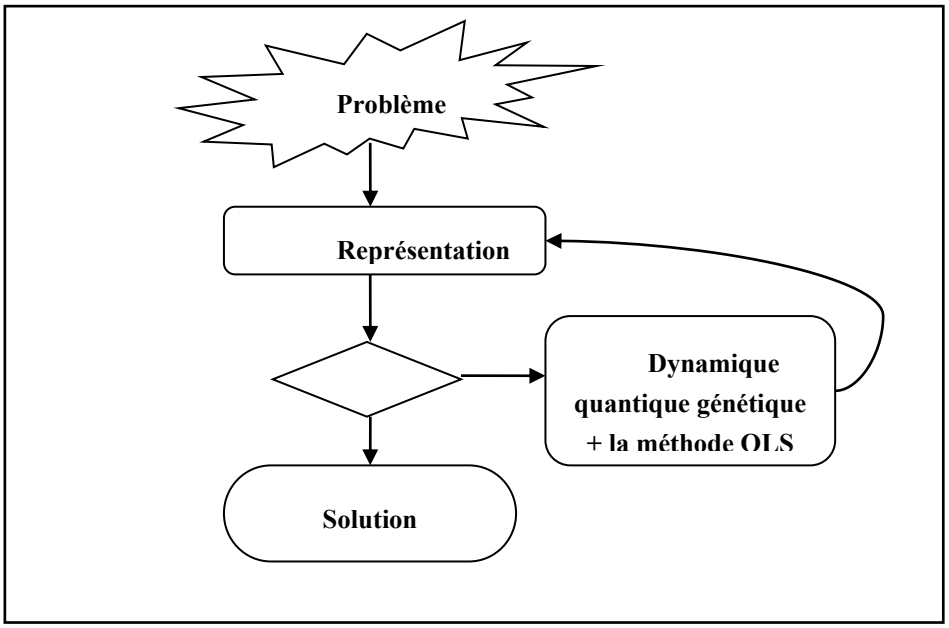


Figure 6.1 : Un cadre de résolution quantique génétique hybridé avec une méthode de recherche locale QLS pour le problème MAX3-SAT.

6.2 Formulation du problème

Nous donnons dans cette section les formulations mathématiques du problème sous forme d'un problème d'optimisation. Etant donné une formule booléenne ϕ exprimé en CNF (Forme Normale Conjonctive) et ayant ***n*** variables booléennes x_1, x_2... x_n, et ***m*** clauses. Le problème MAX-SAT peut être formulé comme suit [Layeb et al, 2008c]:

- L'assignation de ces variables est un vecteur $v = (v_1, v_2 \ldots v_n) \in \{0,1\}^n$.
- Une clause C_i de longueur k est une disjonction de k littéraux :
- $C_i = (x_1 \text{ OR } x_2 \text{ OR} \ldots \text{ OR } x_k)$.
- Chaque littéral est une variable ou une négation d'une variable.
- Chaque variable peut apparaître plusieurs fois dans l'expression.

MAX-SAT est le problème de l'affectation de variables qui satisfait le plus grand nombre possible de clauses. Il peut être défini par la spécification du couple (Ω, SC) où Ω est l'ensemble de toutes les solutions envisageables. SC est une fonction de $\Omega \rightarrow N$, appelée

score de l‟affectation, égal au nombre de clauses vraies. Par conséquent, le problème consiste à définir la meilleure affectation binaire qui maximise le nombre de clauses vraies de la formule booléenne (maximise SC). Evidemment, il ya 2^n solutions possibles qui répondent à ce problème. Par conséquent, il est impossible d'obtenir des solutions exactes en temps polynomial. De ce fait, il a été prouvé que le problème MAX k-SAT est NP-complet pour tout k≥ 3. Il s'agit bien d'un problème d'optimisation combinatoire. Dans une telle situation, les métaheuristiques constituent une bonne alternative pour résoudre ce problème. En effet, l'exploitation des caractéristiques des métaheuristiques pour résoudre le problème MAX-SAT est une idée prometteuse.

6.3 Un cadre de résolution quantique génétique pour MAX 3-SAT

La définition de ce cadre repose fondamentalement sur une représentation quantique de l'espace de recherche associé au problème traité ainsi qu‟une dynamique utilisée pour explorer l'espace de recherche en opérant sur la représentation quantique moyennant des opérations quantiques de base.

6.3.1 La représentation quantique du problème MAX 3-SAT

Dans tous ce qui est suit, N représente le nombre de variables booléennes et M représente le nombre de clauses dans la formule ϕ. Afin d‟appliquer facilement les principes de la quantique sur le problème MAX 3-SAT, une représentation quantique est définit. L‟assignation booléenne est représentée par un vecteur binaire de taille N. En termes d'informatique quantique, chaque affectation de variables est représentée par un registre quantique de taille N (figure 6.2). Le registre contient une superposition de toutes les solutions potentielles. Chaque colonne $\begin{pmatrix} a_i \\ b_i \end{pmatrix}$ représente un seul qubit est correspond à la valeur binaire 0 ou 1. Les a_i et b_i sont des valeurs réelles satisfaisant la relation quantique $|a_i|^2 + |b_i|^2 = 1$. Pour chaque qubit une valeur binaire est calculée selon les probabilités $|a_i|^2$ et $|b_i|^2$(si $|b_i|^2 \geq 0$ alors le qubit vaut un 0 sinon il vaut un 1). Donc, les $|a_i|^2$ et $|b_i|^2$ sont interprétés comme probabilités pour prendre respectivement 0 ou 1. Par conséquent, toutes les solutions possibles peuvent être représentées par un vecteur quantique QV qui contient la superposition de toutes les solutions possibles. Ce vecteur quantique peut être considéré comme une représentation probabiliste de l'ensemble de toutes les solutions de MAX 3-SAT. Cette représentation est efficace quand elle est intégrée dans une approche évolutionnaire car elle contribue à la réduction de la taille de la population des chromosomes. Lorsqu‟on parle de la génétique, le vecteur quantique QV joue le rôle d‟un chromosome. De ce fait, un seul chromosome peut représenter l'ensemble de la population avec une certaine probabilité [Layeb et al, 2008e].

$$\begin{pmatrix} \mathbf{a_1} & \mathbf{a_2} & \cdots & \mathbf{a_m} \\ \mathbf{b_1} & \mathbf{b_2} & \cdots & \mathbf{b_m} \end{pmatrix}$$

Figure 6.2 – Représentation quantiques des solutions de MAX3-SAT.

6.3.2 Les opérations quantiques de base

Le moteur d'optimisation des approches développées est constitué d'un ensemble d'opérations quantiques qui sont à la base de la dynamique quantique génétique proposées. Nous avons utilisés des opérations qui s'appliquent sur un seul chromosome comme la mutation, des opérations qui s'appliquent sur deux chromosomes comme le croisement et des opérations qui s'appliquent sur une population comme la sélection.

6.3.2.1 L'opération de mesure

Cette opération transforme par projection du vecteur quantique en un vecteur binaire. Donc, on aura une solution parmi toutes les solutions présentées dans la superposition. Dans le cadre du problème MAX 3-SAT, ce vecteur représente une interprétation potentielle pour une formule donnée. La valeur d'un qubit est calculée suivant ses probabilités $|a_i|^2$, $|b_i|^2$ et pour chaque résultat de mesure, l'évaluation des solutions se fait par le nombre de clauses satisfaites dans la formule booléenne. La figure suivante montre un d'exemple de l'application de l'opération de mesure.

$$\begin{pmatrix} \mathbf{a_1} & \mathbf{a_2} & \cdots & \mathbf{a}_n \\ \mathbf{b_1} & \mathbf{b_2} & \cdots & \mathbf{b}_m \end{pmatrix} \xrightarrow{Mésure} (0,1,\ldots,1)$$

Figure 6.3 – L'opération de mesure.

6.3.2.2 Sélection quantique

Cette opération permet de choisir p chromosomes quantiques parmi n représentant la population. Elle se déroule en deux étapes : premièrement une mesure est effectuée pour extraire des chromosomes binaires. Ensuite, un choix de p chromosomes parmi n chromosomes est effectué en se basant sur l'évaluation des chromosomes extraits. Il est à noter que nous pouvons également adapter les différentes méthodes de sélection connues dans la littérature des algorithmes génétiques pour le cas des algorithmes quantiques génétiques.

6.3.2.3 L'interférence quantique

Le principe de cette opération est semblable à celle utilisée dans les méthodes proposées pour résoudre le problème d'ordonnancement des variables d'un BDD vues dans le chapitre précédent.

6.3.2.4 La mutation quantique

Cette opération permet l'exploration de nouvelles solutions et l'augmentation des capacités de diversification du processus de recherche. Elle consiste à l'application d'une perturbation sur un ensemble de qubits choisis aléatoirement des chromosomes quantiques. Dans le cas du problème MAX SAT, on peut utiliser deux types de mutations quantiques :

- ***Mutation inter-qubit :*** Elle consiste à choisir aléatoirement une paire de qubits selon une probabilité définie, puis on fait une permutation entre les qubit comme il est montré sur la figure suivante :

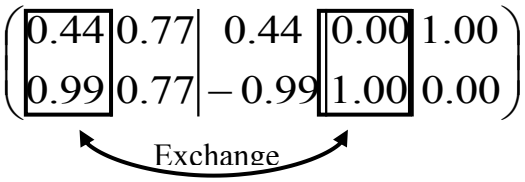


Figure 6.4 – L'opération de mutation quantique (mutation inter-qubit).

- ***Mutation intra-qubit :*** Elle consiste à choisir aléatoirement un qubit selon une probabilité définie, puis on fait une permutation entre ses composants a_i et b_i comme il est montré sur la figure suivante :

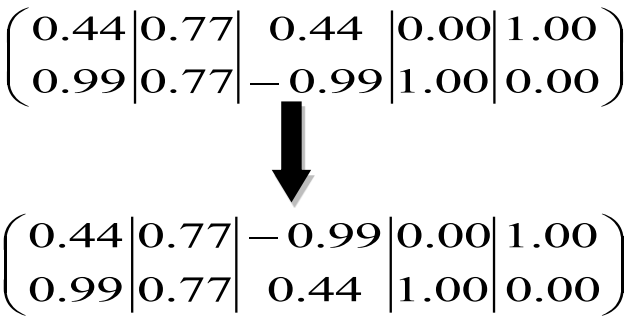


Figure 6.5 – L'opération de mutation quantique (mutation intra-qubit).

6.3.2.5 Le croisement quantique

Les croisements sont importants pour la promotion de l'échange de blocs de haute qualité au sein de la population. Ils échangent des blocs quantiques de deux chromosomes. Par exemple, la figure 6.6 montre un croisement quantique. Nous avons utilisé un opérateur de croisement général quoiqu'il existe dans la littérature des opérateurs de croisement bien spécifiques au problème MAX-SAT [Lardeux, 2005]. Pour plus d'efficacité le croisement doit rendre compte le plus possible de la sémantique des individus. En effet, dans les problèmes de satisfiabilité, les clauses d'une instance génèrent une structure de contraintes entre les variables. Afin de définir des croisements efficaces, on peut par exemple utiliser ces

contraintes structurelles. Plus spécifiquement, en se basant sur la satisfiabilité de chaque clause par rapport aux deux parents, il est judicieux de créer un fils qui bénéficie des deux parents en rendant le plus de clauses vraies possible [Lardeux, 2005]. Dans les futures versions de nos approches, nous allons étudier l'apport de ce type de croisement.

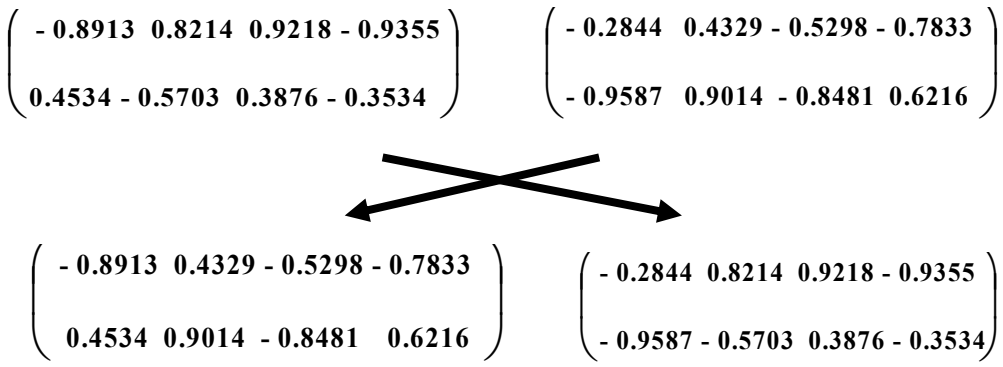


Figure 6.6 – Le croisement quantique.

6.3.3 La recherche locale

Afin d'améliorer l'efficacité du processus d'exploration, nous avons intégré des méthodes de recherche locale dans la dynamique quantique. Cette forme d'hybridation s'est avérée avantageuse dans le cadre des problèmes SAT. En effet, la capacité d'intensification de la méthode de recherche locale permet d'approfondir l'exploration de certaines régions de l'espace des solutions, identifiées comme particulièrement prometteuses. Cependant, la capacité de diversification de l'algorithme quantique génétique permet la réorientation périodique de la recherche d'un optimum vers des régions trop rarement visitées jusqu'ici. Nous avons constaté que les algorithmes proposés n'ont aucune efficacité sans l'utilisation de la recherche locale parce que le problème MAX 3-SAT nécessite plus d'intensification que de diversification.

6.3.3.1 La recherche locale Hill-Climbing quantique pour MAX 3-SAT

Durant les premières étapes de notre travail, notre objectif a été de démontrer la faisabilité des algorithmes génétiques quantiques pour résoudre le problème de satisfiabilité booléenne maximale. Il était donc une question de savoir comment exploiter le cadre de résolution quantique défini auparavant à travers une dynamique évolutionnaire pour résoudre le problème MAX 3-SAT. La réponse à cette question nous a conduit à définir une première version quantique évolutionnaire pour résoudre le problème MAX 3-SAT. L'approche QHILLSAT [Layeb et al, 2008e] a été proposée à cet effet est basée sur un algorithme génétique quantique hybridé avec une méthode de recherche locale simple qui est la méthode de la descente ou Hill-Climbing [Aarts et al, 1997].

Afin de pouvoir intégrer le Hill-Climbing dans le noyau quantique, nous avons défini une version quantique de l'algorithme Hill-Climbing spécifique pour le problème MAX 3-SAT (algorithme 6.1). Le choix du voisinage est très important dans ce type de méthode de recherche locale, il doit être un compromis entre efficacité et qualité. En effet, si les voisins sont très nombreux, on a de fortes chances de trouver l'optimum global mais visiter un voisinage peut être long parce qu'on visitera une grande partie de l'espace des solutions. Si le voisinage est très restreint, on risque fort de rester bloqué dans un optimum local de "mauvaise qualité". Dans notre approche, les déplacements dans l'espace de recherche sont effectués en utilisant un système de voisinage variable dont l'idée de base est de changer dynamiquement la taille de voisinage pour sortir d'un optimum local (figure 6.7). Ceci à contribuer énormément dans l'efficacité de notre algorithme. Dans nos expériences, la taille voisinage de la méthode de la descente est choisie entre 10 et 25.

Algorithme 6.1 : la recherche locale Hill-Climbing pour le problème MAX 3-SAT :
Données: un vecteur de qubit VQ, une formule ϕ en CNF
Résultat: un vecteur de qubit VQ

```
Début
  NbrGénération = nbr ;
  Générer une solution initiale VQ;
  Evaluer VQ;
  Répéter
    Générer aléatoirement un voisinage de VQ;
    Soit Vn la meilleure solution voisine ;
    Si Vn est meilleur que VQ Alors
      VQ= Vn;
      Si (Vn satisfait la formule ϕ) Alors Aller à fin ;
    Sinon
      NbrGénération--;
    finsi
  Jusqu'à (nbrGénération=0) ;
Fin
```

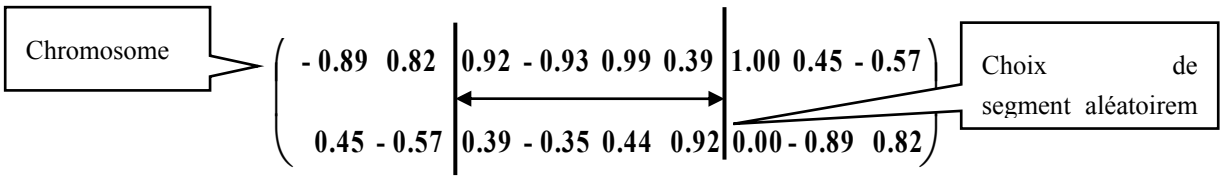


Figure 6.7 – La recherche locale Hill-Climbing quantique.

6.3.3.2 La recherche locale Flip Quantique pour MAX 3-SAT

Suite à l'étude de comportement de l'algorithme QHILLSAT, nous avons constaté que le point faible de cette méthode est sa procédure de recherche locale. En effet, on a constaté que QHILLSAT est lent et parfois il est piégé dans des minima locaux. Cette lenteur est due essentiellement à la méthode de la descente. En effet, cette méthode a l'inconvénient de "rester bloquée" dans un optimum local : une fois un optimum local trouvé, on s'arrête, même si ce n'est pas l'optimum global. Pour remédier à ce problème, nous avons opté pour une deuxième méthode de recherche locale plus performante. La deuxième approche nommée QGASAT [Layeb et al, 2008c] utilise une heuristique QFLIP qui consiste à flipper les qubits représentant les variables de la formule booléenne (algorithme. 6.2), le flip est accepté s'il y a une amélioration dans le nombre de clauses satisfaites. Le processus est répété jusqu'à ce qu'il n'y ait aucune amélioration.

Algorithme 6.2 : la recherche locale Quantum flip pour MAX SAT :

Données: un vecteur de qubit VQ, une formule Φ en CNF

Résultat: un vecteur de qubit VQ

```
début
    improve:=1;
    tant que (improve>0) faire
        improve:=0;
        pour j:=1 jusqu'à nVar faire
            flip le j^ème qubit de VQ;
            calculer le gain après le flip;
            si (gain>=0) alors
                accepter le flip ;
                improve:=improve+gain;
            fin si
        fin pour
    fin tant que
Fin
```

Enfin, il est à noter que notre approche est davantage générique et flexible. En effet, nous pouvons facilement intégrer d'autres méthodes de recherche locale stochastique plus robustes comme le recherche tabou, ou bien intégrer d'autres heuristiques liées au problème lui-même comme celles utilisés dans les différentes versions de l'algorithme WalkSAT.

6.4 Structure et fonctionnement des méthodes QGAGSAT et QHILLSAT

Les méthodes QGASAT et QHILLSAT repose sur un noyau de base défini par une représentation quantique appropriée et une dynamique génétique quantique hybride. Le processus d'optimisation consiste en l'application d'une dynamique quantique constituée d'un ensemble d'opérations quantiques tel que l'interférence, la mutation quantique, la mesure, etc. Cette dynamique quantique est renforcée par des mécanismes de recherche locale afin de donner des résultats plus performants. Le schéma général ce noyau est décrit par la figure 6.8.

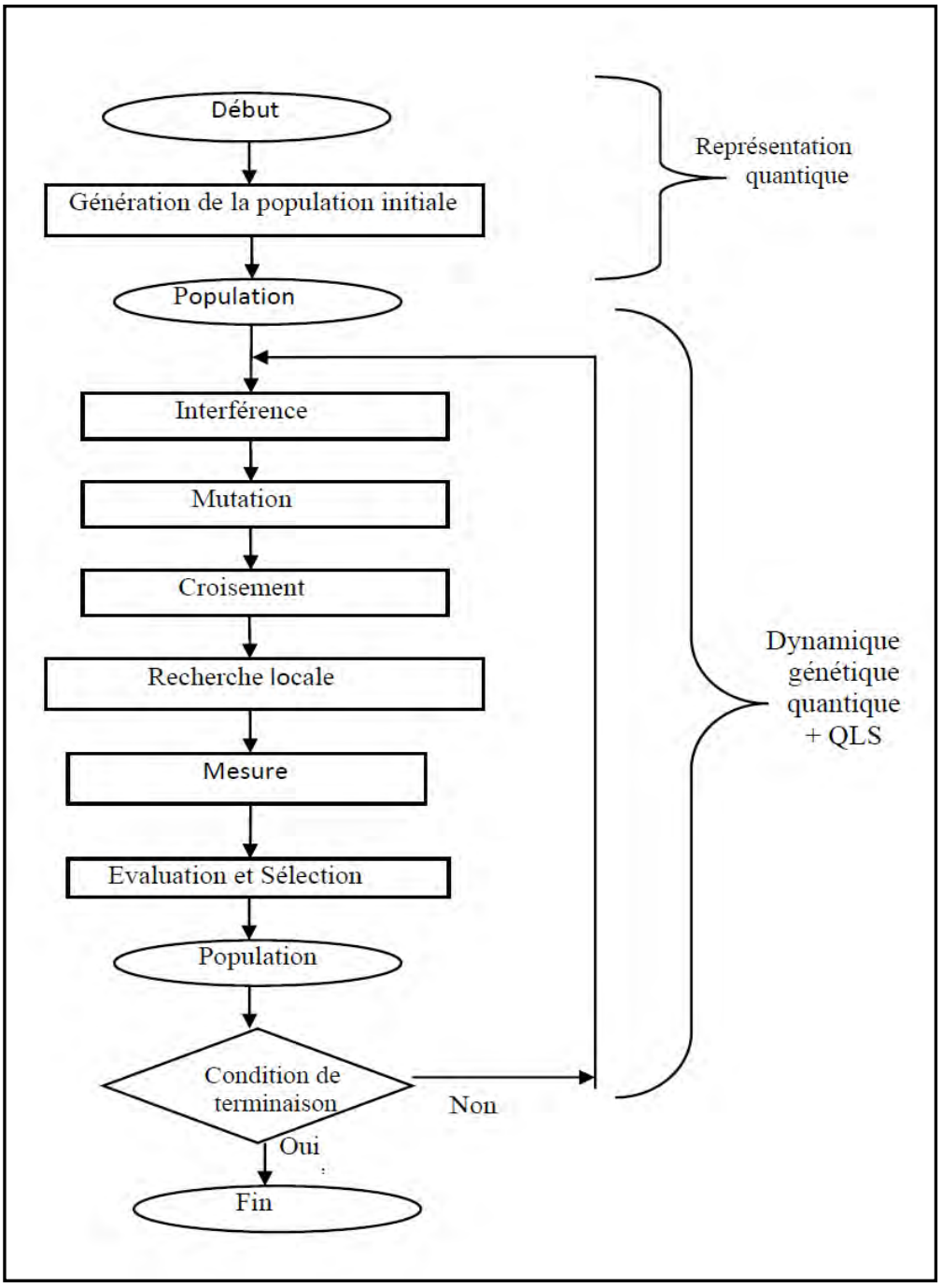


Figure 6.8 – La structure générale de QGAGSAT.

Expliquant maintenant le fonctionnement global de ces approches. Chaque méthode démarre d'une population initiale qu'on essaie ensuite d'améliorer à travers une série de raffinements par rapport à une fonction coût (Fitness). Premièrement, on génère une population initiale. Le choix de la population initiale d'individus conditionne fortement la rapidité de l'algorithme. Une connaissance de solutions de bonne qualité comme point d'initialisation permet à l'algorithme de converger plus rapidement vers l'optimum ou du moins s'y rapprocher. Nous pouvons utiliser, à titre d'exemple, une procédure glouton afin construire une population de bonne qualité. Néanmoins, dans notre cas on a créé la population initiale de façon aléatoire, parce qu'on veut mesurer les capacités d'optimisation de nos approches en démarrant avec des solutions complètement aléatoires. Tous les chromosomes quantiques sont donc initialisés avec des qubits équiprobables $(\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}})$ (figure 6.9).

$$\left(\begin{array}{c|c|c|c|c} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{array}\right)$$

Figure 6.9 – Un chromosome initial avec 5 qubits

Ensuite, on applique itérativement des raffinements sur cette population initiale afin d'améliorer sa qualité (Figure 6.8). Pour l'atteinte de notre objectif, on a procédé comme suit. Après l'étape d'initialisation, qui s'agit d'un codage quantique des solutions, le processus consiste en l'application itérative des transformations quantiques. Une transformation est constituée d'un ensemble d'opérations quantiques unaires et binaires. Les transformations unaires qui s'appliquent sur un seul chromosome tel que l'interférence, la mutation quantique simple, et la recherche locale sont incorporées dans une seule transformation avec certaines probabilités pour chaque opérateur. Le deuxième type de transformation est le croisement quantique qui est une transformation binaire.

Ensuite, on fait l'évaluation de la population courante et on procède à la sélection des individus qui vont constituer la population de la nouvelle génération (Figure 6.8). Pour évaluer les individus quantiques, on doit appliquer d'abord l'opération de mesure afin de trouver des solutions binaires faciles à évaluer. Le score de chaque solution est la somme des clauses vraies dans une formule propositionnelle en utilisant l'assignement des variables booléennes codées dans chaque solution. Dans nos approche, on a utilisé la sélection par tournoi par ce qu'elle est robuste et donne de bons résultats.

Le processus est répété jusqu'à la satisfaction des critères d'arrêt. En plus de la taille de la population, le nombre de génération est l'un des paramètres les plus importants et qui est

déterminant sur la qualité et la rapidité de l'algorithme. Dans notre cas le critère d'arrêt est composé de deux critères :

- *Le nombre de génération global :* on arrête l'algorithme après n générations
- *le nombre de générations sans amélioration :* L'algorithme peut aussi être arrêté lorsque la population n'évolue plus pendant m générations.

6.5 Implémentation et évaluation

Dans cette section, nous décrivons, dans un premier temps, le contexte expérimental de l'algorithme quantique génétique implémenté dans la plateforme *ParadisEO* [Cahon *et al.,* 2004]. Ensuite, dans la section suivante, nous présentons l'étude expérimentale (les instances utilisées, comparaison avec d'autres algorithmes, etc.) que nous avons menée pour mesurer les performances de ces approches. Nous terminons cette section par la présentation des résultats obtenus de la simulation, que nous avons effectuée sur ces algorithmes, et par la comparaison de ces derniers à ceux d'autres algorithmes.

6.5.1 Implémentation

Nous avons utilisé la plateforme ParadisEO [Cahon et al., 2004] pour implémenter les algorithmes que nous avons conçus pour la résolution du problème MAX 3-SAT. ParadisEO est un cadre de travail mettant en œuvre des librairies ANSI-C++ pour la conception et l'élaboration de métaheuristiques parallèles et hybrides dédiées à la résolution mono et multi-objectifs des problèmes d'optimisation combinatoire. L'usage d'une telle plateforme permet de développer des algorithmes flexibles pouvant facilement utiliser différents mécanismes fournis par cette plateforme. ParadisEO est caractérisé principalement par:

- Une librairie Open Source C++ (STL-Template).
- Indépendante de tout paradigme.
- Flexible / problème traité.
- Composants génériques (opérateurs de recherche, sélection, remplacement, …).

La structure générale de ParadisEO est montrée sur la figure 6.10. ParadisEO est formé de quatre librairies:

- ***Evolving Object (EO)* :** Cette librairie est à la base de tout le projet. Elle fournit des outils pour élaborer des métaheuristiques basées sur des populations (Algorithme génétique, Programmation génétique, etc.).
- ***Moving Object (MO) :*** La librairie MO permet d'implémenter des métaheuristiques à base de solution telles que la recherche avec tabous (RT), le recuit simulé (RS) ou encore le hill climbing

- ***Multi Objective Evolving Object (MOEO) :*** Cette librairie permet d'implémenter des métaheuristiques multi-objectifs avec plusieurs modèles tels que MOGA, NSGA-II, SPEA2, etc. Elle permet aussi d'effectuer des calculs de performances des métaheuristiques.
- ***Parallel Evolving Object (PEO) :*** Cette librairie fournit des outils pour mettre en œuvre des métaheuristiques parallèles et distribuées. Elle a été créée à la base pour assurer la parallélisation de la librairie EO de manière transparente et robuste. Nous pouvons trouver notamment le modèle en île, le modèle à parallélisation d'évaluation, le modèle cellulaire, etc.

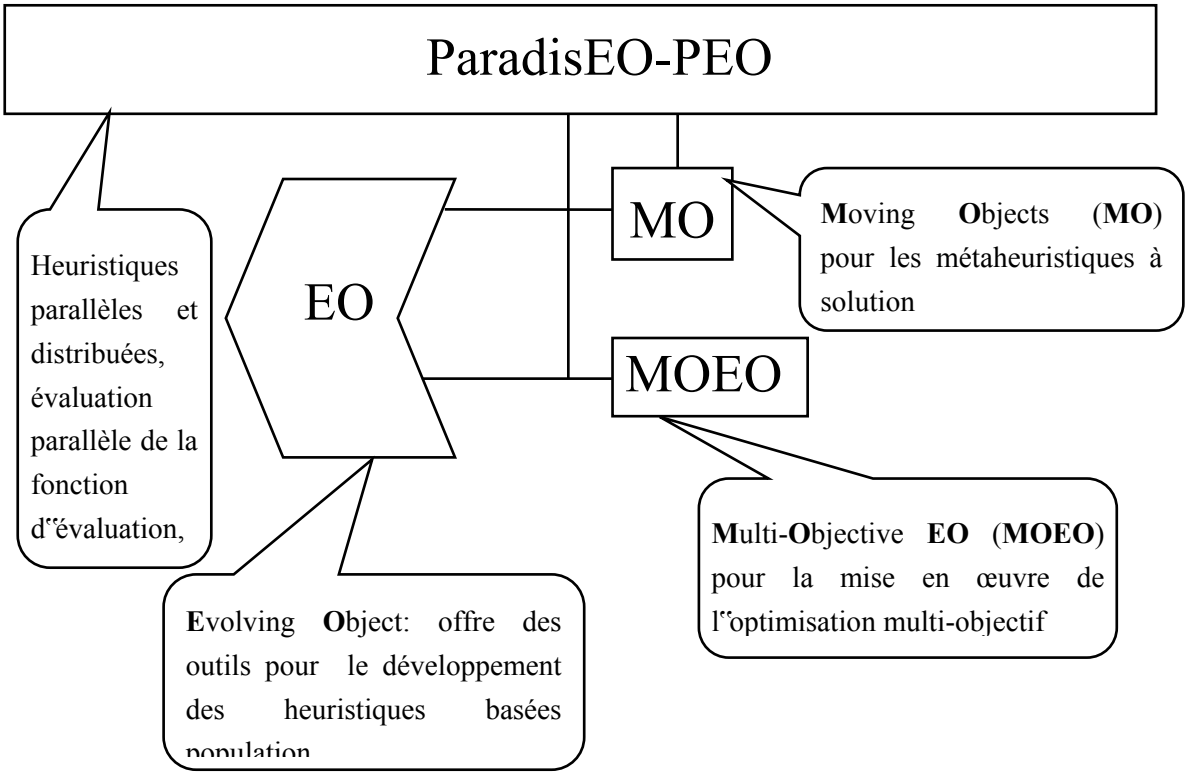


Figure 6.10 – La structure générale de ParadisEO.

Pour les besoins de notre travail, nous n'avons utilisé que la bibliothèque EO. En outre, Nous avons implémenté nos approches sur la même machine (1 GB de RAM et 3 GHZ) et sous un environnement Windows XP avec Visual C++ 2008.

Avant de pouvoir utiliser ParadisEO pour traiter notre problème, il était question d'intégrer le cadre de résolution quantique au sein de ParadisEO. Pour cela, nous avons conçu un environnement d'optimisation quantique au sein de ParadisEO. Cette contribution consiste en :

- Définir d'une nouvelle entité le **"eoQbit"** afin de pouvoir créer des individus avec un codage quantique.

- Définir un nouvel opérateur de mutation nommé **"eoQmutation"**. Cet opérateur unaire est une redéfinition de l'opérateur prédéfini de mutation eoMutation de ParadisEO qui s'applique sur un chromosome quantique.
- Définir un nouvel opérateur de croisement nommé **"eoQXover"**. Cet opérateur binaire est une redéfinition de l'opérateur prédéfini de croisement eoUBitXover de ParadisEO qui s'applique sur des chromosomes quantiques.
- Définir un nouvel opérateur de transformation nommé **eoQInterference**, qui implémente l'opération d'interférence.
- Définir une nouvelle transformation **(eoSGATransform)** qui combine tous les opérateurs définis précédemment.

Le diagramme suivant (figure 6.11) montre l'architecture interne du cadre de résolution quantique génétique développé pour le problème MAX SAT.

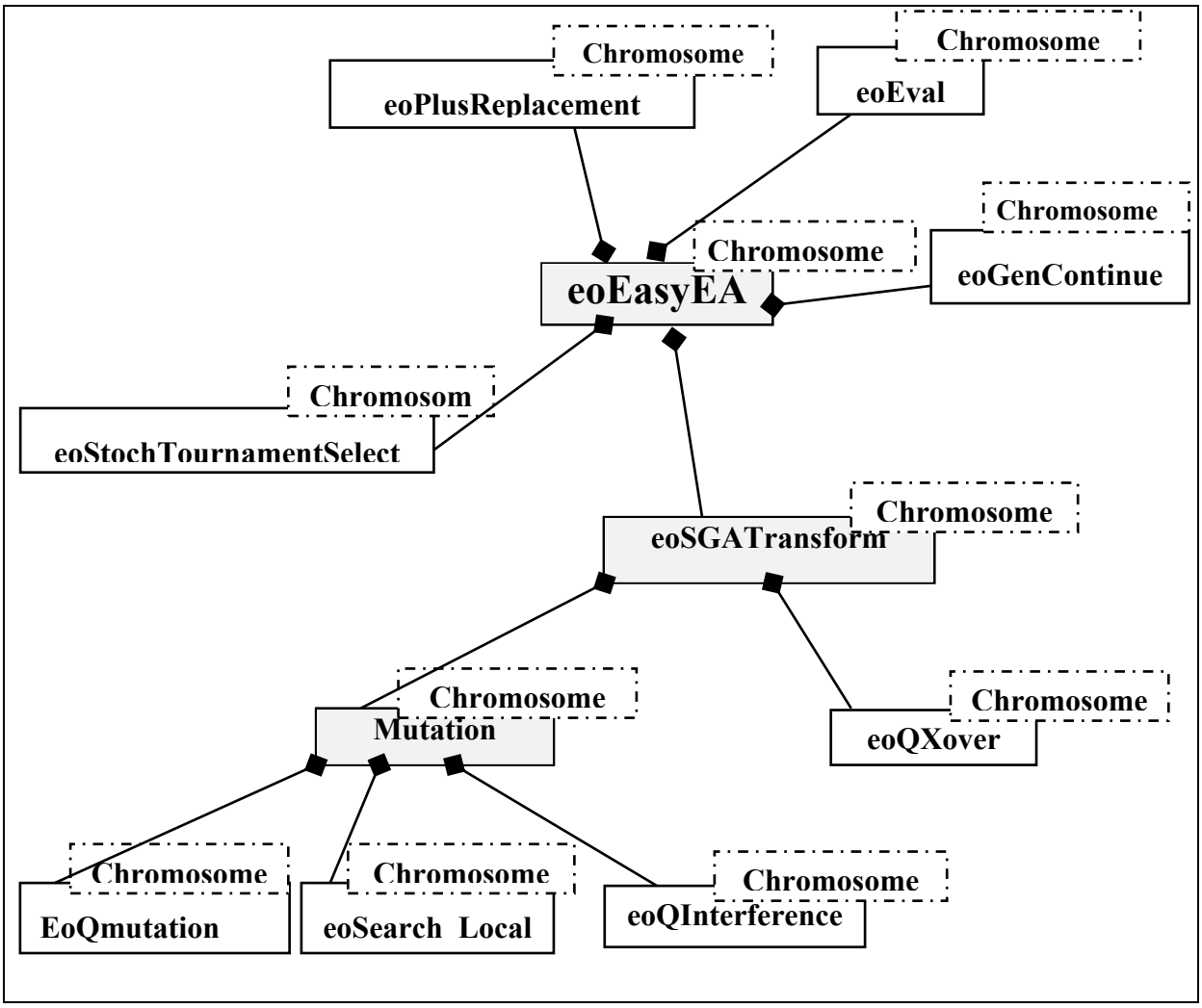


Figure 6.11 – Le diagramme de classes du noyau quantique proposé.

La classe principale est dérivée de la classe prédéfinie ***eoEasyEA*** qui implémente les paramètres de l'algorithme quantique génétique :

- La fonction objectif est implémentée par la classe **eoEval**
- Une méthode de remplacement implémentée par la classe **eoPlusReplacement**
- Une méthode de sélection implémentée par la classe **eoStochTournamentSelect**
- Une transformation implémentée par la classe **eoSGATransform**
- Une opération de croisement implémentée par la classe **eoQXover**
- Une opération de mutation simple implémentée par la classe **EoQmutation**
- La recherche locale implémentée par la classe **eoSearch_Local**
- Une opération d'interférence implémentée par la classe **eoQInterference**
- Un critère d'arrêt implémenté par la classe **eoGenContinue.**

La partie suivante du code en C++ de l'algorithme QGASAT montre comment intégrer les différents paramètres pour créer un algorithme quantique génétique :

```
QGA (checkpoint , eval,select, transform, merge, reduce)
```

L'exécution de l'algorithme est déclenchée en exécutant l'instruction :

```
QGA (population)
```

Algorithme 6.3 : Un pseudo code en C++ de l'approche QGASAT

```
……
eoIndividual indiv ;
eoIndividualInit init ;
init(indiv) ;
eoIndividualEval  eval ;
eoQInterference interference ;
eoQmutation mutation ;
eoQXover cross ;
eoQTransmut<eoIndividual>  transform(interference,  mutation,0.5,
cross, 1) ;
eoGenContinue <eoIndividual> cont (NUM_GEN)   ;
eoCheckPoint <eoIndividual> checkpoint (cont) ;
eoElitism <eoIndividual> merge (1) ;
eoStochTournamentTruncate <eoIndividual> reduce (0.7) ;
eoStochTournamentSelect <eoIndividual> select_one ;
eoSelectNumber <eoIndividual> select (select_one, 4) ;
eoQGA  <eoIndividual>  QGA  (checkpoint  ,  eval,select,  transform,
merge, reduce) ;
eoPop <eoIndividual> population (POP_SIZE, init) ;
QGA (population) ;
…
```

6.5.2 Evaluation

Nous présentons dans cette section les instances que nous avons utilisées pour mesurer les performances, et nous comparons nos résultats avec d‟autres algorithmes de la littérature. QGASAT et QHILLSAT sont testés sur un micro-ordinateur avec un processeur de 3 GHz et 1 GB de mémoire. Pour évaluer efficacités QGASAT et QHILLSAT, on a procédé à un certain nombre d'expériences. Ces expériences nous permettent également de différencier les deux méthodes en termes de performance sur le problème MAX 3-SAT en les comparant à des approches plus classiques. Nous avons utilisé un nombre de tests satisfiables et non satisfiables de la base des tests AIM [Asahiro et al, 1996]. Les tests de la base AIM sont générés par un générateur Random-3-SAT.

Afin d‟analyser les performances des approches développées, on a effectué une comparaison avec d‟autres programmes populaires dans la littérature. Le premier programme nommé GASAT est basé sur un algorithme génétique pur. Le but de cette comparaison avec GASAT et de montrer l‟impact de la recherche locale dans la résolution du problème MAX 3-SAT. Deuxièmement, nous avons comparé nos résultats avec les deux programmes de référence basés sur la recherche locale pure : le GSAT [Selman et al., 1992] et le WalkSat [Selman et Kautz, 1993]. Finalement, nous avons comparé nos résultats avec ceux d‟une méthode exacte basée sur l‟algorithme exact DPLL [Davis et al., 1962]. Pour la validation statistique de nos résultats, on a utilisé les tests statistiques de Friedman et de Wilcoson (Wilcoxon's signed rank sum test) [Hollander et al, 1999]. Ces deux tests sont utilisés pour déterminer les différences de performances entre les méthodes et trouver donc les méthodes qui résolvent efficacement le problème MAX 3-SAT par rapport à d'autres méthodes. Le test de Friedman est utilisé pour tester si la différence entre les médianes des méthodes n'est pas significative. Par ailleurs, le test Wilcoxon signed rank teste si la médiane de la différence entre les résultats de deux méthodes est égale à 0. Nous avons utilisé les meilleurs paramètres pour chaque méthode QGASAT et QHILLSAT qui nous donnent les meilleurs résultats. Finalement, nous avons pris le meilleur résultat trouvé de trois exécutions successives.

6.6 Résultats et discussions

Les tableaux 6.1 montrent les résultats des expérimentations que nous avons menées pour évaluer l'efficacité des méthodes proposées. Ces résultats montrent clairement que la méthode QGASAT donne des résultats proches de l‟optimal. En effet, dans le test de Friedman pour un seuil de 0.05, seuls QGASAT et WalkSat qui sont les plus proches à la solution exacte (figure 6.12). D‟autre part, selon le même test de Friedman, les méthodes QGASAT, GSAT, WalkSat et QHILLSAT ont les mêmes performances parce qu‟il n‟existe pas une différence significative c‟est-à-dire elles sont statistiquement similaires.

D‟après les statistiques effectuées, le programme WalkSat est le plus proche à la méthode exacte, contrairement au programme GASAT basé sur un pur algorithme génétique et qui est le moins bon dans cette expérimentation (figure 6.12). Cet algorithme est moins compétitif dans le domaine de la recherche locale, malgré une recherche globale relativement efficace, cela confirme les conclusions faites par Rana et Whitley [Rana et al, 1998] sur l‟inefficacité des algorithmes génétiques classiques pour un problème aussi compliqué que le MAX 3-SAT. Il faut donc hybrider les AGs avec une technique de recherche locale afin d‟améliorer la qualité des résultats trouvés. Par ailleurs, QGASAT égalise ou améliore strictement les résultats trouvés par QHILLSAT, ce qui approuve que la méthode de recherche locale utilisée dans le QGASAT est plus efficace que celle utilisée dans QHILLSAT. En effet, sur 60 tests, QGSAT trouve 41 tests exacts, alors que QHILLSAT trouve seulement 35 tests. En outre, le test de Wilcokson confirme que les performances des deux programmes sont différentes.

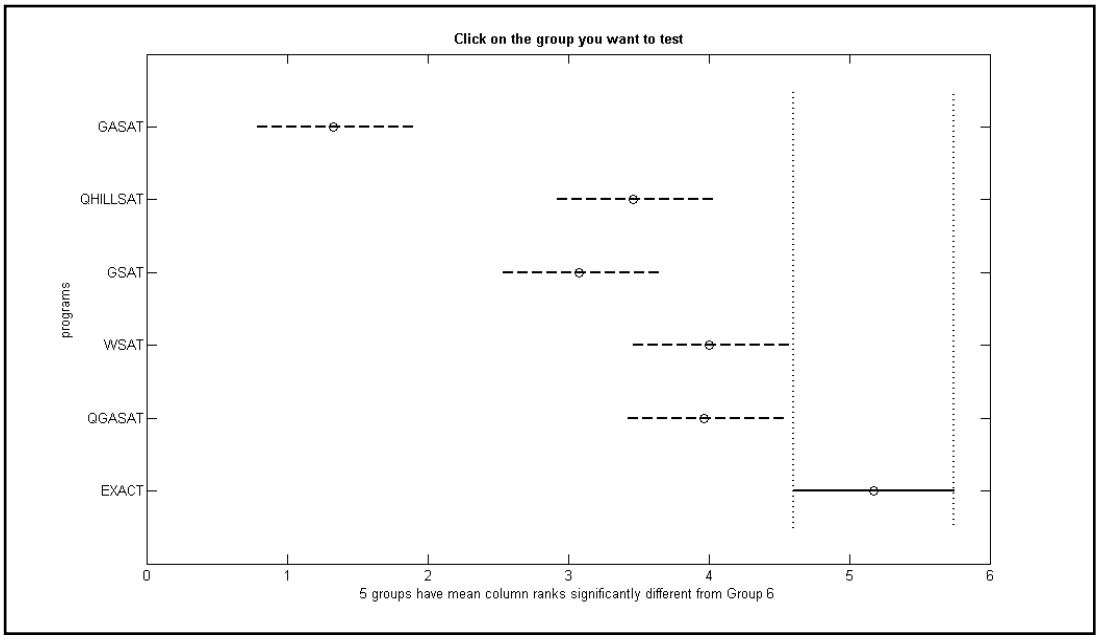


Figure 6.12 – Test de Friedman (a=0.05) pour les méthodes QGAGSAT, GASAT, QGASAT (tous les tests).

Après que nous avons vu les performances de chaque programme dans la totalité des tests utilisés, maintenant nous allons étudier les performances de chaque programme sur deux ensembles de tests. Le premier ensemble contient seulement des tests satisfiables et l‟autre des tests non satisfiables. Concernant les performances des programmes dans les tests non satisfiables, à l‟exception du programme GASAT, tous les autres programmes ont réussi dans ce test (figure 6.13). En effet, le test de Friedman montre clairement que les résultats trouvés par les programmes QGSAT, QHILLSAT, GSAT, et WalkSat égalisent les résultats trouvés par une méthode exacte. En revanche, aucun programme n‟est réussi dans les tests satisfiables comme il est confirmé par le test de Friedman (figure 6.14).

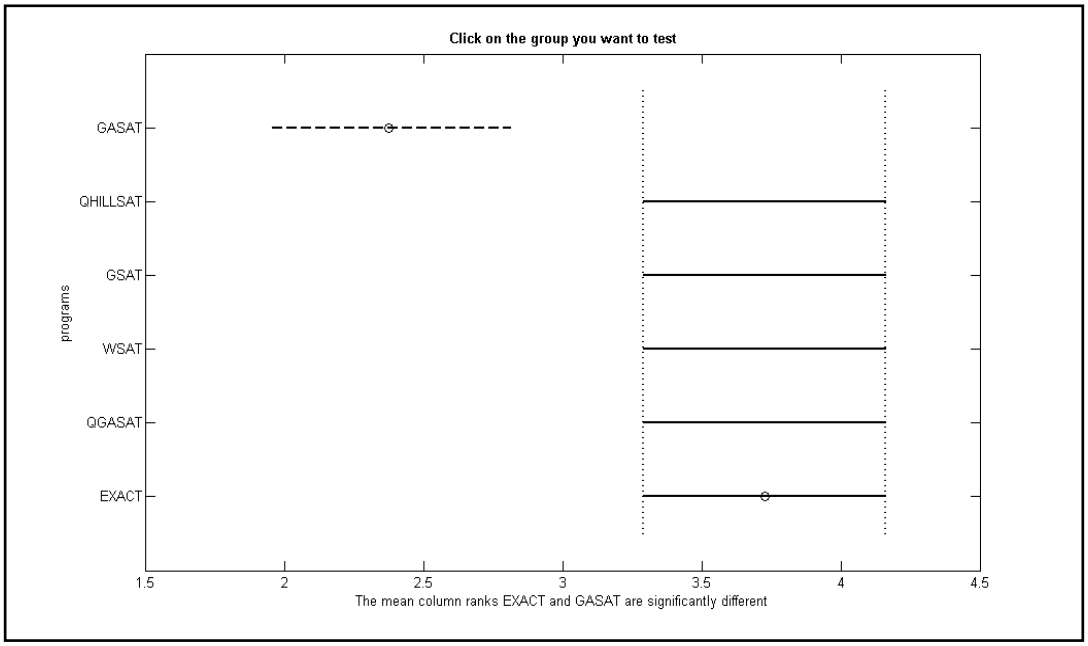


Figure 6.13 – Test de Friedman (a=0.05) pour les méthodes QGAGSAT, GASAT, QGASAT, (les tests non satisfiables).

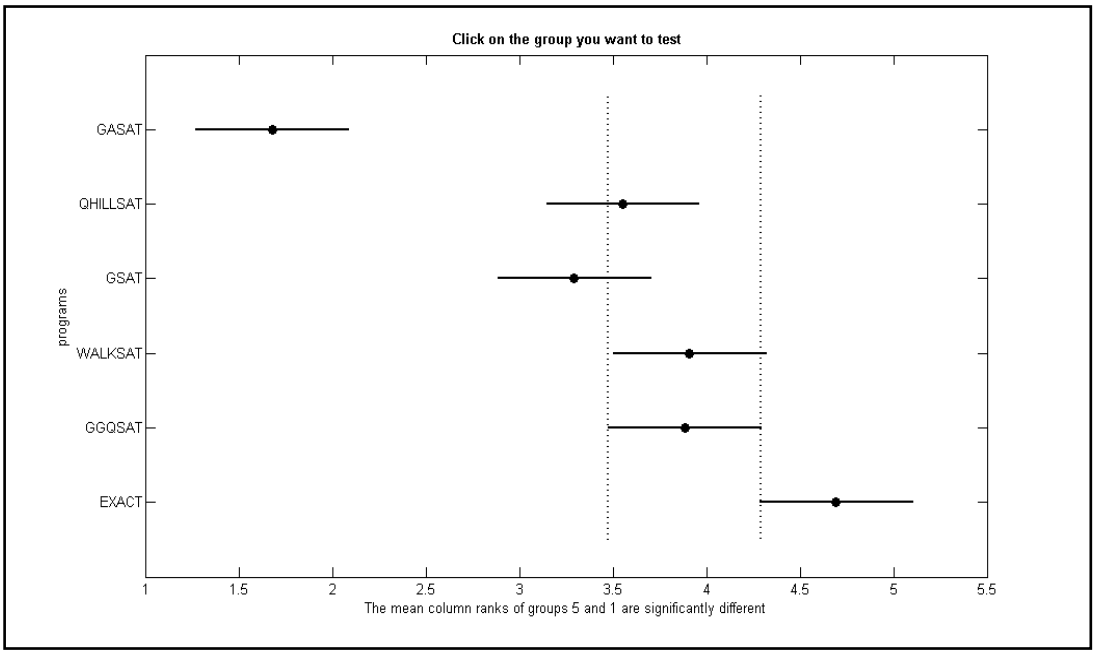


Figure 6.14 – Test de Friedman (a=0.05) pour les méthodes QGAGSAT, GASAT, QGASAT, (les tests satisfiables).

La figure suivante montre le comportement de la fonction de fitness pour un test est contient 100 variables et 200 clauses. D‟après cette figure, on voit clairement la capacité d‟optimisation de l‟approche QGASAT, et que cette optimisation n‟est pas faite d‟une manière aléatoire.

Itérations

Figure 6.15 – Courbe de progression de la meilleure solution pour le test aim-100-2-0-no-1.

6.7 Hybridation entre un AQG est une méthode exacte

Dans cette section, nous nous intéressons aux possibilités d'hybridation entre les méthodes exactes et les méthodes heuristiques afin de pouvoir tirer avantage de chacune des deux approches : systématicité et optimalité de la résolution exacte, caractère moins déterministe et rapidité de la composante heuristique [Layeb et al, 2010a]. Nous montrons comment un algorithme quantique génétique peut être grandement amélioré en l'hybridant avec une méthode exacte. Les méthodes exactes donnent théoriquement des solutions optimales parce qu'elle effectue une recherche exhaustive et donc exploite complètement l'espace de recherche. Malheureusement, elles sont impraticables dans le cas des tests de grandes tailles. Par ailleurs, les méthodes approchées sont caractérisées par une grande simplicité et rapidité parce qu'elles explorent que certaines zones de l'espace de recherche. Elles donnent toutefois des solutions moins optimales. Afin de tirer profit des caractéristiques des deux approches, L'hybridation entre ces deux approches parait intéressante. La notion d'hybridation est ici un abus de langage. Il existe seulement deux classes d'algorithmes de résolution pour SAT : complets ou incomplets. La notion d'hybridation provient du fait que l'on intègre généralement entre eux des algorithmes complets et incomplets. Mais, le résultat appartient soit à la classe des algorithmes complets ou incomplets selon la nature de cette intégration. Mais en aucun cas il ne s'agit d'une troisième classe de résolution.

Notre idée consiste donc à intégrer le pouvoir d'intensification de la méthode exacte DPLL au sein de l'algorithme QGASAT qui possède des facultés de diversification et ce, afin d'obtenir un bon compromis entre ces deux stratégies fondamentales de recherche et sans augmenter grandement le temps de calcul. Pour cela, l'algorithme exact DPLL est incorporé dans l'algorithme QGASAT en tant que qu'une méthode de recherche locale (figure 6.16). Dans ce cas, notre algorithme hybride se comporte comme un algorithme incomplet. L'idée est basée sur un concept très connu en informatique qui est *diviser pour régner* « divide to

conquer » qui consiste à diviser arbitrairement le problème en plusieurs sous-problèmes facilement résoluble. Afin d‟utiliser l‟efficacité des algorithmes exactes sans toutefois augmenter le temps d‟exécution, notre méthode de recherche locale applique d‟une manière itérative la procédure DPLL sur des parties de la formule booléenne. Le principe consiste d‟abord à choisir un intervalle continu de clauses (figure 6.17). Par conséquent, Cet intervalle constitue une formule booléenne avec peu de variables et peu de clauses, ce qui rend l‟application de l‟algorithme exacte facile. La taille de la fenêtre de sélection est dynamique afin de maintenir plus d‟efficacité

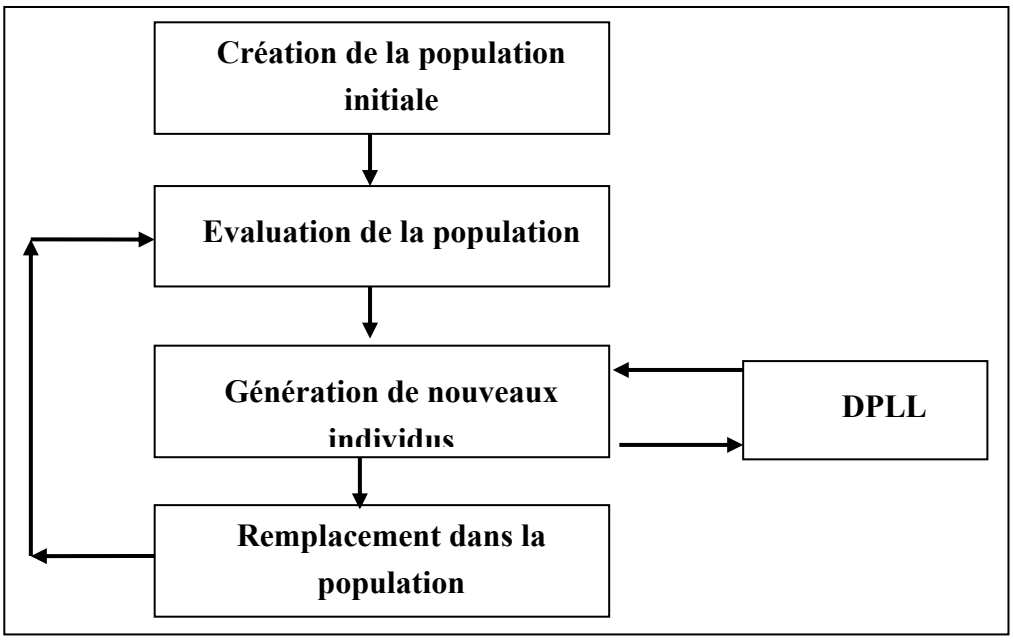


Figure 6.16 – Schéma générale d'un AQG hybride avec une méthode exacte.

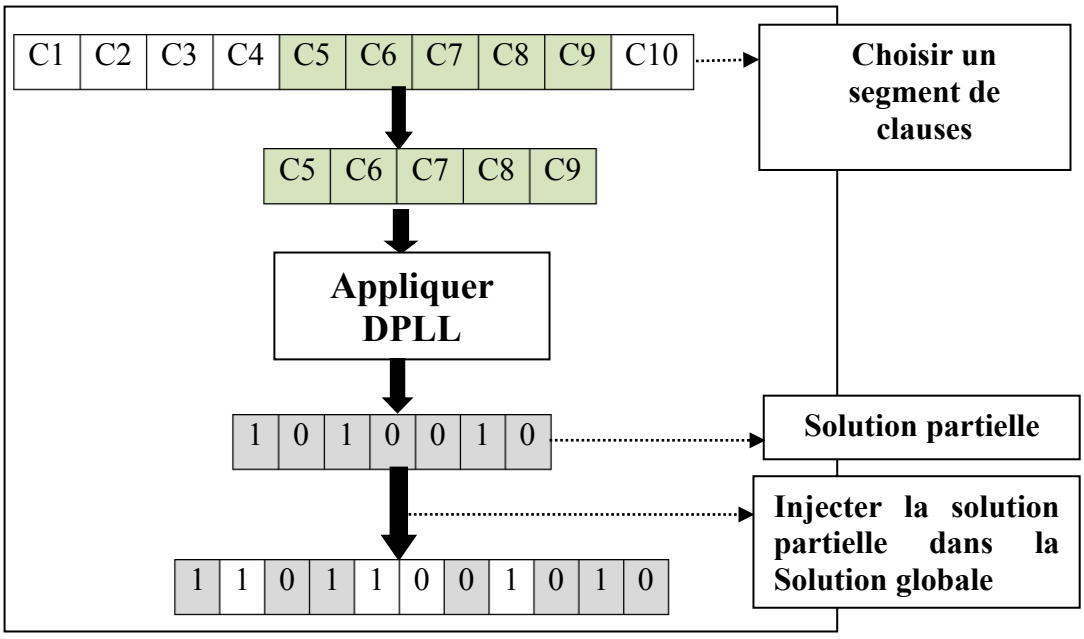


Figure 6.17 – Exemple d'application partielle de l'algorithme DPLL.

D‟une manière abstraite, cette nouvelle méthode de recherche locale opère comme suit :

Agorithme 6.4 : une recherche local basée DPLL pour MAX SAT :
Données: un vecteur de qubit VQ, une formule ϕ en CNF, Nbtries.
Résultat: un vecteur binaire VB

```
Début
  VB=mesure(VQ) ;
  Gain0=Evaluer (VB) ;
  Répéter
    choisir aléatoirement un intervalle de clauses IC;
    Appliquer la procédure DPLL sur IC;
    Modifier VB selon le résultat trouvé par DPLL ;
    Gain1=Evaluer(VB) ;
    Si (Gain1- Gain0)>0 Alors
      Sauvegarder VB;
    Sinon
       annuler les modifications apporter à VB;
    finsi
  Jusqu'à Nbtries ;
Fin
```

Notre nouvelle méthode hybride nommée QGADPLL [Layeb et al, 2010a] est implémentée en java et testée sur un micro-ordinateur. On a utilisé le package SAT4J [LeBerre, 2006] qui contient une implémentation en java de l‟algorithme exacte DPLL. Nous avons testé cette méthode sur un nombre de tests satisfiables et non satisfiables de la base des tests AIM. Les résultats trouvés semblent être très prometteurs et démontrent la faisabilité et l‟efficience de notre approche (table 6.3). En effet les résultats trouvés sont très proches à la solution exacte (figure 6.18). La figure 6.19 montre la variation de la meilleure solution trouvée par la méthode QGADPLL ce qui montre l‟amélioration graduelle de la meilleure solution. QGADPLL permet ainsi d'assurer un compromis entre la qualité de la solution retenue et le temps nécessaire de calcul. En effet, le temps moyen pour trouver une bonne solution et largement inferieure que les autres méthodes évolutionnaires. Cette expérience a montré que l'utilisation d‟une procédure exacte au sein de l‟algorithme quantique génétique apporte une amélioration significative en termes de performance et d'efficacité. En outre, notre idée constitue une voie très prometteuse pour résoudre des instances satisfiables de grande taille.

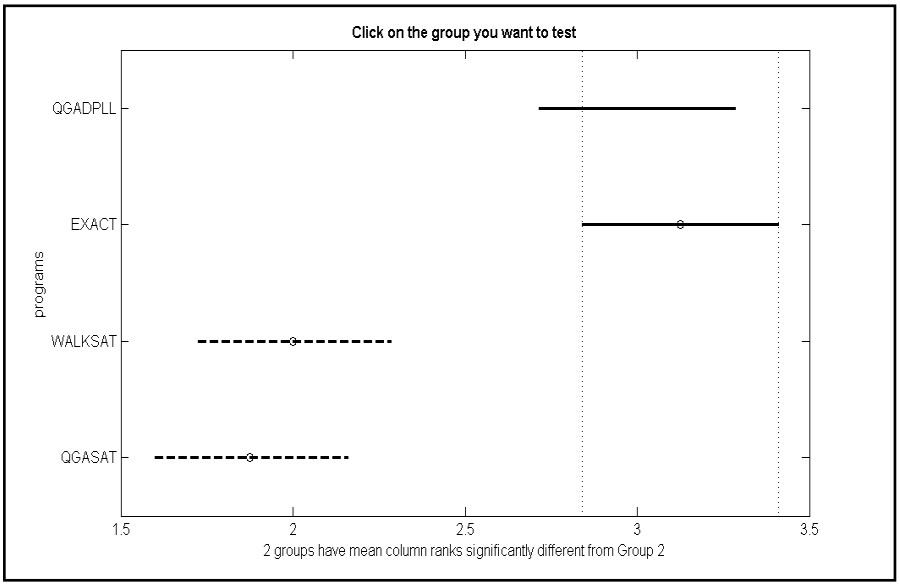


Figure 6.18 – le test de Freidman: QGADPLL vs exacte , walksat and QGSAT.

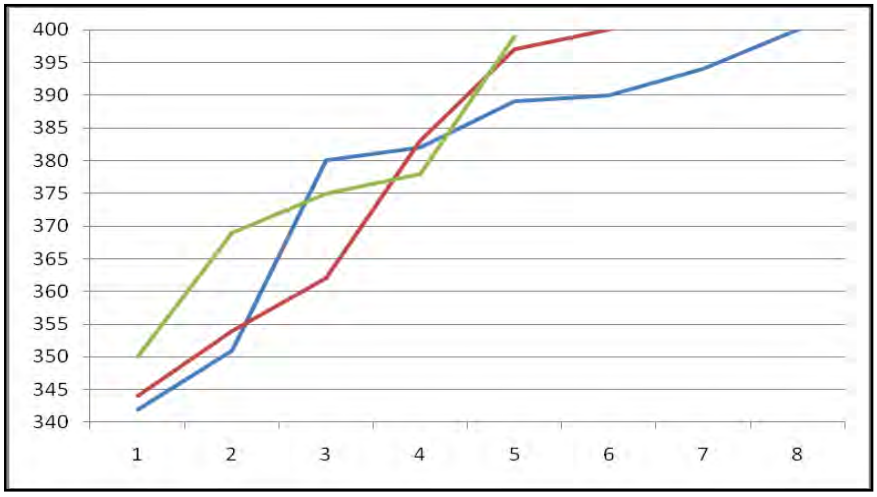


Figure 6.19 – Le comportement de la meilleure solution pour le test : Aimyes25. Les trois courbes montrent l'évolution de trois exécutions différentes.

6.8 Utilisation des systèmes adaptatifs pour les problèmes SAT

Après qu‟on a vu les performances des algorithmes évolutionnaires hybrides pour la résolution du problème de satisfiabilité booléenne. Dans cette section nous allons étudier les performances des systèmes adaptifs pour ce type de problème. Parmi les systèmes adaptatifs les plus connus, on distingue les Systèmes Immunitaires Artificiels (Artificial Immune Systems AIS). Les domaines d'application des systèmes immunitaires artificiels sont très variés et ceci grâce à la polyvalence et la puissance du système immunitaire biologique duquel ils s'inspirent. Parmi ces domaines, on trouve la détection de fautes et des anomalies, reconnaissance de formes, l'apprentissage, la sécurité des systèmes d'informations, les méthodes d'optimisation, etc. Dans ce contexte, nous proposons dans cette thèse, une nouvelle

approche itérative nommée CloneSat [Layeb et al, 2010b] basée sur un AIS pour résoudre le problème de satisfiabilité maximale. Les traits de cette nouvelle approche consistent en l'application des principales opérations immunitaires, comme la sélection clonale et la mutation afin d'optimiser une certaine fonction objectif. Pour accélérer la convergence vers l'optimal, une méthode de recherche locale a été intégrée dans la dynamique d'optimisation. Les résultats trouvés montrent un comportement intéressant de cette nouvelle hybridation.

6.8.1 Description de l'algorithme ClonSat

La structure détaillée de l'algorithme basé sur un système immunitaire artificiel pour le problème de satisfiabilité booléenne appelé ClonSat [Layeb et al, 2010b] est décrite par l'algorithme suivant :

Algorithme 6.5 : le schéma de l'algorithme ClonSat :
Données: Une formule ϕ en CNF, Nbtries.
Résultat: un vecteur binaire VB

Début

Générer aléatoirement une population initiale d'anticorps *Ab*.

Diviser cette population en de deux sous-ensembles Ab_m (population mémoire) et Ab_r (population réservoir).

Répéter

a) pour chaque élément de *Ab* calculer son affinité.

b) Sélectionner les meilleurs n anticorps et générer pour chaque anticorps sélectionné un nombre de clones pour créer une nouvelle population C^i.

c) Muter les éléments de la population de clones C^i produire une population mature C^{i*}.

d) Réappliquer la fonction d'affinité aux membres de la population mature C^{i*} et sélectionner le meilleur individu comme cellule candidate. Si son affinité est meilleure que celle de la cellule mémoire courante Abm_i, alors la cellule candidate remplace l'ancienne cellule mémoire.

e) Remplacer les anticorps de Abr par les meilleurs anticorps de Ci*.

f) Eliminer les mauvais anticorps de Abr et les remplacer par de nouveaux anticorps générés aléatoirement.

Jusqu'à G génération ;

Fin

Maintenant, nous décrivons comment ClonSat résout le problème MAX-SAT. ClonSAT commence par une population initiale d'anticorps. Ces derniers encodent les solutions initiales. Généralement, pour résoudre un problème en utilisant les principes de la sélection

clonale, chaque solution possible est représentée par un anticorps et le problème est un antigène. Afin de pouvoir adapter l'AIS pour le problème posé, nous avons utilisé une représentation binaire pour encoder les solutions potentielles (anticorps). De ce fait, une interprétation possible est représentée par un vecteur binaire de taille n où n est le nombre de variables booléennes dans la formule en CNF. ClonSat peut utiliser des solutions initiales obtenues en utilisant un algorithme de recherche locale ou utiliser des solutions complètement aléatoires. Ensuite, ClonSat effectue d'une manière itérative les opérations suivantes:

Premièrement, il calcule l'affinité de chaque élément de la population. L‟affinité d‟un élément est la mesure de la fitness, dans notre cas l‟affinité est la somme de clauses satisfaites par l‟interprétation représentée par un anticorps. Dans la deuxième étape, on sélectionne N meilleurs anticorps en tenant compte de leurs affinités. Ensuite, les anticorps sélectionnés sont clonés afin de produire une population clone. Pour chaque anticorps sélectionné le nombre de clones produits est calculé par l'équation suivante [Layeb et al, 2010b]:

$$nbClones_i = \frac{affinité_i}{\sum_j affinité_i} \times \beta \qquad (6.1)$$

Où $affinité_i$ est l'affinité de l'i^{éme} anticorps sélectionné et β est un paramètre indiquant la taille de la population clone.

La troisième opération utilisée est la mutation. Elle consiste à muter chaque clone pour produire la population *mature*. Le processus de mutation est le suivant:

a) Pour chaque clone $clone_i$: Calculer l'affinité normalisée du $clone_i$:

$$affinitéN_i = \frac{affinité_i - \min Affinité}{\max Affinité - \min Affinité} \qquad (6.2)$$

Où min *affinité* (max *affinité*) est l'affinité minimale (maximale) trouvée dans la population *clone*.

b) Calculer le nombre de mutations à appliquer au $clone_i$:

$$\text{nbMutations}_i = affinité_i \times M\min + (1 - affinité_i) \times M\max \qquad (6.3)$$

Mmin (Mmax) est le nombre de mutations du meilleur (mauvais) clone respectivement.

c) Pour chaque mutation de $nbMutations_i$, ClonSat applique une mutation simple qui consiste à flipper quelques bits pris aléatoirement.

Après l'étape de mutation, ClonSat évalué l'affinité de la population mature et sélectionne les meilleurs anticorps comme des cellules candidates. Si la cellule candidate est meilleure que la cellule mémoire elle devient la nouvelle cellule mémoire. Dans la dernière étape, ClonSat remplace tous les anticorps de la population des anticorps avec les meilleurs de la population des anticorps et la population mature. Enfin, il remplace les mauvais éléments de

la population des anticorps par de nouveaux anticorps générés aléatoirement. Le processus est réitéré jusqu'à la satisfaction des conditions finales.

Afin d'augmenter la capacité d'intensification du processus de recherche, une méthode de recherche locale a été intégrée. L'algorithme ClonSat utilise la recherche locale pour améliorer une cellule mémoire quand il n y a pas d'amélioration pendant un certain nombre de génération. Pour cela nous avons essayé ClonSat avec deux méthodes de recherche locale différentes. La première est une procédure FLIP similaire à celle utilisée dans la méthode QGASAT à l'exception qu'on travaille avec des bits au lieu des qubit. La deuxième méthode est basée sur une version basique de la bien connue méthode de recherche locale WalKSat.

6.8.2 Implémentation, résultats et discussion

La méthode ClonSat est implémentée en java et testée sur un micro-ordinateur de 3 GHZ et 1 GB de mémoire. Nous avons testé cette méthode sur un nombre de tests satisfiables et non satisfiables de la base des tests AIM. ClonSat est expérimenté avec les deux méthodes de recherche locale Flip et WalkSat. Des comparaisons avec d'autres algorithmes sont également faites. Dans toutes les expériences, nous avons utilisé les paramètres montrés sur le tableau 6.4. Bien qu'il faille plus d'expériences et de comparaisons, les résultats trouvés montrent la faisabilité des systèmes immunitaires artificiels pour traiter le problème MAX-SAT. En effet, dans la plus part des cas, les résultats de ClonSat avec la méthode de recherche Flip sont presque similaires au programme classique GSAT. D'autre part, ClonSat est nettement meilleur que le programme QSAT (table 6.5). Ce dernier est un algorithme évolutionnaire quantique avec une simple procédure de recherche locale. L'ajout de la méthode WalkSat a permis de traiter les instances les plus dures et d'accélérer la convergence de l'algorithme vers les solutions optimales. De même, ce travail constitue une plateforme extensible pour d'autres problèmes de satisfiabilité booléenne.

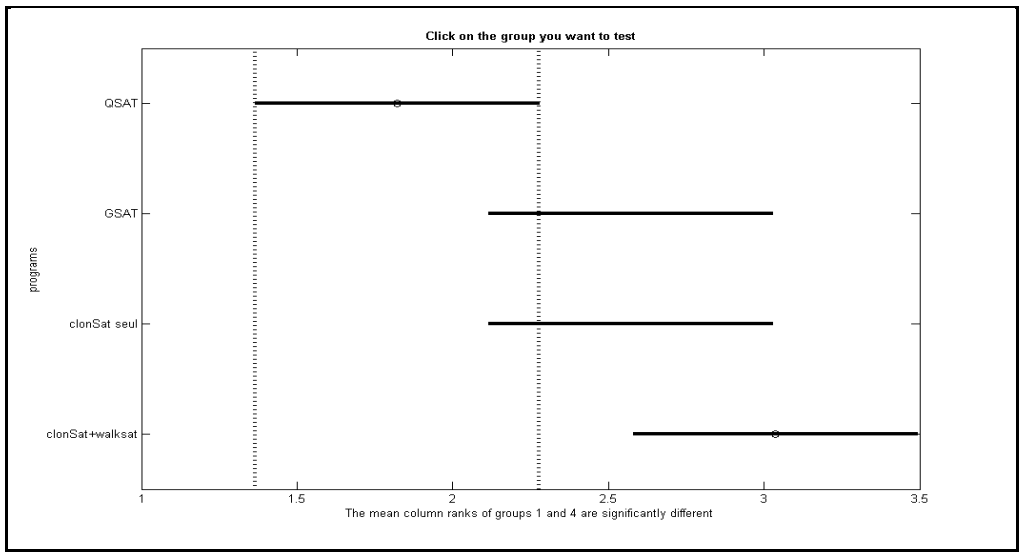


Figure 6.20 – Test de Friedman (a=0.05) compare ClonSat avec d'autres méthodes.

6.9 La complexité des approches développées

La complexité des approches développées est dépendante de plusieurs facteurs. La complexité pour résoudre une formule qui possède ***n*** variables et ***m*** clauses est :

6.9.1 La complexité des approches QHILLSAT, QGASAT, et QGADPLL

Les étapes les plus gourmandes en temps CPU dans ces méthodes sont les suivantes:

- La complexité pour de construire un individu est de l'ordre de **n**
- La complexité pour créer une population de taille **p** est **: n* p**
- La complexité de l'évaluation de la fonction objectif est de l'ordre: **n*m**
- La complexité de la mutation, le croisement, la mesure et de l'interférence est de l'ordre de **n**
- On suppose que la complexité de la recherche locale est : **C_sls.** Généralement, la complexité des méthodes de recherche locale utilisées QGASAT et QHILLSAT est inférieure à O(n∗it), ou *it* est le nombre d'itérations de la procédure de recherche locale.
- En conséquence, la complexité des approches développées pour un nombre de génération **G** est approximativement égale à:

$$\textbf{\textit{Complex}} \cong n^* p + G^* p^*(4^* n^* m + C\_sls\ ) \quad (6.4)$$

6.9.1 La complexité de l'approche ClonSAT

Les étapes qui consomment considérablement le temps CPU dans ClonSat sont :

- La complexité pour de construire un anticorps est de l'ordre de **n**
- La complexité pour créer une population d'anticorps de taille **p** es**t : n* p**
- La complexité de l'évaluation de la fonction d'affinité affinité est de l'ordre: **n*m**
- la Complexité pour générer un nombre de clones est **Nb_clones**
- la Complexité pour muter tous les clones dans une seule génération est**:**

$$C\_mut = (affinitéN + affinité \times M\min + (1 - affinité) \times M\max\ )^* Nbclones \quad (6.5)$$

où Affinité est l'affinité normalisée (algorithme 6.4)

- La complexité totale pour l'évaluation de la fonction d'affinité de tous les clones dans une seule génération est de l'ordre: ***n*m* Nb_clones.***
- On suppose que la complexité de la recherche locale FLIP ou WalkSAT est : **C_sls.**

En conséquence, la complexité de la méthode ClonSat pour un nombre de génération **G** est approximativement égale à :

$$\textbf{\textit{Complex}} \cong n^* p + G^* P^*(n^* m^* Nb\_clones + C\_mut\ + C\_sls\ ) \quad (6.6)$$

Table 6.1. Résultats pour les tests satisfiables.

Benchmark	#var	#clause	GASAT	QHILLSAT	GSAT	WSAT	QGASAT	EXACTE
aim-50-1_6-yes1-1	50	80	79	79	79	79	79	80
aim-50-1_6-yes1-2	50	80	79	79	79	79	79	80
aim-50-1_6-yes1-3	50	80	78	79	79	79	79	80
aim-50-1_6-yes1-4	50	80	79	79	79	79	79	80
aim-50-2_0-yes1-1	50	100	99	99	99	99	99	100
aim-50-2_0-yes1-2	50	100	99	99	100	100	100	100
aim-50-2_0-yes1-3	50	100	99	99	99	99	99	100
aim-50-2_0-yes1-4	50	100	98	99	100	100	100	100
aim-50-3_4-yes1-1	50	170	168	170	168	170	170	170
aim-50-3_4-yes1-2	50	170	167	170	168	170	170	170
aim-50-3_4-yes1-3	50	170	167	170	168	170	170	170
aim-50-3_4-yes1-4	50	170	168	170	167	170	170	170
aim-50-6_0-yes1-1	50	300	288	298	287	300	300	300
aim-50-6_0-yes1-2	50	300	288	300	289	300	300	300
aim-50-6_0-yes1-3	50	300	287	300	288	300	300	300
aim-50-6_0-yes1-4	50	300	285	300	286	300	300	300
aim-100-1_6-yes1-1	100	160	158	159	159	159	159	160
aim-100-1_6-yes1-2	100	160	158	159	159	159	159	160
aim-100-1_6-yes1-3	100	160	157	159	159	159	159	160
aim-100-1_6-yes1-4	100	160	157	159	159	159	159	160
aim-100-2_0-yes1-1	100	200	196	199	199	199	199	200
aim-100-2_0-yes1-2	100	200	194	199	199	199	199	200
aim-100-2_0-yes1-3	100	200	195	199	199	199	199	200
aim-100-2_0-yes1-4	100	200	195	199	199	199	199	200
aim-100-3_4-yes1-1	100	340	323	335	336	340	339	340
aim-100-3_4-yes1-2	100	340	325	335	336	340	340	340
aim-100-3_4-yes1-3	100	340	328	336	336	340	340	340
aim-100-3_4-yes1-4	100	340	325	334	335	340	340	340
aim-100-6_0-yes1-1	100	600	583	600	600	600	600	600
aim-100-6_0-yes1-2	100	600	572	600	600	600	600	600
aim-100-6_0-yes1-3	100	600	575	600	600	600	600	600
aim-100-6_0-yes1-4	100	600	574	600	600	600	600	600
aim-200-2_0-yes1-1	200	400	383	399	399	399	399	400
aim-200-2_0-yes1-2	200	400	381	398	399	399	399	400
aim-200-2_0-yes1-3	200	400	387	399	399	399	399	400
aim-200-2_0-yes1-4	200	400	384	399	399	399	399	400
aim-200-6_0-yes1-1	200	1200	1114	1200	1146	1200	1200	1200
aim-200-6_0-yes1-2	200	1200	1112	1200	1200	1200	1200	1200
aim-200-6_0-yes1-3	200	1200	1112	1200	1165	1200	1200	1200
aim-200-6_0-yes1-4	200	1200	1129	1200	1200	1200	1200	1200

Table 6.2. Résultats pour les tests non-satisfiables.

Benchmark	#var	#clause	GASAT	QHILLSAT	GSAT	WSAT	QGASAT	EXACTE
aim-50-1_6-no-1	50	80	79	79	79	79	79	79
aim-50-1_6-no-2	50	80	79	79	79	79	79	79
aim-50-1_6-no-3	50	80	79	79	79	79	79	79
aim-50-1_6-no-4	50	80	79	79	79	79	79	79
aim-50-2_0-no-1	50	100	99	99	99	99	99	99
aim-50-2_0-no-2	50	100	99	99	99	99	99	99
aim-50-2_0-no-3	50	100	99	99	99	99	99	99
aim-50-2_0-no-4	50	100	99	99	99	99	99	99
aim-100-1_6-no-1	100	160	159	159	159	159	159	159
aim-100-1_6-no-2	100	160	158	159	159	159	159	159
aim-100-1_6-no-3	100	160	159	159	159	159	159	159
aim-100-1_6-no-4	100	160	157	159	159	159	159	159
aim-100-2_0-no-1	100	200	199	199	199	199	199	199
aim-100-2_0-no-2	100	200	198	199	199	199	199	199
aim-100-2_0-no-3	100	200	196	199	199	199	199	199
aim-100-2_0-no-4	100	200	197	199	199	199	199	199
aim-200-2_0-no-1	200	400	384	399	399	399	399	399
aim-200-2_0-no-2	200	400	383	399	399	399	399	399
aim-200-2_0-no-3	200	400	383	399	399	399	399	399
aim-200-2_0-no-4	200	400	385	399	399	399	399	399

Table 6.3. Résultats de QGAPLL.

Tests	#var	#clause	QGADPLL	Exact	WSAT	QGASAT	Temps d'exécution (milliseconde)	
							Moyen	Meilleur
Aimyes1	200	1200	1200	1200	1200	1200	3835	1980
Aimyes2	200	1200	1200	1200	1200	1200	2312	412
Aimyes3	200	1200	1200	1200	1200	1200	1650	415
Aimyes4	200	1200	1200	1200	1200	1200	2492	515
Aimyes5	100	600	600	600	600	600	4596	1401
Aimyes6	100	600	600	600	600	600	516	296
Aimyes7	100	600	600	600	600	600	1200	425
Aimyes8	100	600	600	600	600	600	482	347
Aimyes9	50	80	80	80	79	79	473	352
Aimyes10	50	80	80	80	79	79	521	403
Aimyes11	50	80	80	80	79	79	910	191
Aimyes12	50	80	80	80	79	79	752	323
Aimyes13	100	200	200	200	199	199	1138	301
Aimyes14	100	200	200	200	199	199	699	433
Aimyes15	100	200	200	200	199	199	1631	940
Aimyes16	100	200	200	200	199	199	2603	304
Aimyes17	100	340	340	340	340	339	1593	292
Aimyes18	100	340	340	340	340	337	779	400
Aimyes19	100	340	340	340	340	340	1765	773
Aimyes20	100	340	340	340	340	340	726	273
Aimyes21	100	160	159	160	159	159	782	363
Aimyes22	100	160	160	160	159	159	2098	563
Aimyes23	100	160	160	160	159	159	834	310
Aimyes24	100	160	160	160	159	159	797	359
Aimyes25	200	400	400	400	399	399	1801	861
Aimyes26	200	400	400	400	399	399	1472	366
Aimyes27	200	400	400	400	399	399	1567	499
Aimyes28	200	400	400	400	399	399	1972	1045
Aimyes29	50	100	100	100	99	99	619	343
Aimyes30	50	100	100	100	100	100	1903	738
Aimyes31	50	100	100	100	99	99	739	305
Aimyes32	50	100	100	100	100	100	722	192

Table 6.4. Les paramètres de ClonSat.

Paramètre	**valeur**
La taille de population	100
La probabilité de mutation minimale	1 / nb variables
La probabilité de mutation maximale	0.01
La taille des clones (β)	200
La taille du replacement	10
Le nombre total de générations avant l'arrêt	1000
Le nombre total de générations avant l'application de WalkSat	100
La probabilité p de WalkSat	0.2
Le nombre de flips pour WalkSat	500

Table 6.5. les Results de Clonsat.

Benchmark	**# Var (n)**	**# clauses (m)**	**QSAT**	**GSAT**	**ClonSAT + Flip**	**ClonSAT +WalkSat**
Aim-50-1_6-no-1	50	80	79	79	79	79
Aim-50-1_6-no-2	50	80	79	79	79	79
Aim-50-1_6-no-3	50	80	78	79	79	79
Aim-50-1_6-no-4	50	80	79	79	79	79
Aim-200-2_0-no-1	200	400	396	399	399	399
Aim-200-2_0-no-2	200	400	397	399	399	399
Aim-200-2_0-no-3	200	400	397	399	399	399
Aim-50-1_6-yes1-1	50	80	80	79	79	80
Aim-50-1_6-yes1-2	50	80	80	79	79	79
Aim-50-1_6-yes1-3	50	80	79	79	79	80
Aim-100-2_0-no-1	100	200	198	199	199	199
Aim-100-2_0-no-2	100	200	197	199	199	199
Aim-100-3_4-yes1-3	100	340	335	335	336	340
Aim-100-3_4-yes1-4	100	340	333	340	336	340

6.10 Conclusion

Dans ce chapitre nous nous sommes intéressés au problème de satisfiabilité booléenne maximale. Au cours de ce travail de doctorat, nous avons proposé quatre nouvelles approches basées sur des hybridations entre des algorithmes évolutionnaires, des méthodes de recherche locale, et des méthodes exactes pour résoudre le problème MAX 3-SAT. L'originalité de ces approches tient au fait qu'elles sont basées sur des cadres de résolution développés spécifiquement pour ledit problème.

Tout d'abord, nous avons présenté deux méthodes mémétiques combinant deux méthodes de recherche locales différentes. Le premier algorithme développé appelé QHILLSAT est basée sur un algorithme de la descente du gradient. Bien qu'il donne des résultats satisfaisants son utilisation pratique semble difficile à cause de la lenteur de la procédure de recherche. Tandis que la deuxième approche appelée QGASAT est basée sur une heuristique de flip. Cette dernière semble être plus rapide et donne des résultats meilleurs que QHILLSAT. Les deux méthodes développées sont basées sur un noyau évolutionnaire quantique. La représentation quantique de la solution permet le codage de l'ensemble des solutions possibles avec une certaine probabilité. Le processus d'optimisation consiste en l'application d'une dynamique quantique constitué d'opérations quantique telles que l'interférence, la mutation, le croisement et de la mesure quantique, renforcé par une procédure de recherche locale. Ce noyau quantique hybride a bénéficié d'une synergie entre les avantages de l'algorithme génétique et ceux de la recherche locale.

Une troisième amélioration en termes d'efficacité est apportée par la troisième méthode QGADPLL qui est une méthode hybride prometteuse qui tire ses avantages, d'une part, de la rigueur et du caractère optimal de l'approche exacte, et d'autre part, de la grande flexibilité de l'approche évolutionnaire quantique. L'idée consiste en l'application itérative de l'algorithme exact DPLL sur des parties plus petites de la formule. Cela a permis de trouver des solutions optimales de certaines instances. En effet, les résultats trouvés par cette approche sont les plus proches à la méthode exacte par rapport aux autres méthodes présentées dans cette thèse.

La quatrième contribution dans ce domaine concerne la proposition d'un cadre de résolution basé sur un système immunitaire artificielle pour résoudre les problèmes de satisfiabilité (ClonSat). Une formulation du problème en termes de représentation clonale a été dérivée et une dynamique évolutionnaire empruntant des opérateurs de mutations et de sélection clonale a été définie. Le processus d'optimisation consiste en l'application d'une dynamique immunitaire constituée d'un ensemble d'opérations immunitaire telles que le clonage, la mutation et la sélection. Les expériences ont montré que notre approche donne des résultats encourageants.

L'enseignement majeur à retenir de cette étude est que le choix de la procédure de recherche locale est crucial pour l'efficacité de l'algorithme résultant. Les résultats obtenus

montrent que l'utilisation d'approches hybrides apporte une amélioration significative de la performance et que l'efficacité de l'heuristique ou de la métaheuristique sur laquelle se base l'hybridation influence également sur la qualité des solutions trouvées. Toutefois, on note également une augmentation importante des temps d'exécution.

Les résultats obtenus montrent l‟intérêt des méthodes proposées et laissent entrevoir les nombreuses perspectives ouvertes par ce type d‟hybridation. En effet, Il reste encore une grande place pour apporter des améliorations à nos travaux tels que l‟adjonction de certains types d‟opérations de croisement bien spécifiques au problème SAT. Une autre amélioration consiste à identifier et l‟exploiter des structures cachées dans un problème SAT qui sont des clés permettant de contrecarrer l‟explosion combinatoire de sa résolution. Mais la grande perspective est la parallélisation de ces méthodes hybrides que nous avons proposées. La parallèlisation peut apporter une grande poussée pour ces approches, par exemple on peut utiliser le modèle en ilots pour accélérer la convergence de ces méthodes. Pour la méthode QGADPLL, on peut diviser une grande instance en plusieurs sous problèmes, puis exécuter la méthode de recherche locale à base DPLL parallèlement sur l‟ensemble de ces sous problèmes.

Conclusions générales et perspectives

Dans cette thèse nous nous sommes intéressés à la résolution de deux problèmes fondamentaux dans le domaine de la vérification formelle à savoir le problème d'ordonnancement des variables des diagrammes de décision binaire et le problème de maximum satisfiabilité booléenne MAX SAT.

Le but de la vérification est de s'assurer qu'un programme ou un matériel électrique fonctionnent comme on le souhaite. Certaines techniques de vérification permettent seulement de détecter des erreurs et d'autres, comme la vérification par évaluation de modèle, permettent de déterminer avec certitude si un certain type d'erreur peut ou non survenir. Un des principaux problèmes en vérification est le problème d'explosion du nombre d'états. Ce problème est causé par une explosion combinatoire qui engendre un très grand nombre d'états dans les modèles à vérifier. Ceci peut entraîner un manque de mémoire et un temps-CPU plus élevé lors de la vérification. Une des solutions proposées pour ce problème est l'utilisation des OBDDs pour représenter l'espace d'états du modèle ainsi que ses transitions. Cette méthode a été utilisée avec succès dans la vérification de systèmes déterministes et probabilistes à espace d'états discret. Malheureusement, pour une seule fonction booléenne, il peut exister plusieurs BDD représentant cette fonction selon l'ordre pris des variables de la fonction lors de la construction du BDD correspondant. Il est démontré que trouver le bon ordre qui minimise la taille d'un BDD est NP-difficile.

Dans cette optique, un premier objectif spécifique visait à concevoir des méthodes de résolution hybrides pour résoudre le problème d'ordonnancement des variables des BDDs. À cet effet, nous avons proposé deux différentes formes d'hybridation basées sur un noyau évolutionnaire quantique: l'approche QGABDD et l'approche QDEBDD.

Récemment, le problème de satisfiabilité SAT est de plus en plus utilisé dans le domaine de la vérification. La présente recherche visait à proposer de nouvelles approches efficaces pour résoudre le problème de satisfiabilité SAT. Pour cela plusieurs approches ont été proposées à cet effet. La plus compétitive méthode développée est sans aucun doute la méthode QGADPLL basée sur une hybridation entre un algorithme quantique génétique est une méthode exacte. L'idée est d'intégrer le pouvoir d'intensification de la méthode exacte DPLL au sein de l'algorithme quantique génétique qui possède des facultés de diversification

et ce, afin d'obtenir un bon compromis entre ces deux stratégies fondamentales de recherche et sans augmenter grandement le temps de calcul.

À l'exception de l'approche ClonSat basée sur un système immunitaire artificiel, les autres approches développées au cours de cette thèse sont basées sur un noyau quantique évolutionnaire développé spécifiquement pour les problèmes traités. En effet, nous avons opté une représentation quantique pour coder les solutions potentielles. L'avantage de cette représentation est la possibilité de représenter toutes les solutions possibles en utilisant seulement un seul chromosome grâce à la nature probabiliste de la représentation quantique. Ce qui réduit donc la taille de la population de chromosomes utilisée par rapport aux algorithmes évolutionnaires classiques. L'autre caractéristique des approches développées est l'utilisation d'une dynamique évolutionnaire quantique permettant une stratégie efficace pour la recherche de la bonne solution. En effet, cette dynamique permet un meilleur équilibre entre la capacité d'exploration et d'exploitation de l'espace de recherche afin d'avoir des solutions de bonnes qualités. Cependant, les approches développées se différent dans plusieurs détails à savoir le type de l'hybridation utilisée.

On peut également conclure, à la lumière des résultats présentés, que ce sont les hybridations effectuant des intensifications sur une solution, telles que les méthodes de recherche locales qui produisent les meilleurs résultats.

Perspectives

Beaucoup d'améliorations peuvent encore être apportées à nos travaux. Dans le cas du problème de BDD, deux grandes améliorations peuvent être effectuées. La première concerne la solution initiale, on envisage de créer une procédure basée sur la méthode de recherche en profondeur (Depth-First Search) pour générer une population avec de bons individus. La deuxième amélioration consiste à intégrer dans le processus d'optimisation une méthode de recherche plus performante afin d'accélérer le processus de convergence. Pour le cas des problèmes du SAT. On envisage d'essayer d'autre mode de coopération entre les méthodes complètes et incomplètes. Notre future contribution concerne également l'implémentation parallèle de ces approches proposées. Le parallélisme peut se faire au niveau de la recherche locale ou de la fonction objectif. Cela nous permet d'accélérer l'évaluation des individus et trouver les solutions dans un temps raisonnable.

Cependant, notre grande perspective est d'intégrer ces approches dans un modèle-checking à la base de BDD et de Solveur SAT.

Bibliographie

[Laguna, 2002] Laguna M.: Scatter Search In Handbook of Applied Optimization, *P. M. Pardalos and M. G. C. Resende (Eds.), Oxford University Press*, pp. 183-193 (2002).

[Aarts et al, 1989] E.H.L. Aarts and J. Korst. Simulated annealing and boltzmann machines: a stochastic approach to combinatorial and neural computing. *Wiley edition ,* Chichester, 1989.

[Aarts et al, 1997] E.H.L. Aarts and J.K Lenstra. Local Search in Combinatorial Optimization. *Discrete Mathematics and Optimization. Wiley-Interscience*, Chichester, England, June 1997.

[Akers, 1978]. Akers, S.B. Binary decision diagrams. *IEEE Trans. on Comp.*, 27:509–516, 1978.

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

[Asahiro et al, 1996]. AsahiroY., Iwama K., and Miyano E. Random Generation of Test Instanzes with Controlled Attributes. *In D.S.Johnson and M.A.Trick, editors, Cliques, Coloring, and Satisfiability: The Second DIMACS Implementation Challenge. Vol. 26 of DIMACS Series on Discr. Math. and Theor. Comp. Sci.* pages 377-394, 1996. http://www.cs.ubc.ca/~hoos/SATLIB/Benchmarks/SAT/D

[Baeck et al, 1995] Baeck T., and Schwefel H-P. Evolution strategies i: Variants and their computational implementation. *In Genetic Algorithms in Engineering and Computer Science, Proc*. First Short Course EUROGEN-95.1995

[Baeck et al., 1997] Baeck T., Fogel D.B., and Michalewicz Z. Handbook of Evolutionary Computation. *Institute of Physics Publishing and Oxford University Press*, 1997.

[Baeck et al., 2000] Baeck T., Fogel D.B., and Michalewicz Z., editors. Evolutionary Computation: Advanced Algorithms and Operators. *Institute of Physics Publishing*, Bristol, 2000.

[Bailleux, 1996] Bailleux O. Contribution à l"étude des paysages de recherche locale associe es au problème SAT. *Thèse de doctorat, Université de Bourgogne.* janvier 1996.

[Banzhaf et al,1999] Banzhaf W., Colin R. *Foundations of Genetic Algorithms* – 1999.

[Barichard, 2003] Barichard, V. Approches hybrides pour les problèmes multiobjectifs, PHD thesis. Angers, 2003, France.

[Behrmann et al, 2001] Behrmann G., Fehnker T.S. Hune A., Larsen K.G., Pettersson P. ,and Romijn J:Efficient Guiding Towards Cost-Optimality in UPPAAL, *Proc. TACAS 2001*, 174- 188, LNCS 2031, Springer, 2001.

[Bendiab et al, 2003] Bendiab E.,Meshoul S. , and Batouche M., An AIS for Multi-Modality Image Alignment, Second International Conference on Artificial Immune systems, ICARIS 2003, Edinburgh, UK, September 1-3, pp. 13-21,2003. Proceedings, Lecture Notes in Computer Science, LNCS 2787, Springer-Verlag Berlin Heidelberg, pp. 0302-9743, 2003

[Berman,1989] Berman C. L. Ordered Binary Decision Diagrams and Circuit Structure Extended *Abstract IEEE Conference on Computer Design*, Oct 1999.

[Biere et al, 1999] Biere A. , A. Cimatti, E. Clarke, and Y. Zhu. Symbolic model checking without BDDs. In Proc. of the Workshop on Tools and Algorithms for the Construction and Analysis of Systems (TACAS"99), LNCS. Springer-Verlag, 1999.

[Biron et al, 98]: D. Biron, O. Biham, E. Biham, M. Grassl and D A. Lidar. Generalized Grover Search Algorithm for Arbitrary Initial Amplitude Distribution. *Los Alamos Physics Preprint Archive, /quant-ph/9801066*, 1998.

[Blais et al, 00]: A. Blais, A.M. Zagoskin. Operation of universal gates in a solid-state quantum computer based on clean Josephson junctions between d-wave superconductors. *Phys. Rev. A*, 61:042308, 2000.

[Blais, 02]: A. Blais. Algorithmes et architectures pour ordinateurs quantiques supraconducteurs. *Thèse de doctorat, faculté des sciences, université de Sherbrooke*, canada, 2002.

[Bollig et al.,1996] Bollig B. and Wegener I. Improving the variable ordering of OBDDs is NPcomplete. *IEEE Trans. on Comp.*, 45(9):993–1002, 1996.

[Bontoux et al., 2008] Bontoux B., Artigues C., and Feillet D. A Memetic Algorithmwith a Large Neighborhood Crossover Operator for the Generalized Traveling Salesman Problem. University of Technology of Troyes, France. Metaheuristics for Logistics and Vehicle Routing, EU/ME, the European Chapter on Metaheuristics,2 008.

[Boros et al., 1990] Boros E., Crama Y., Hammer P., et Saks, M. A complexity index of satisfiability problems. *Annals of Mathematics and Artificial Intelligence*, 1 :21–32,1990.

[Boughaci et al, 2005] D. Boughaci, Habiba Drias, B. Benhamou, Combining a Unit Propagation with Genetic Algorithms to Solve max-SAT", in: *Workshop of Combination of Metaheuristic and Local Search with Constraint Programming (MLS+CP)*, University of Nantes, France., November 2005.

[Brassard, 96] Brassard G. New Trends in Quantum Computing. *arxive.org, quant-ph/9602014 19*, 1996.

[Brucker 1998] Brucker P., Knust S., Schoo A. and Thiele O., A branch & bound algorithm for the resource-constrained project scheduling problem. European Journal of Operational Research,107 :272–288, 1998.

[Bruni, 2004] Bruni R. Brchaff : a chaff-like dpll solver using ras heuristic. *In Proc. of the SAT 2004 Competition : Solver Descriptions*, Vancouver, Canada, may 2004.

[Bryant, 1988] Bryant R. E. On the Complexity of VLSI implementation and graph representations of Boolean functions with application to integer multiplication, CMU, Rapport de Recherche, Septembre 1988.

[Bryant, 1992]. Bryant R. E. Symbolic Boolean Manipulation with Ordered Binary Decision Diagrams, ACM Computing Surveys, Vol. 24, No. 3, pp. 293-318, September, 1992.

[Burch et al,1992] J. R. Burch, E. M. Clarke, K. L. McMillan, D. L. Dill, L. J. Hwang, Symbolic Model Checking: 1020 states and beyond, *Information and Computation*, vol. 98, no. 2, June 1992, pp. 142-70.

[Cahon et al., 2004] Cahon, S., Melab, N., and Talbi, E. G.. ParadisEO: A framework for the reusable design of parallel and distributed metaheuristics. *Journal of Heuristics*, 10(3):357–380. (2004)

[Chandru et al, 1999] Chandra V. and Rao M.R. Linear Programming, *Chapter 31 in Algorithms and Theory of Computation Handbook*, edited by M.J.Atallah, CRC Press 1999.

[Charbonneau, 2002] Charbonneau P. An introduction to genetic algorithms for numerical optimization. *NCAR Technical Note 450+IA (Boulder: National Center for Atmospheric Research)*, Etats-Unis, 2002.

[Chelouah et al, 2000] Chelouah, R. and Siarry P. Tabu Search Applied to Global Optimization. *European Journal of Operational Research*, vol 123, pp. 256-270, 2000.

[Clarke et al,1994] Clarke E. M., Grumberg O., and Long, D. E. Model checking and abstraction.ACM Transactions on Programming Languages and Systems, 16(5):1512–1542, septembre 1994.

[Cook, 1971] Stephen A. Cook. The complexity of theorem proving procedures. In 3rd ACM symp. on Theory of Computing, pages 151–158, Ohio, 1971.

[Cordeau et al, 2004] Cordeau, J. F., and Laporte, G. Metaheuristic Optimization via Memory and Evolution : Tabu Search and Scatter Search, *Chapter Tabu Search Heuristics for the Vehicle Routing Problem*, 145–163. Kluwer, Boston,2004.

[Corrêa, 1997] R. Corrêa, Recherche Arborescente Parallèle : de la Formulation Algorithmique aux Applications', *PhD Thesis, Institut National Polytechnique de Grenoble*, France, 1997.

[Costa et al, 2000] Costa U. S., Déharbe D. , and Moreira A. M.. Variable ordering of bdds with parallel genetic algorithms. *In Proceedings of PDPTA'2000*, 2000.

[Dasgupta et al, 2002]Dasgupta D, Gonzalez F: An Immunity-Based Technique to Characterize Intrusions in Computer Networks, IEEE Trans. Evol. Comput. Vol 6; 3, pp 1081-1088, 2002.

[Davis et al, 1960] Davis M., and Putnam H. A computing procedure for quantification theory. *In Communications of the ACM*, pages 201–215, 1960.

[Davis et al., 1962] Davis M., Logemann G., and Loveland D. A machine program for theorem-proving. *Communications of the ACM*, 5(7) :394–397, Jul 1962.

[De Castro et al, 2002] De Castro L. N.,and Von Zuben F. J. Learning and Optimization Using the Clonal Selection Principle. IEEE Transactions on Evolutionary Computation, Special Issue on Artificial Immune Systems, vol. 6, n. 3, pp. 239-251, 2002

[De Castro, 2000] De Castro L. N: The clonal selection algorithm with engineering applications. In Proc. GECCO, Workshop on Artificial Immune Systems, pp. 36-37, 2000.

[de Garis et al, 2003] de Garis, H. and Zhang Z.: "Quantum Computation vs. Evolutionary Computation: Could Evolutionary Computation Become Obsolete?", *Congress on Evolutionary Computation (CEC2003),* IEEE, 2003.

[De Jong, 2006] De Jong K.A. Evolutionary computation: a unified approach. *MIT Press*, Cambridge MA, 2006.

[Deutsch, 85] D. Deutsch. Quantum theory, the Church-Turing principle and the universal quantum computer. *Proceedings of the Royal Society of London Ser. AA400*, pp. 97:117, 1985.

[Devarenne, 2007] Devarenne I. Études en recherche locale adaptative pour l"optimisation combinatoire, *Thèse*

de doctorat, 30 novembre 2007.

[Dirac, 59]: P. Dirac, The Principles of Quantum Mechanics, *Oxford University Press*, (4th ed.), 1958.

[Doyle et al, 1995] Doyle S. A. and Dugan J.B. Dependability assessment using binary decision diagrams. *Proc. 25th International Symposium on Fault-Tolerant Computing*, pages 249{258,1995.

[Draa et al, 04]: Draa A., Talbi H., Batouche M.: A Quantum-Inspired Differential Evolution Algorithm for Rigid Image Registration. International Conference on Computational Intelligence, pp. 408-411, 2004.

[Draa et al, 10]: Draa A., Meshoul S., Talbi H., Batouche A Quantum-Inspired Differential Evolution Algorithm for Solving the N-Queens Problem. *Int. Arab J. Inf. Technol*. 7(1):pp. 21-27,2010.

[Drechsler et al,1998]. Drechsler R. And Becker, B. Binary Decision Diagrams: Theory and Implementation. *Kluwer Academic Publisher*, 1998.

[Drechsler et al.,1996] Drechsler R. et al. A genetic algorithm for variable ordering of OBDDs. IEEE Proc. Comp. Digital Techniques, 1996.

[Dréo et al. 2006] Dréo J., Pétrowski, A. Siarry P., and Taillard É. D., Metaheuristics for Hard Optimization: Methods and Case Studies, 3-540-23022-X, Springer, 2006.

[Dumas et al. 1986] Y. Dumas, J. Desrosiers, J. Soumis, "A dynamic programming solution of the large-scale single vehicle dial-ride problem with time windows". American Journal of Mathematical and Management Science 16, 301–325, 1986.

[Eiben et al, 2003] Eiben A.E., and Smith J.E. Introduction to Evolutionary Computing, *Springer*, 2003,

[Feoktistov,2006] Feoktistov V.: Differential Evolution: In Search of Solutions, *Springer Verlag*, 2006,

[Fonseca and Fleming, 1993] Fonseca, C. M. and Fleming, P. J. Genetic Algorithms for Multiobjective Optimization: Formulation, Discussion and Generalization. *In Proceedings of the Fifth International Conference on Genetic Algorithms*, pp. 416:423, San Mateo, California 1993.

[Friedman et al, 1987] Friedman S.J., Supowit K.J. Finding the Optimal Variable Ordering for Binary *Decision Diagrams Design Automation Conference*, June 1987.

[Fujii et al, 1993]. Fujii, H., Ootomo, G. and C. Hori. Interleaving based variable ordering methods for ordered binary decision diagrams. *In Int'l Conf. on CAD*, pages 38–41, 1993.

[Fujita et al, 1988]. Fujita, M., Fujisawa, H. and Kawato, N., Evaluation and improvements of Boolean comparison method based on binary decision diagrams. *In Int'l Conf. on CAD*, pages 2–5, 1988.

[Garey et al., 1979] Garey M. R., and Johnson D. S. Computers and Intractability: A guide to the Theory of NPCompleteness. *W.H. Freeman and Company*, New York, 1979.

[Garrett, 2005] Garrett S.: How Do We Evaluate Artificial Immune Systems? Evolutionary Computation, vol. 13, no. 2, pp. 145--178, 2005.

[Glover and Laguna, 1997] Fred Glover and Manuel Laguna. Tabu Search. *Kluwer Academic Publishers*, 1997.

[Glover, 1998] Glover, F. A : Template for Scatter Search and Path Relinking, in Artificial Evolution, *Lecture Notes in Computer Science 1363, J.-K. Hao, E. Lutton, E. Ronald , M. Schoenauer and D. Snyers (Eds.), Springer*, pp. 3-51.1998.

[Goldberg, 1989] Goldberg, D. E. Genetic algorithms in search, optimization, and machine learning. *Addison-Wesley*, 1989.

[Goldberg, 1994] Goldberg G.E., Algorithmes génétiques, *Éditions Addison-Wesley*, Paris, 1994.

[Grover, 96]: Grover L K., A fast quantum mechanical algorithm for database search. *The Twenty-Eighth Annual ACM Symposium on the Theory of Computing*, pp. 212–219, 1996.

[Guéret, 2000] Christelle Guéret, Christian Prins et Marc Sevaux, Programmation Linéaire, *Eyrolles*, 2000.

[Haijun et al, 2008] Haijun S, Yupu Y. Quantum-Inspired Differential Evolution for Binary Optimization Natural Computation, 2008. *ICNC '08.* Volume: 1, pp. 341-346 2008

[Hamdy, 2007] Hamdy A. T. Operations Research: An Introduction, *8th ed., Prentice Hall*, 2007.

[Han et al, 00] K. Han and J. KIM, Genetic Quantum Algorithm and its Application to Combinatorial Optimization Problem. *In Proceedings of the 2000 Congress on Evolutionary Computation, IEEE Press*, pp. 1354-1360, 2000.

[Han et al, 01] K. Han and J. KIM, Analysis of Quantum-inspired Evolutionary Algorithm. *In Proceedings of the 2001 International Conference on Artificial Intelligence, CSREA Press,* Vol. 2, pp. 727-730, June 2001.

[Haouari et al, 2003] Haouari L., and Ladhari,T. A branch-and-bound-based local search method for the flow shop problem. *The Journal of the Operational Research Society* 54(10), 1076–1084, 2003.

[Heudin, 1994] Heudin J-C, La Vie artificielle, *éditions Hermès Science*, Paris, 1994.

[Holland, 75] Holland J H. Adaptation in Natural and Artificial Systems, *Ann Arbor: The University of Michigan Press*, 1975.

[Hollander et al, 1999] Hollander, M., and Wolfe D. A. Nonparametric Statistical Methods. *Hoboken, NJ: John Wiley & Sons, Inc.,* 1999.

[Hoos et al, 2000] Hoos, H. and Stutzle, T. Local Search Algorithms for SAT: An Empirical Evaluation. *J. Autom. Reason.,* 24(4) :421–481, 2000.

[Hous et al, 2005] Hoos, H. and Stutzle, T. Stochastic Local Search: Foundations and Applications, *Morgan Kaufmann*, 2005.

[Huth et al ,2000] Huth M., and Ryan M. Logic in Computer Science: Modeling and Reasoning About Systems. *Cambridge University Press*, 2000.

[Ishiura et al, 1991] IshiuraN. , Sawada H., and Yajima S. Minimization of binary decision diagrams based on exchange of variables. *In Int'l Conf. on CAD*, pages 472–475, 1991.

[Jaeger, 2006] Jaeger G.: Quantum Information: An Overview. Berlin: Springer. ISBN 0-387-35725-4, 2006.

[Jerne, 1973] Jerne NK Towards a network theory of the immune system Annals of Immunology, vol. 125, no. C, pp. 373-389, 1973.

[Kim et al, 03] Kim K., Hwang j., Han k., and Kim J., A Quantum-inspired Evolutionary Computing Algorithm for Disk Allocation Method. *IEICE Transactions on Information and Systems, IEICE Press*, Vol. E86-D, No. 3, pp. 645-649, 2003.

[Kirkpatrick et al., 1983] Kirkpatrick, S., Gelatt, C. D., and Vecchi, M. P. Optimization by simulated annealing. *Science* 220, 671–680, 1983.

[Knuth, 2009] Knuth D. E. The Art of Computer Programming Volume 4, Fascicle 1: Bitwise tricks & techniques; Binary Decision Diagrams. *Addison-Wesley Professional* viii+260, March 27, 2009.

[Koza, 1999] Koza, J.R., Bennett, F.H., Andre, D., and Keane, M.A. Genetic Programming III: Darwinian Invention and Problem Solving, *Morgan Kaufmann*, 1999.

[Lardeux et al., 2005a] Lardeux F., Saubion F., and Hao J-K. GASAT : a genetic local search algorithm for the satisfiability problem. *Evolutionary Computation Journal*, 2005. to appear.

[Lardeux et al., 2005b] Lardeux F., Saubion F., and Hao J-K. Three truth values for the SAT and MAX-SAT problems. In Proc. of the Nineteenth International Joint Conference on Artificial Intelligence (IJCAI‟05). pages 187–192, Edinburgh, Scotland, aug 2005.

[Lardeux et al., 2006] Lardeux F., Saubion F., and Hao J-K. Une étude empirique des heuristiques de branchement pour le problème MAX-SAT, 2006.

[Lardeux, 2005] Lardeux F. Approches hybrides pour les problèmes de satisfiabilité (SAT et MAX-SAT), *Thèse de doctorat*, 25 Novembre 2005.

[Larsen et al, 1999] Larsen K., Pearson J., Weise C., and Yi W.: Clock difference diagrams. *Nordic Journal of Computing* 6, 271-298, 1999.

[Lassaigne et al,1996] Lassaigne R., and de Rougemont M. Logique et complexité. *Hermes*. 1996.

[Layeb et al, 2008f] Layeb A., Saidouni D.E, Talbi E-G.: A New Quantum Scatter Search Algorithm for MAX 3-SAT Problem. In the proc. of international workshop of Complexity and Intelligence of the Artificial and Natural Complex Systems. CANS08, *Petru Maior University Press, ISSN 2065-0426, pp.115-120.* Romania November 2008.

[Layeb et al,2006] Layeb A. Meshoul S., and Batouche M. Multiple Sequence Alignment by Quantum Genetic Algorithm. *In The 7th International Workshop on Parallel and Distributed Scientific and Engineering Computing of the 20th International Parallel and Distributed Processing Symposium*, ISBN: 1-4244-0054-6, pp 1 – 8, Greece, April 2006.

[Layeb et al,2007] Layeb A., Saidouni D.E.: Quantum Genetic Algorithm for Binary Decision Diagram Ordering Problem. *In the International Journal of Computer Science and Network Security*, Vol.7, No 9, pp. 130-135, ISSN 1738-7906, September 2007.

[Layeb et al, 2010a] Layeb A., Saidouni D.E.: A Hybrid Quantum Genetic Algorithm and Local Search Based DPLL for Max 3-SAT Problems. Accepted in the UKSim 12th International Conference on Computer Modelling and Simulation, England, 24 – 26 March 2010.

[Layeb et al, 2010b] Layeb A., Deneche A., and Meshoul M.: A New Artificial Immune System for Solving the Maximum Satisfiability Problem.. *In: N. García-Pedrajas et al. (Eds.): IEA/AIE 2010*, part II, Springer LNAI 6097, pp. 136-142, Spain, 2010.

[Layeb et al,2008a] Layeb A., Saidouni D.E.: Quantum Differential Evolution Algorithm for Variable Ordering Problem of Binary Decision Diagram. *In the Proceedings of the 13th International CSI Computer Conference CSICC'08, Springer: Communications in Computer and Information Science*, CCIS Vol. 6, pp. 942–945, Iran Mars 2008.

[Layeb et al, 2008b] Layeb A., Meshoul S., and Batouche M. , Quantum Genetic Algorithm for Multiple RNA Structural Alignment. *in the IEEE proceedings of the 2nd Asia International Conference on Modelling & Simulation (AMS 2008),* ISBN 978-0-7695-3136-6, pp 873-877, Kuala Lumpur, Malaysia 13 - 15 May 2008.

[Layeb et al, 2008c] Layeb A., Saidouni D.E.: A New Quantum Evolutionary Local Search Algorithm for Max 3-SAT Problem. *In the 3rd International Workshop on Hybrid Artificial Intelligence Systems. LNCS/LNAI 5271*, ISSN: 0302-9743, ISBN: 978-3-540-87655-7. pp. 172–179, 2008. Spain. September 2008

[Layeb et al, 2008d] Layeb A., Saidouni D.E.: Quantum differential evolution Algorithms Based on Hill Climbing for Solving BDD Ordering Problem. In the proc. of international workshop of Complexity and Intelligence of the Artificial and Natural Complex Systems. *CANS08, Petru Maior University Press*, ISSN 2065-0426, pp.121-124. Romania November 2008.

[Layeb et al, 2008e] Layeb A., Saidouni D.E. : Hill Climbing and Quantum Genetic Algorithms for MAX 3-SAT Problems. *In the Journal of Theoretical and Applied Information Technology*, ISSN 1992-8645, Vol4. No11.pp.1033-1039, November 2008.

[Layeb, et al, 2007] Layeb A. And Deneche A.: Multiple Sequence Alignment by Immune Artificial System. IEEE/ACS International Conference on Computer Systems and Applications (AICCSA 2007). ISBN: 1-4244-1031-2, pp. 336-342, Jordan, Mai 2007.

[Le Bellac ,03] Le Bellac M. : Introduction à l"informatique quantique, *Cours donné à l'Ecole Supérieure de Sciences Informatiques (ESSI)*, Octobre 2003.

[Le Berre, 2000] Le Berre D. : Autour de SAT : le calcul d"impliquants P-restreints, algorithmes et applications, thèse de doctorat, L"université Toulouse,2000.

[Lind-Nielsen, 1999] Jørn Lind-Nielsen BuDDy - A Binary Decision Diagram Package. Department *of Information Technology, Technical University of Denmark*, 1999.

[Malik et al,1988] Malik S., Wang A.R., Brayton R.K., and A. Sangiovanni-Vincentelli.: Logic Verification using Binary Decision Diagrams. *In a Logic Synthesis Environment ICCAD*. November 1988.

[Marchiori et al, 1999] Marchiori E., and Rossi C.: A flipping genetic algorithm for hard 3-SAT problems. *In Proc. of the Genetic and Evolutionary Computation Conference*, volume 1, pages 393–400, 1999.

[Mautor 1997] Mautor T. et Michelon P., Mimausa : A new hybrid method combining exact solution and local search. In MIC"97, 2nd Metaheuristics International Conference, Sophia Antipolis, France, 1997.

[Mazure et al., 1998] Mazure B., Sais L., and Grègoire E.: Boosting complete techniques thanks to local search methods. *Annals of Mathematics and Artificial Intelligence*, 22 :319–331, 1998.

[McMillan ,1992] McMillan K.L.: Symbolic Model Checking: An Approach to the State Explosion Problem. *Ph.D Thesis*, School of Computer Science, Carnegie-Mellon University

[McMillan, 2003]McMillan K.L.: Interpolation and SAT-based Model Checking, *Computer Aided Verification (CAV03), Springer, LNCS*, Vol 2404, pages 27-39, July 2003.

[Mellab, 2005]: Melab N., Contributions à la résolution de problèmes d'optimisation combinatoire sur grilles de calcul, HDR, Lille France, 2005.

[Meshoul et al, 2005b]: Meshoul S., Layeb A., and Batouche M.: a Quantum Evolutionary Algorithm for effective Multiple Sequence Alignment. In the proc of 12th Portuguese conference on artificial intelligence (EPIA 2005 CMB workshop). *Lecture notes in artificial intelligence (LNAI)*, no 3808, pp.190-201, 2005.

[Meshoul et al,2005a] Meshoul S. , Deneche A., and Batouche M.: Combining an artificial immune system with a clustering method for effective pattern recognition, in proceedings of the international conference on machine intelligence, pp. 782-787, Tozeur – Tunisia, November 5-7, 2005.

[Michalewicz et al, 2000] Zbigniew M., and Fogel D-B.: editors. How to solve it: modern heuristics. *Springer-Verlag*, 2000.

[Minato, 1993] Minato S. Zero-suppressed BDDs for set manipulation in combinatorial problems. In Proceedings of the 30th international on Design automation conference, ACM Press. pages 272–277., 1993.

[Minato, 1996] Minato S.: Binary Decision Diagrams and Applications for VLSI CAD. *Kluwer Academic Publisher, 1996.*

[Moore, 95]: MOORE M.: Quantum-inspired computing, department of computer science, *old library*, university of Exeter, UK, 1995.

[Moscato et al, 2005] Moscato, P., and Cotta, C. A Gentle Introduction to Memetic Algorithms. *Operations*

Research & Management Science 57(2), 105–144, 2005.

[Narayan et al, 1996] NarayanA., Jain J. , Fujita M. , A. Sangiovanni-Vincentelli. Partitioned ROBDDs - A Compact, Canonical and Efficiently Manipulable Representation for Boolean Functions. *ICCAD* 1996

[Palpant, 2005] Palpant M. : Recherche exacte et approchée en optimisation combinatoire : schémas d"intégration et applications. *Thèse de Doctorat*, Université d"Avignon, 2005.

[Panda et al, 1995] Panda S., Somenzi F. Who are the Variables in your Neighborhood. *ICCAD* November 1995

[Papadimitriou et al., 1982] Papadimitriou C.H and Steiglitz K., Combinatorial Optimization: Algorithms and Complexity. Prentice-Hall, 1982.

[Paquette, 2005] Paquette S. : évaluation symbolique de systèmes probabilistes à espace d"états continu. *thèse (M.Sc.)* , Faculté des études supérieures université de LAVAL 2005.

[Pearl, 1984] Pearl J. Heuristics: Intelligent Search Strategies for Computer Problem Solving. *Addison-Wesley*, 1984.

[Pessan et al., 2007] Pessan, C., Bouquard, J.-L.and Néron,E. Genetic Branch-and-Bound or Exact Genetic Algorithm ?. *Artificial Evolution*, 136–147, 2007.

[PHQ,2009] physique quantique. http://www.physique quantique.wikibis.com/informatique_quantique.php, 2009.

[Pravossoudovitch, 2009] Pravossoudovitch S. Représentation et Synthèse des Systèmes Logiques.Ecole Polytechnique Universitaire De Montpellier, 2009.

[Preskill, 1998] J. Preskill. Quantum Information and Computation. *Lecture Notes for Physics,* 229, pp. 1:321, California institute of technology, 1998.

[Ratschek et al, 1995] Ratschek, H. and Rokne, J. G.: Interval Methods. *In Handbook of Global Optimization: Nonconvex Optimization and Its Applications* (Ed. R. Horst and P. M. Pardalos). Dordrecht, Netherlands: Kluwer, pp. 751-828, 1995.

[Reeves, 1995] Reeves, C.: Modern Heuristic Techniques for Combinatorial Problems. Advances topics in computer science. Mc Graw-Hill, 1995.

[Rieffel et al, 00]: E. Rieffel and W. Polak. An introduction to quantum computing for non-physicists. *arxive.org, quant-ph/9809016 v2*, 2000.

[Rodriguez-Tello et al, 2005] Rodriguez-Tello E., and LERIA J.K. Hao, Recherche Tabou Réactive pour le Problème de l'Arrangement Linéaire Minimum, *Proceedings of the ROADEF*, 2005.

[Rudell, 1993] Rudell R. Dynamic Variable Ordering for Ordered Binary Decision Diagrams. *ICCAD,* November 1993

[Rustichini 98] Rustichini A. Dynamic Programming Solution of Incentive Constrained Problems. *Journal of Economic Theory*, 78, 2, 329-354,1998.

[Sasao et al, 1996] Sasao T., Fujita, M., editors: Representations of Discrete Functions. *Kluwer Academic Publisher*, London 1996.

[Sbihi, 2003] Sbihi A. : Les méthodes hybrides en optimisation combinatoire : algorithmes exacts et heuristiques, *thèse de doctorat*, université de paris, le18 décembre 2003.

[Schrijver,1986] Schrijver,A. Theory of linear and integer programming. *Interscience series in discrete mathematics and optimization Wiley, 1986.*

[Schumacher, 1995] Schumacher B.: Quantum coding. *Physical Review A*. 51: 2738–2747, 1995.

[Selman et al., 1992] Selman, B., Levesque, H., et Mitchell, D. A new method for solving hard satisfiability problems. *In AAAI'92: Proceedings of the Tenth National Conference on Artificial Intelligence*, pages 459–465,1992.

[Selman et al., 1994] Selman, B., Kautz, H. A., et Cohen, B. : Noise Strategies for Improving Local Search. *In press, M., editor, Proceedings of the 12th National Conference on Artificial Intelligence AAAI'94, volume* 1, pages 337–343,1994.

[Selman et Kautz, 1993] Selman, B. et Kautz, H. A. An empirical study of greedy local search for satisfiability testing. *In Proceedings of the Eleventh National Conference on Artificial Intelligence(AAAI-93)*, Washington DC 1993.

[Shannon, 1940] Shannon C. E. A symbolic analysis of relay and switching circuits, *Massachusetts Institute of Technology, Dept. of Electrical Engineering*, 1940.

[Shor, 1994] Shor P.: Algorithms for quantum computation: Discrete log and factoring, *In Proceedings of the 35th Annual Symposium on Foundations of Computer Science*. pp. 124–134, 1994.

[Shannon, 1940] Shannon C. E. A symbolic analysis of relay and switching circuits, Massachusetts Institute of

Technology, Dept. of Electrical Engineering, 1940.

[Shor, 1994] Shor P.: Algorithms for quantum computation: Discrete log and factoring, In Proceedings of the 35th Annual Symposium on Foundations of Computer Science. pp. 124–134, 1994.

[Shor, 1998] Shor, P.: Quantum Computing. documenta mathematica, extra volume ICM –I-, pp. 467:484, 1998.

[Shor, 2004] Shor, P.: Progress in quantum computing, *Quantum Information Processing* 3, 5-13, 2004.

[Simon, 1994] Simon D. On the Power of Quantum Computation. *Proc. 35th Annual Symposium on Foundationsof Computer Science*, pp. 116-123, 1994.

[Somenzi, 2001]. Somenzi F. : Efficient Manipulation of Decision Diagrams, in International Journal on Software Tools for Technology Transfer(STTT), vol. 3, pp.171-181, 2001.

[Storn et al, 1997] Storn R and Price K.: Differential Evolution – A Simple and Efficient Heuristic for global Optimization over Continuous Spaces, 1997.

[Talbi et al, 06]: Talbi H., Draa A., Batouche M. A Novel Quantum-Inspired Evaluation Algorithm for Multi-Source Affine Image Registration. Int. Arab J. Inf. Technol. 3(1): pp.9-15,2006.

[Vose, 1998] Vose M. D.: The Simple Genetic Algorithm: Foundations and Theory. MIT Press, 1998.

[Whaley, 2005]. Whaley J.: JAVABDD, a Java Binary Decision Diagram Library, Stanford University http://javabdd.sourceforge.net/

[Wilhelm, 00] Wilhelm R.: Informatics: 10 Years Back - 10 Years Ahead. (Ed.), Springer, p. 341-355, ISBN 3-540-41635-8, 2000

[Williams et al, 1998]. Williams, C.P. and Clearwater, S.H. Explorations in quantum computing. Springer Verlag, Berlin, Germany 1998.