N° d'ordre : 02/2006-M/MT

REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE

MINISTÈRE DE L'ENSEIGNEMENT SUPÉRIEUR ET DE LA RECHERCHE SCIENTIFIQUE

UNIVERSITÉ DES SCIENCES ET DE LA TECHNOLOGIE HOUARI BOUMEDIENNE

FACULTÉ DES MATHÉMATIQUES

Mémoire présenté pour l'obtention du diplôme de Magister

en Mathématiques

Spécialité : Recherche Opérationnelle (Génie Mathématique)

Par

GUEHAM Assia

THÈME

Heuristiques pour les problèmes d'affectation sous contraintes par l'approche de la coloration des sommets d'un graphe.

Soutenu publiquement le 13 / 03 / 2006, **devant le jury composé de :**

M^elle. I. BOUCHEMAKH	**Maître de conférences**	**U.S.T.H.B.**	**Présidente.**
M^r. H. AIT HADDADENE	**Maître de conférences**	**U.S.T.H.B.**	**Directeur de thèse.**
M^r. S. BOUROUBI	**Maître de conférences**	**U.S.T.H.B.**	**Examinateur.**
M^r. D. CHAABANE	**Maître de conférences**	**U.S.T.H.B.**	**Examinateur.**
M^r. M. YAGOUNI	**Chargé de cours**	**U.S.T.H.B.**	**Examinateur.**

Table de matières

Chapitre 5 : Situations envisageables pour un problème de disposition

Remerciements

Mes premiers remerciements vont à Monsieur Hacéne AIT HADDADENE, mon directeur de thèse, Maître de Conférences, Directeur de recherche et Directeur du Laboratoire LAID3 à l'université des sciences et de la technologie Houari Boumediene (USTHB), de m'avoir encadré, pour l'intéressant sujet qu'il m'a proposé, qu'il trouve ici mes vifs et profonds respects, considérations, gratitude ainsi que reconnaissances.

Que Melle Isma BOUCHEMAKH, Maître de Conférences et Directrice de recherche à l'USTHB, trouve ici mes plus grands remerciements de m'avoir fait l'honneur de présider le jury de cette thèse, je lui suis très reconnaissante.

Que les Messieurs Sadek BOUROUBI, Djamel CHAABANE, Maîtres de conférences et Maîtres de recherche à l'USTHB et que Monsieur Mohamed YAGOUNI Maître assistant et chargé de cours à l'USTHB, trouvent mes plus vifs remerciements et ma sincère reconnaissance d'avoir accepté d'examiner ce travail et d'être dans le jury de cette thèse.

Mes grands remerciements vont à l'encontre de ma mère, mes soeurs et mes frères, pour leurs encouragements, leurs compréhensions et qu'Allah les aides et leurs faits à tous miséricorde.

Que Monsieur M. AOUANE trouve mes sincères remerciements et reconnaissances, qu'Allah le récompense et lui fait miséricorde.

Que tous ceux qui m'ont de prés ou de loin aidé et qui ne trouvent pas leur nom ici, qu'ils me pardonnent et trouvent toute ma reconnaissance et gratitude.

Introduction générale

Prendre une décision est une action de tous les jours, mais il arrive qu'une telle ou telle décision puisse engendrer d'énormes pertes ou de grands profits. Ainsi une étude scientifique devient par ce fait plus qu'indispensable, c'est ce que fait la théorie de la décision. Choisir est donc une action centrale dans cette théorie. Divers problèmes se présentent sous cette optique de choix, mais elle devient le centre de l'étude dans les problèmes d'affectation sous contraintes. « Rendre à César ce qui appartient à César » n'est souvent pas applicable dans les problèmes d'A. S. C qui consiste en l'affectation des objets, représentants les candidats en des positions représentants les postes, sous diverses contraintes souvent difficilement modélisables ; en effet il est même quasi-impossible d'avoir une affectation idéale répondant à toutes les contraintes. La NP- complétude de ces problèmes fait que la solution optimale est rarement atteinte voir quasi- impossible. C'est justement dans la classe de ces problèmes, les plus difficiles, que se trouve les problèmes traités dans ce mémoire «l*es problèmes d'affectation sous contraintes par l'approche de la coloration des sommets d'un graphe* » qui sont connus plus pour être des problèmes de choix. Or choisir telle ou telle décision revient tout simplement à choisir un sous-ensemble de sommets ou d'arêtes dans un graphe modélisant le problème. Ainsi la résolution de tels problèmes passe par l'étude des sous-ensembles remarquables de sommets ou d'arêtes d'un tel graphe. Le recours à des heuristiques[1] et des métaheuristiques pour tenter la résolution de tels problèmes se trouve donc justifiée. La modélisation adoptée dans ce mémoire est faite à travers la coloration du graphe modélisant les problèmes d'affectation sous contraintes. Ceci en assimilant les contraintes des problèmes d'affectation sous contraintes aux contraintes d'adjacence dans la coloration.

En effet, dans les problèmes d'affectation sous contraintes, une solution consiste en la recherche des affectations optimales, des éléments de deux ensembles E et F, représentant les objets réels des problèmes de façon bijective et formant les meilleurs couples (x, y) de sorte que $x \in E$ et $y \in F$. Autrement dit, les solutions dans les problèmes d'affectation sous

[1] Heuristique, discipline visant à dégager les règles de la recherche scientifique et de la découverte, en Mathématiques, elle vise essentiellement l'élaboration de méthodes rapides mais conjoncturelles.

contraintes sont les meilleurs[1] sous ensemble de $E \times F$. Il est évident que les problèmes de choix où encore d'investigation des parties d'un ensemble est un problème NP- complet, du fait que le nombre de ces parties est d'ordre 2^n ainsi quasiment tout algorithme déterminant tous les couples (quasiment toutes les solutions réalisables) s'exécute pour la plus pire des instances en un temps d'ordre 2^n . Cette NP- Complétude justifie la nécessité du recours aux méthodes heuristiques, énoncées dans le thème, pour tenter d'obtenir un meilleur sous ensemble de couples formant les affectations optimales ou quasi-optimales dans les problèmes d'affectation sous contraintes.

Un problème d'affectation sous contraintes peut être vu comme étant une correspondance (pas forcément une bijection) sous une famille de contraintes C, entre un ensemble X des objets à affecter, et d'un autre ensemble D des différentes affectations possibles. Un problème de cette classe est noté par le triplet (X, D, C), tel que à chaque objet Xi correspondra une affectation D(Xi) satisfaisant certaines contraintes C(Xi). Cependant, il y a des problèmes d'affectation sous contraintes qui peuvent s'exprimer de façon classique (programme linéaire) et d'autres non, ceci est dû entre autre aux relations entre les contraintes qui ne sont pas linéaire.

La résolution de ces problèmes consiste à affecter des valeurs aux variables, de telle sorte que le maximum de contraintes soit satisfait.

Notre motivation à ce travail se situe dans la même optique que celle qui a motivé les travaux de H. AIT HADDADENE et autres sur la coloration des sommets d'un graphe. L'utilité pratique de cet invariant, nombre chromatique, ne cesse d'augmenter et d'avoir des extensions pratiques (problèmes industriels, problèmes d'aménagement urbain, problèmes d'affectation sous contraintes), ceci nous a amené à approcher un des problèmes pratique, NP- Complet de l'optimisation combinatoire par l'approche de la coloration qui est le problème d'affectation sous contraintes.

Notre travail est donc, une contribution à la résolution des problèmes d'affectation sous contraintes. Cependant, notre recherche nous a permis de concevoir une procédure de coloration polynomiale que nous avons nommée $P^{\oplus}$. Pour quelques graphes ayant certaines

[1] Sous ensemble meilleur de $E \times F$ est celui qui réalise l'optimum ou le quasi-optimum recherché dans le problème d'A.S.C.

propriétés, notre procédure nous permet d'avoir une solution qui converge vers la solution optimale, en l'absence de ces propriétés, les solutions procurées sont d'assez bonnes qualités.

Nous avons structuré notre mémoire en cinq chapitres où :

Le premier chapitre est consacré essentiellement, aux définitions préliminaires et aux notions les plus utilisées. Nous exposerons, dans celui-ci, des rappels qui seront nécessaires pour la suite. Nous introduirons un certain nombre de notions fondamentales en optimisation combinatoire. On y trouvera en particulier les principaux concepts et résultats, de la théorie des graphes, des heuristiques et des métaheuristiques utilisés dans ce travail et enfin les éléments de bases de la théorie de la complexité algorithmique.

Dans le second chapitre, nous traiterons les principales méthodes heuristiques et métaheuristiques qui peuvent être adaptées à la résolution du problème abordé dans ce mémoire. Nous nous pencherons principalement sur les méthodes tabou, recuit simulé, les algorithmes génétiques et la méthode de colonie de fourmis.

Le troisième chapitre sera consacré à l'approche heuristique pour la résolution du problème de la coloration des sommets d'un graphe quelconque, donc les problèmes d'affectation sous contraintes. Deux résultats fondamentaux, seront présentés, un algorithme polynomial de coloration $\left(P^{\oplus}\right)$ et une métaheuristique de coloration tabou et la méthode de colonie de fourmis. Il est important de signaler à ce niveau la particularité de cette heuristique hybride, en terme de rapidité d'exécution, en l'absence des propriétés souhaitables des graphes modélisant les problèmes d'affectation sous contraintes.

Le quatrième chapitre est dédié à la caractérisation des graphes d'une classe que nous avons appelé G^*. Nous établirons une démonstration de la perfection des graphes G^*, nous donnerons ainsi une justification de la coloriabilité de G^* de façon polynomiale par $P^{\oplus}$ et nous étudierons les relations d'inclusion de cette classe de graphe parfait avec celles déjà existantes.

En fin, dans le cinquième chapitre nous illustrons ce travail par un exemple pratique envisageable des problèmes d'affectation sous contraintes, où nous aborderons des situations envisageables pour une modélisation et une résolution d'un problème de disposition et d'affectation des produits chimiques dans une banque de stockage d'un centre de recherche.

Une conclusion finale, nous permettra de synthétiser les principaux résultats de cette recherche et les perspectives potentielles de ce travail.

Chapitre 1

Définitions et généralités

Sommaire

Dans ce chapitre, nous présenterons les concepts de base des mathématiques de l'optimisation combinatoire, de la théorie des graphes et principalement ceux des graphes parfaits.

1. OPTIMISATION COMBINATOIRE

Avant de s'intéresser à notre problème, il est nécessaire de rappeler brièvement quelques notions fondamentales de l'optimisation combinatoire, indispensables pour la compréhension de la suite de cette étude. En premier lieu, il convient de donner certaines définitions relatives à la combinatoire.

1.1. Définitions préliminaires

La *Combinatoire* est une branche des mathématiques qui étudie les configurations d'éléments discrets, les opérations et/ou l'analyse à faire sur ces configurations. Cette branche englobe plusieurs disciplines parmi elles, nous citerons, la théorie des graphes, la combinatoire énumérative, les problèmes de dénombrement, la théorie polyédrale. Les frontières entre ces branches ne sont pas hermétiques, leurs différences sont visibles à travers leurs orientations méthodologiques [6], [16], [29].

Définition 1

Un problème est dit combinatoire lorsqu'il comprend un grand nombre de solutions admissibles parmi lesquelles on cherche une solution optimale ou proche de l'optimum.

L'*Optimisation Combinatoire* (O.C) est une branche des mathématiques où, il est question de choisir (rechercher) un élément, une configuration parmi un ensemble de même structure qui doit optimiser un certain objectif totalement fixé à priori. Cette branche de la recherche opérationnelle occupe une place prépondérante des mathématiques discrètes et de l'informatique. Son importance se justifie par le large éventail de problèmes d'intérêt pratique et opérationnel qu'elle englobe [29].

Cette branche de la combinatoire, l'optimisation combinatoire, s'intéresse à la recherche d'une solution optimale dans un ensemble de solutions réalisables de grande cardinalité. Autrement dit, minimiser ou maximiser une fonction, avec ou sans contraintes, sur un ensemble dénombrable. Pour ce faire il s'agira de développer des algorithmes qu'on voudrait polynomiaux. L'optimisation combinatoire se situe au carrefour de la Théorie des Graphes, de

la Programmation Mathématique et de l'Informatique théorique.

Les problèmes de l'optimisation combinatoire se caractérisent par une formulation facile et souvent par une résolution très ardue [29].

Définition 2

Un problème d'O.C consiste, étant donnés un ensemble fini de configurations S et une application f, en la recherche d'un élément s^* *tel que :*

$$f(s^*) = \min_{s \in S}(f(s)).$$

1.1.1. Configuration valide et complète

Définition 1

On appellera une configuration valide une solution d'un problème ne violant aucune contrainte de celui-ci, elle sera dite non valide dans le cas contraire.

Définition 2

Une configuration est dite complète si elle est valide, et elle sera dite partielle si elle est en cours de formation « prématurée ».

L'impossibilité d'énumérer et d'examiner exhaustivement toutes les solutions et souvent une situation que l'on rencontre lorsque l'on veut résoudre d'une manière exacte des problèmes de l'optimisation combinatoire. Ainsi s'impose, la nécessité de distinguer entre les algorithmes selon leurs performances à nous procurer des solutions. La tâche qui consiste à classer, comparer et retenir les meilleurs algorithmes selon des critères bien précis, a donné naissance à une élégante théorie : « la théorie de la complexité » des algorithmes. Cette dernière a permis d'élaborer et de développer de nouveaux moyens capables de venir à bout et d'élucider les difficultés auxquelles étaient confrontées d'une manière générale l'optimisation discrète et l'optimisation combinatoire en particulier.

1.2. Notions sur la théorie de la complexité

Avant d'aborder la théorie de la complexité des algorithmes, il y a lieu de donner quelques concepts et définitions de base indispensables pour la compréhension de cette théorie.

Définition 1

On appelle instance I d'un problème (P), un cas de celui-ci, une situation particulière, où est défini l'ensemble des paramètres, fixés à des valeurs données.

Définition 2

Un algorithme (A) de résolution d'un problème (P) donné, est une séquence finie d'opérations élémentaires, complémentaires, logiques et chronologiques transformant une chaîne de caractères représentant les données de n'importe quelle instance de (P) (Input), en une chaîne représentant sa solution (Output).

Définition 3

Un problème de décision est un problème posé sur un ensemble de concepts dont la solution est une réponse par oui ou non.

Définition 4

Un algorithme est dit efficace s'il résout n'importe quelle instance I *d'un problème (P) en un temps de calcul polynomial en la taille de ses données.*

La *théorie de la complexité* permet d'étudier de manière formelle la *difficulté* des problèmes en informatique. Dans cette théorie, un problème est formalisé de la manière suivante : un ensemble de données en entrée, et une question sur ces dernières (et éventuellement un calcul à effectuer), elle ne traite que des problèmes de décision.

Un des résultats forts de cette théorie est apparent à travers la catégorisation des problèmes en fonction des algorithmes qui les résolvent [35].

1.2.1. Classe P

La classe P contient tous les problèmes relativement faciles, c'est-à-dire ceux dont la résolution est faite par des algorithmes efficaces. Plus formellement, ce sont les problèmes pour lesquels on peut construire une machine déterministe (machine de Turing[1]) dont le temps d'exécution est de complexité polynomiale (le sigle P signifie « Polynomial time ») [35], [32].

1.2.2. Classe NP

Les problèmes de la classe NP sont ceux pour lesquels on peut construire une machine de Turing non déterministe dont le temps d'exécution est de complexité polynomiale. Le sigle

[1] La machine de Turing est composée d'une bande, supposée infinie, et d'une tête de lecture écriture qui se déplace sur la bande pour lire des données ou écrire, sur la bande.

NP provient de « Non determinist Polynomial time » (et non de “Non Polynomial”).

Il est important de remarquer à ce stade que la classe P est incluse dans la classe NP, on écrit $P \subseteq NP$, car si l’on peut construire une machine déterministe pour résoudre efficacement un problème qui est dans NP, alors on peut certainement construire une machine non déterministe qui résout aussi efficacement le même problème [35].

Dans la classe NP, une sous classe composée de problèmes ayant certaines propriétés et liés entre eux se caractérisent par une résolution difficile, il s’agit de la classe des problèmes NP-Complets.

1.2.3. Classe NP-complet

Un problème NP-complet possède la propriété que tout problème dans NP peut être transformé en celui-ci en temps polynomial.

Définition 5

La définition d’une application qui met en relation les instances de P1 avec celles de P2 : est appelée réduction de P1 en un problème P2, et on écrit $P_1 \alpha P_2$.

Cette réduction est qualifiée de polynomiale dès qu’elle s’opère en un temps polynomial.

Définition 6

Un problème (P) appartient à la classe des problèmes NP- Complets si on ne connaît aucun algorithme efficace pour sa résolution.

Ainsi, découle la conjecture stipulant que les problèmes NP- Complets ne sont pas résolubles en temps polynomial.

1.2.4. Classes de complexité

Les algorithmes usuels peuvent être classés en un certain nombre de grandes classes de complexité [20]:

- Les algorithmes sub-linéaires dont la complexité est en général en $O(\log n)$;
- Les algorithmes linéaires en complexité $O(n)$ et ceux en complexité en $O(n\log n)$ sont considérés comme rapides ;
- Les algorithmes polynomiaux en $O(n^k)$ pour $k \geq 3$ sont considérés comme lents, sans parler des algorithmes exponentiels (dont la complexité est supérieure à tout polynôme

en *n*) que l'on s'accorde à dire impraticables dès que la taille des données est supérieure à quelques dizaines d'unités.

1.3. Les méthodes de résolution

Les méthodes de résolution exactes ou approchées proposées au cours de ces dernières années sont nombreuses, et sont le reflet de l'éventail des méthodes dont on dispose pour traiter les problèmes d'optimisation combinatoire. Nous nous proposons d'en étudier les principales.

1.3.1. Les méthodes exactes

Ces méthodes sont ainsi qualifiées en raison des solutions exactes (optimales) qu'elles procurent. Elles nous permettent d'obtenir une solution optimale des instances des problèmes résolus. Leur principe général consiste en une énumération intelligente, et le plus efficacement possible, toutes les solutions du problème pour en extraire une solution optimale [36]. Parmi ces dernières, nous citerons :

1.3.1.1. La programmation linéaire

C'est une partie de la programmation mathématique, de recherche d'extremum liés d'une fonction linéaire, dite fonction objectif, sur un ensemble défini par des relations linéaires de type équation ou inéquation (contraintes linéaires). La procédure suivie à ce niveau s'articule autour de deux étapes essentielles, la première consiste en la formulation mathématique du problème. Cette modélisation consiste en la détermination des variables, de leur nature, et des contraintes du problème sous formes d'équations et/ou d'inéquations linéaire ; le modèle obtenu est dit programme linéaire. Dans la seconde étape, il s'agit d'appliquer la méthode de résolution du modèle.

Le premier algorithme décrit pour la résolution des programmes linéaires est la méthode du simplexe qui reste à ce jour la plus utilisée. Malgré son efficacité en pratique, l'algorithme s'avère non polynomial, pour certains cas. Cependant, la programmation linéaire est considérée comme un problème de la classe P, grâce à l'algorithme de KACHYIAN [36].

1.3.1.2. La programmation dynamique

L'origine des principales méthodes de la Programmation dynamique sont dues à Richard BELLMAN, et sont utilisées pour résoudre des problèmes, dont la solution optimale s'obtient successivement en démarrant d'une solution réalisable et ceci en se basant sur le principe

d'optimalité[1]. C'est une méthode séquentielle, elle permet d'optimiser une fonction séparable de plusieurs variables, liées par des contraintes sous formes d'équations ou d'inéquations. La décomposition et la *séquentialité* sont les principes de base de cette méthode. Il est nécessaire, pour pouvoir l'appliquer que le problème puisse être décomposé en étapes, ce qui induit un gain de temps appréciable. Elle se caractérise par l'application du principe déjà énoncé.

Cette approche a permis l'élaboration d'algorithmes de résolution pour un grand nombre de problèmes combinatoires.

1.3.1.3. La méthode des plans sécants

C'est une approche qui se base sur la recherche d'une restructuration de l'espace des solutions réalisables et ce en imposant quelques contraintes supplémentaires à l'espace originel. L'idée générale est que ces contraintes additionnelles coupent des portions de l'espace des solutions sans altérer ni exclure aucun point de l'espace des solutions réalisables [36].

1.3.1.4. Techniques Séparation et Evaluation « Branch and bound »

La technique branch and bound (SEP) est la première qui a été utilisée lors de l'exploration d'arbres de possibilités trop complexes pour être parcourus intégralement, sur le domaine S des solutions d'un problème d'optimisation combinatoire. Ce domaine est décomposé progressivement sous forme d'arborescence dont la racine serait l'ensemble S.

La recherche de la solution optimale consisterait alors en un parcours judicieux de cette arborescence [36].

La plupart des problèmes auxquels nous nous intéressons, sont dans la classe NP-Complet, c'est pourquoi on privilégie des heuristiques, et encore des métaheuristiques et ceci en raison de l'***explosion combinatoire***. Celles-ci permettent d'obtenir des solutions de bonne qualité, bien qu'elles ne soient pas nécessairement optimales.

Explosion combinatoire

Lors de la résolution de certains problèmes, le nombre de solutions réalisables possibles peut évoluer rapidement avec leur taille ce qui rend quasi-impossible l'exploration de l'ensemble de ces solutions en vue d'une optimisation, cette situation est qualifiée d'explosion combinatoire, en dépit de l'essor des moyens de calcul.

[1] Toute sous politique d'une politique optimale est optimale.

1.3.2. Les méthodes de résolution approchées

Du fait des résultats de la NP-Complétude de certains problèmes de l'optimisation combinatoire, ceux qui jouissent d'un grand intérêt à la fois théorique et pratique, il est peu probable, d'envisager leur résolution à l'aide de méthodes exactes et ce en raison de leur temps d'exécution qui évolue exponentiellement avec la taille des instances du problème en question.

La quasi-possibilité de trouver des algorithmes efficaces au sens complexité du terme, étant écartée, les chercheurs ont orienté leurs efforts vers l'élaboration des méthodes heuristiques, qui sont capables de fournir de « bonnes » solutions réalisables en un temps raisonnable, ainsi la raison d'être des heuristiques [31].

Définition 1.1

Une méthode approchée est une méthode de recherche des solutions de bonnes qualités (c'est à dire quasi-optimales) en un temps de calcul raisonnable, sans toutefois pouvoir en garantir ni l'optimalité, ni la réalisabilité, ni même (dans de nombreux cas) l'éloignement de la solution par rapport la plus proche solution réalisable ou optimale.

Dans les méthodes approchées nous distinguons deux types : les heuristiques et les métaheuristiques dont la principale différence réside dans le mode opératoire. C'est ainsi que les dernières ont un pouvoir plus générique (le degré de généricité) plus élevé car indépendante du problème, il convient alors de détailler leur définition :

Définition 1.2

Une heuristique est une méthode approchée dédiée spécifiquement à la résolution d'un problème donné et qui tente d'exploiter au mieux sa structure par des critères de décision déduits de la connaissance du problème à résoudre, dont la solution optimale n'est pas garantie.

Définition 1.3

Une métaheuristique est une méthode, ou plus précisément, un canevas de méthodes, pour résoudre de manière approchée tous les problèmes dont la solution optimale n'est pas garantie. Cependant, ces méthodes ne dépendent pas du type du problème que nous tentons de résoudre.

L'intérêt des heuristiques

Il ressort des ces définitions que les heuristiques sont fondamentalement des moyens de résolution spécifiques au problème donné. Les métaheuristiques, en revanche, peuvent davantage être interprétées comme des principes d'optimisation que comme des méthodes proprement dites. En ce sens, on peut établir un parallèle avec les langages et métalangages : les métaheuristiques présentent un degré supplémentaire d'abstraction par rapport aux heuristiques dans la mesure où leur instanciation, pour un problème donné, peut s'apparenter à une heuristique. Ainsi, il est courant de reconnaître une plus grande qualité de résultat aux métaheuristiques, au détriment, il est vrai, du temps de calcul [15].

Le but d'une heuristique, est de ne pas essayer toutes les combinaisons possibles avant de trouver celle qui répond au problème, afin de trouver une solution approchée convenable (qui peut être exacte dans certains cas) dans un temps raisonnable. C'est ainsi que les programmes de jeu d'échecs font appel de manière très fréquente à des heuristiques qui modélisent l'expérience d'un joueur. Certains logiciels antivirus se basent également sur des heuristiques pour reconnaître des virus non répertoriés dans leur base, en s'appuyant sur des ressemblances avec des virus connus.

Une heuristique est l'utilisation de *règles empiriques,*

- Pratiques, simples et rapides ;
- Facilitant la recherche des faits et l'analyse de situations ;
- Dans un objectif de résolution de problèmes et de *prise de décision* ;
- Dans un domaine particulier.

1.3.2.1. Heuristiques constructives

Elles consistent à construire une solution pas à pas, son objectif est d'obtenir une bonne solution en utilisant des principes souvent très simples.

Parmi celles-ci nous citerons :

a. les algorithmes gloutons

Ils sont caractérisés par le fait qu'il s'agit de procédure sans retour arrière : un choix fait à un moment n'est plus remis en question ultérieurement. Ces choix itératifs visent la construction de la solution réalisable. Sans doute, le principe des méthodes gloutonnes est le plus utilisé. Nombreuses méthodes exactes sont basées sur le principe glouton. Elles tirent parti du fait

que, dans un matroîde, toute heuristique gloutonne fournit une solution optimale [15].

b. Le principe de construction progressive

C'est une extension du principe glouton dans la mesure où l'on s'autorise, cette fois-ci, de modifier des valeurs déjà assignées. Ceci correspond à accepter une remontée dans l'arbre décisionnel. Les algorithmes de backtracking en sont des cas extrêmes puisqu'ils reposent sur un parcours exhaustif de l'arbre. Les heuristiques de construction de cycle hamiltonien pour le TSP, basées sur l'insertion itérative de nouveaux sommets sont à la fois des méthodes gloutonnes et à construction progressive selon qu'on se place sous l'angle des sommets insérés ou des arcs utilisés [15].

c. Le principe de partitionnement

Il reprend le principe réductionniste de "diviser pour régner" ; résoudre le problème global se révèle souvent plus complexe que résoudre la somme des sous problèmes qui le composent. Toute la difficulté réside alors dans la fusion des solutions de chaque sous problème (solutions approchées ou non et fusion exacte ou non) [15], [36].

Plusieurs méthodes exactes sont basées sur cette approche (la décomposition).

1.3.2.2. Heuristiques d'amélioration itérative (recherche locale)

Contrairement aux méthodes constructives dont l'objectif est de construire une solution, les méthodes d'amélioration modifient une solution initiale, en vue d'améliorer sa valeur. Cette solution initiale est souvent le résultat d'une méthode constructive ; un algorithme général sera composé de deux phases : une méthode constructive suivie d'une méthode d'amélioration.

La plupart de ces méthodes utilisent la notion de voisinage. Il s'agit de trouver, à chaque itération, une « bonne » solution parmi l'ensemble des solutions qui définissent un voisinage d'une solution courante. Parmi ces méthodes, nous distinguons celles qui améliorent, à chaque itération, la valeur de la fonction objectif, dites méthodes de descente, et celles qui permettent de choisir une solution qui n'améliore pas forcément la valeur de la fonction objectif. D'une manière générale, le principal inconvénient des méthodes de descente est qu'elles donnent souvent des solutions correspondant à des optima locaux qui ne sont pas de très bonne qualité, alors que, en choisissant une solution qui n'est pas de descente, on peut sortir des minima locaux [15].

Les métaheuristiques, quant à elles, reposent essentiellement sur le principe d'amélioration

itérative. Elles nécessitent donc la possession d'une solution initiale.

Dans ce qui suit, on présentera brièvement les méthodes métaheuristiques à savoir, la méthode de Recuit Simulé, la Recherche Tabou, les Algorithmes Génétiques et les algorithmes basés sur le principe des Fourmis qui explorent également un voisinage. Le principe de ces méthodes fait l'objet d'une étude plus détaillée dans le deuxième chapitre.

a. Le recuit simulé

Le recuit simulé s'inspire du processus de recuit physique. Le processus du recuit simulé répète une procédure itérative qui cherche des configurations de coût plus faible (dans un problème de minimisation) tout en acceptant de manière contrôlée des configurations qui dégradent la fonction coût [15], [5].

b. La méthode tabou

Cette méthode a été élaborée par GLOVER, elle est basée sur la notion de mouvements interdits (tabou). Chaque itération consiste à trouver le mouvement qui nous donne la meilleure solution dans le voisinage de la solution courante ; sachant que certains mouvements sont interdits. Parfois, nous choisissons une solution qui détériore légèrement la solution courante pour s'échapper des minima locaux. L'inverse du mouvement effectué à chaque itération est rajouté dans une liste, appelée liste tabou, qui contient les mouvements interdits. Initialement, la liste tabou est vide [22].

c. Les algorithmes génétiques

Ils sont proposés, pour la première fois, au milieu des années 70 par J. HOLLAND [24]. Ces algorithmes se proposent d'imiter la sélection naturelle et la génétique de la théorie de l'évolution. La terminologie de la génétique a été également empruntée pour la description de tels algorithmes.

Contrairement aux méthodes, Tabou et le Recuit Simulé qui manipulent une seule solution, les algorithmes génétiques considèrent un ensemble de solutions (individus) appelé population. Le but de la méthode est de faire évoluer la population en effectuant des mutations et des croisements suivis de sélection d'individus. La mutation est une opération unitaire qui modifié la structure d'un individu. Le croisement est une opération binaire, qui à partir de deux individus, en produit deux nouveaux.

L'idée de base de ces algorithmes réside dans le fait que travailler avec une population permet d'identifier et d'explorer les propriétés communes des bonnes solutions [5].

d. Optimisation par colonies de fourmis

La métaheuristique colonie de fourmis, ACO (Ant Colony Optimization), s'inspire du comportement collectif des colonies de fourmis pour résoudre des problèmes à base de graphes. L'optimisation par colonies de fourmis est basée sur le constat que, dans la nature, les fourmis (des animaux aveugles) sont capables d'établir par une somme d'interactions élémentaires le plus court chemin de leur nid à un objectif (une source de nourriture par exemple). Le but de cette approche est de trouver une solution annulant le coût et ceci en utilisant une multitude d'agents élémentaires (les fourmis) effectuant chacun de très simples actions [26].

Dressons à présent un récapitulatif des méthodes de résolution, existantes pour les problèmes d'optimisation [31].

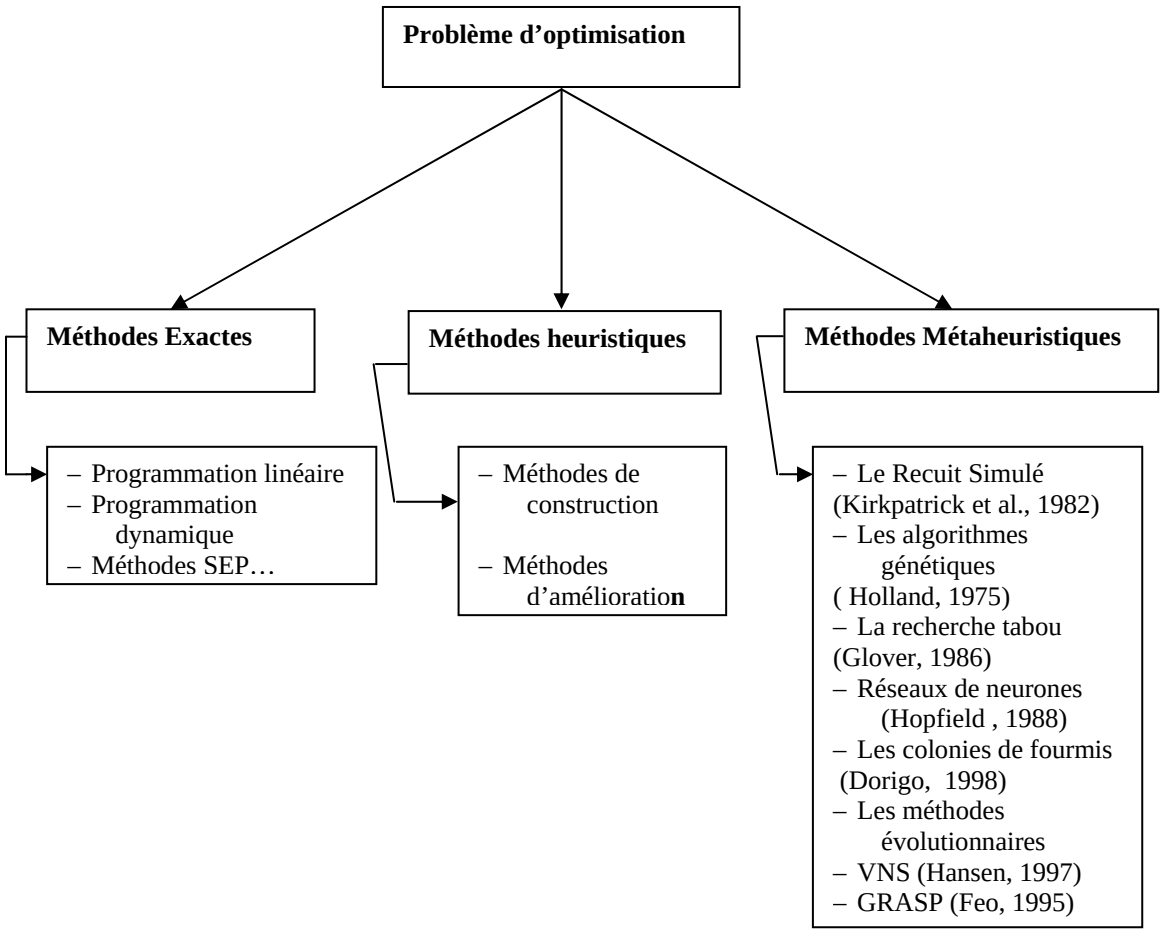


Figure. 3 : Les méthodes de résolution des problèmes d'optimisation

2. GENERALITES SUR LES CONCEPTS DE BASE DE LA THEORIE DES GRAPHES

Avant de décrire en détails le problème qui nous intéresse, nous faisons un bref détour en récapitulant quelques éléments de théorie des graphes.

2.1 Définitions de base

La théorie des graphes est utilisée dans un grand nombre de disciplines (mathématiques, physique, économie, etc.).

Un graphe G est la donnée de deux ensembles finis non vides E et V, où V représente l'ensemble des sommets noté V(G) et E l'ensemble des couples des éléments de V noté E(G) (couples de sommets) appelés arêtes. Le graphe ainsi obtenu se note $G=(V,\ E)$ [4], [32], [34].

Deux sommets u, v sont dits adjacents dans G s'ils se trouvent reliés par une arête, une arête est donc notée {u, v} où u et v sont appelées extrémités de cette arête.

Un graphe orienté est un graphe où tout couple sommets adjacents possède une extrémité initiale et une extrémité terminale et si {u, v} est orienté alors {u, v} est appelé arcs.

Un graphe simple est un graphe sans boucle (une arête dont les deux extrémités sont confondues) où tout couple de sommets est relié par au plus une seule arête

Deux arcs (arêtes) sont dits (es) adjacents (es) si elles ont au moins une extrémité en commun.

Un voisin d'un sommet v est un sommet qui est adjacent à v, on notera $N(u)=\{v\in V(G)/\,v$ est voisin de u dans $G\}=\{$ensemble des voisins de $u\}$.

Une chaîne simple de longueur k est une séquence $\{v_0,\ v_1,\ ...,\ v_k\}$ de sommets tels que v_i est adjacent à $v_{i+1},\ \forall i=0,\ 1,\ ...,\ k$ et $v_0,\ v_k$ sont les deux extrémités de la chaîne, elle est dite élémentaire s'elle n'utilise pas plus d'une fois le même sommet.

Un cycle est une chaîne simple dont les extrémités sont confondues, il est élémentaire s'il n'utilise pas plus d'une fois le même sommet.

Une corde est une arête qui relie deux sommets non consécutifs d'une chaîne ou d'un cycle.

Une *chaîne minimale* de longueur k est une séquence $\{v_0, v_1, ..., v_k\}$ de sommets distincts tels que v_i est adjacent à v_{i+1} $\forall i = \overline{0,k}$ sans corde et v_0, v_k représentent les deux extrémités de la chaîne, une telle chaîne est notée par P_k.

Un cycle sans corde ayant k sommets (de longueur k) est désigné par C_k. Un cycle élémentaire de longueur au moins quatre sans corde est appelé *trou* et un *anti-trou* est le complémentaire d'un trou.

Un sous graphe induit de $X \subseteq V(G)$ est le graphe $G(X) = (X, E_X)$ où $E_X = \{\{u,v\} \in E : u \in X, v \in X\}$.

Un graphe partiel est le graphe engendré par $E' \subseteq E$, il est noté $G = (V, E')$.

Un graphe complémentaire d'un graphe $G = (V, E)$ est le graphe $\overline{G} = (V, \overline{E})$ où $\{u,v\} \in \overline{E}$ si et seulement si $\{u,v\} \notin E$.

Un graphe est dit connexe si pour toute paire de sommets, il existe une chaîne joignant ces deux sommets.

Une composante connexe d'un graphe G est un sous graphe induit connexe maximal de G.

Un graphe simple est dit complet si tout sommet v est adjacent à tous les autres sommets du graphe (il existe une arête entre chaque paire de sommets).

Une clique d'un graphe G est un sous graphe complet de G.

Une clique d'un graphe G est dite maximale si elle est le plus grand sous graphe complet de G (grand en terme d'inclusion de sommets) et on note la cardinalité maximale d'une clique de G par le nombre ω(G).

Soit C une application, définie de $V(G)$ (resp. $E(G)$) dans *IR* qui associe à tout sommet v (resp. arête e) de G une valeur réelle $C(v)$ (resp. $C(e)$). C est appelée fonction poids et $C(v)$ (resp. $C(e)$) poids du sommet v (respectivement de l'arête e) de G. Le graphe obtenu est dit un graphe à sommets (resp. arêtes) *pondérés* (es).

2.2. Le problème de la coloration des sommets d'un graphe

a. Historique du problème de la coloration

Le plus ancien problème, dans ce sens, est le célèbre « Problème de la 4 – coloration, (the fourth map coloration) ». Autrement dit ; 4 couleurs sont-elles suffisantes pour colorier une carte géographique de telle sorte que deux pays voisins aient des couleurs distinctes ? Ce dernier s'est révélé très difficile et a suscité un très grand intérêt [34].

Considérons un graphe non orienté $G(V,\ E)$, où V l'ensemble de sommets et E l'ensemble des arêtes, reliant certaines paires de sommets.

Définition

Etant donné un graphe $G = (V,E)$, il s'agit d'associer à chaque sommet une couleur de telle sorte que :

Deux sommets adjacents aient des couleurs distinctes ;

Le nombre de couleurs utilisées soit minimum.

Ce nombre minimum de couleurs, $\gamma(G)$, est appelé « nombre chromatique » du graphe G. En d'autres termes, il s'agit de trouver le nombre γ minimum tel qu'il existe une application $c : V \to \{1,2,...,\gamma\}$ avec : $(v_1,v_2) \in E \mapsto c(v_1) \neq c(v_2)$.

Remarque

Les ensembles de sommets ayant la même couleur sont, par définition, des stables de G. Si on dresse la liste de tous les stables maximaux de G et si on considère la $m \times n$ – matrice A dont les lignes sont en bijection avec les sommets de G et les colonnes avec les stables et si on associe à chaque stable un coût égal à 1, on voit que la recherche d'une coloration des sommets d'un graphe avec un minimum de couleurs se formule comme un problème de recouvrement [33].

Notons, toutefois que cette construction n'est pas opérationnelle en général car le nombre des stables maximaux d'un graphe devient vite astronomique.

Ce problème étant NP-complet, nous aborderons dans le cadre de ce mémoire, sa résolution à l'aide des méthodes approchées.

2.3. L'intérêt du problème de la coloration

De nombreux problèmes peuvent être modélisés sous forme de problèmes de coloration de graphes, et ce dans divers domaines, il s'agit des problèmes se ramenant à la recherche d'une partition d'un ensemble d'objets en sous ensembles ne comportant que des éléments compatibles deux à deux. A titre d'exemple, la détermination d'un horaire scolaire; est une partition du temps en périodes durant lesquelles seuls des cours pouvant avoir lieu simultanément y sont donnés. Citons également comme illustration l'élimination des collisions dans des réseaux sans fils, le problème d'allocation des registres en informatique et le problème d'ordonnancement dont il existe souvent deux types de contraintes simultanément : des contraintes de précédence et des contraintes de disjonction [17]... etc.

3. LES GRAPHES PARFAITS

Historique

L'historique des graphes parfaits a commencé en 1961 lorsque Claude BERGE a eu l'intuition de ce qu'il appelait « ***la belle propriété*** » et qui porte aujourd'hui le nom de *perfection*.

Il introduisait les concepts de graphe $\gamma - parfait$ et $\alpha - parfait$. Il avait défini un graphe $\gamma - parfait$ (respectivement $\alpha - parfait$) comme suit : un graphe est $\gamma - parfait$ si et seulement si pour tout sous graphe induit H de G, le nombre chromatique $\gamma(H)$ est égal à $\omega(H)$, la taille de la plus grande clique de H (respectivement $\theta(H)$, la cardinalité minimum d'une couverture par des cliques de H, est égal à $\alpha(H)$, la taille du plus grand stable de H) [1].

L'intérêt de C. BERGE dans l'introduction de ces concepts, a son origine les travaux de C.E. SHANNON en théorie de l'information sur la capacité d'un canal de communication, à partir d'un ensemble de signaux que l'on peut émettre, on construit un graphe G dont l'ensemble des sommets est en bijection avec les différents signaux possibles. Dans le graphe G modélisant ce problème, deux sommets sont adjacents si et seulement si les signaux correspondant à ces sommets peuvent être confondus.

Un code est un ensemble de signaux qui ne peuvent être confondus. Le problème est de trouver le code le plus riche, le code transmettant un nombre maximum de signaux, ce qui revient à déterminer la cardinalité maximum $\alpha(G)$ d'un stable de G.

Après avoir défini ces deux concepts, BERGE s'intéressait à la recherche d'une caractérisation structurelle. Comme nous l'avons déjà signalé, les travaux de C.E. SHANNON ont incité BERGE à s'intéresser aux graphes parfaits. SHANNON avait remarqué que le nombre de stabilité d'un C_5 est 2 tandis que la taille minimale d'une partition en cliques de ce dernier est 3, et que C_5 est le plus petit graphe ayant cette propriété. Le complémentaire de C_5 est $\overline{C}_5$, on aura aussi que $\gamma\left(\overline{C}\right)=2$ et $\omega\left(\overline{C}_5\right)=3$. Plus généralement, pour les trous impairs, C_{2k+1} *pour* $k\geq 2$, et les anti-trous, $\overline{C}_{2k+1}$ *pour* $k\geq 2$, on a :

$$\gamma\left(C_{2k+1}\right)=3 \text{ et } \omega\left(C_{2k+1}\right)=2 \text{ ; } \omega\left(\overline{C}_{2k+1}\right)=k \text{ et } \gamma\left(\overline{C}_{2k+1}\right)=k+1 \text{ pour } k\geq 2.$$

C. BERGE avait proposé deux conjectures.

- **Première conjecture :** conjecture faible des graphes parfaits

  Un graphes est $\alpha-$ *parfait* si et seulement s'il est $\gamma-$ *parfait* .

- **Deuxième conjecture :** conjecture forte des graphes parfaits

  Un graphe est $\alpha-$ *parfait* (respectivement $\gamma-$ *parfait*) si et seulement si, ni lui ni son complémentaire ne contient un trou impair.

3.1. Quelques définitions complémentaires

3.1.1. Graphe imparfait critique

Un graphe imparfait critique est un graphe non parfait dont tout sous graphe propre est parfait.

3.1.2. Graphe de BERGE

Un graphe de BERGE est un graphe ne contenant pas comme sous graphe induit de trou ni d'anti-trou impair.

Théorème des graphes parfaits

Un graphe est parfait si et seulement si son complémentaire est parfait.

LOVASZ introduisait la deuxième conjecture forte des graphes parfaits : G parfait si et seulement si G est de BERGE.

Citons maintenant quelques classes des graphes parfaits pour lesquelles le problème de la coloration est résolu d'une manière exacte par un algorithme polynomial.

3.2. Les classes des graphes parfaits [1], [30]

Le but de cette partie n'est pas de faire un inventaire complet de ces classes, mais plutôt d'en présenter quelques unes ; en particulier, celles qui feront l'objet d'analyse dans notre étude. Ces classes présentent un intérêt soit d'un point de vue historique, algorithmique ou par leur relation avec des techniques particulières.

3.2.1. Les graphes de BERGE

3.2.1.1. Les graphes parfaits BERGE implicite

a. Les graphes bipartis

Un graphe G est dit biparti si l'ensemble de ses sommets peut être partitionné en deux sous ensembles V_1 et V_2 tels que toute arête de G relie un sommet de V_1 à un sommet de V_2. Autrement dit : l'ensemble des sommets de G admet une partition en deux stables.

Si G contient toutes les arêtes reliant tous les sommets de V_1 à tous les sommets de V_2, alors le graphe G sera dit ***biparti complet***. Dans ce cas, si V_1 contient *p* sommets et V_2 en contient *q*, alors G est noté $K_{p,\,q}$. ***Le k-parti*** (multiparti) complet noté $K_{p_1,\,\ldots,\,p_k}$ a un ensemble de sommets qui peut être partitionné en k parties disjointes V_1, ..., V_k telles que V_i possède P_i sommets, $i = 1, \ldots, k,$ et deux sommets sont adjacents si et seulement s'ils appartiennent à deux parties distinctes.

Théorème 1 [1]

Pour un graphe G, les conditions suivantes sont équivalentes :

1) G est biparti ;
2) $\gamma(G) = 2$;
3) G n'admet pas de cycle impair.

Théorème 2 [1]

Tout graphe biparti est parfait.

b. Graphes de parité

Un graphe *G* est dit de *parité* si, et seulement si, pour toute paire *x*, *y* de sommets de *G*, toutes les chaînes sans corde joignant *x* et *y* ont la même parité. BURLET et UHRY ont montré que cette définition est équivalente à celle selon laquelle tout cycle impair contient au moins deux

cordes croisées (où deux cordes $ab,\ cd \in E$ sont dites *croisées* si, et seulement si, les quatre sommets *a*, *b*, *c* et *d* sont distincts et exactement un des sommets *c* et *d* apparaît sur chaque arc déterminé par *a*, *b* sur le cycle). La preuve de la perfection de cette classe est due à OLARU et SACHS. Le problème de la reconnaissance, ainsi que les problèmes classiques d'optimisation, sont résolus par BURLET et UHRY.

c. Graphes triangulés

Un graphe est dit *triangulé* s'il ne possède pas de cycle induit de longueur supérieure ou égale à 4 sans corde. La perfection de cette classe a été établie par BERGE ainsi que par HAJNAL et SURANYI. Ces deux derniers ont montré que le nombre de stabilité d'un graphe triangulé est égal à la taille minimale d'une partition en cliques. BERGE remarqua que cela signifie que les complémentaires des graphes triangulés sont $\gamma - parfaits$. Ensuite, C. BERGE démontra que dans un graphe triangulé, la taille maximale d'une clique est égale au nombre chromatique, comme tout sous graphe induit d'un graphe triangulé est triangulé.

Une autre voie consiste à imposer des conditions sur les cycles impairs, par exemple sur leurs cordes éventuelles.

d. Graphes i-triangulés

Un graphe est dit *i-triangulé* si tout cycle impair contient au moins deux cordes non croisées. La preuve de la perfection de cette classe est due à GALLAI ainsi qu'à SURANYI. Les graphes i-triagulés contiennent la classe des graphes triangulés, ainsi que les graphes bipartis.

Un algorithme polynomial de reconnaissance a été construit par BURLET et FONLUPT. Ces deux classes ont été généralisées par MEYNIEL.

e. Graphes de MEYNIEL

Un graphe *G* est dit de *MEYNIEL* si tout cycle impair de *G* possède deux cordes. La perfection de cette classe fut établie par MEYNIEL en utilisant l'échange bi-chromatique, ainsi que par MARKOSIAN et KARAPETIAN. Une caractérisation, ainsi qu'un algorithme de reconnaissance, sont dus à BURLET et FONLUPT. Le problème de la coloration est également résolu pour ces graphes.

Les conditions qui font intervenir deux cordes étant épuisées, tout en ayant fourni de bons résultats, il est naturel de considérer les conditions relatives à une seule corde. Cette classe généralise celle de graphes de parité ainsi que celle des graphes i-triangulés et les graphes de parité sont de MEYNIEL.

f. Les graphes de comparabilité

Un graphe G = (V, E) est dit de comparabilité s'il est possible d'orienter ces arêtes de façon transitive autrement dit : $\left.\begin{matrix}(x,\ y)\in U\\(y,\ z)\in U\end{matrix}\right\}\Rightarrow(x,\ z)\in U\ \ (transitive)$

$(x,\ y)\in U\Rightarrow(y,\ x)\notin U\ \ (antisymétrique)$.

Théorème [1]

Les graphes de comparabilité sont parfaits.

g. Les graphes d'intervalles

Soit $I=\{I_1,\ ...,\ I_n\}$ une famille d'intervalles sur une droite. On peut former un graphe G, dont les sommets $v_1, ..., v_n$ représentent respectivement les intervalles $I_1, ..., I_n$; deux sommets étant joints si les intervalles correspondants s'intersectent. Un tel graphe est dit représentatif de la famille I ou graphe d'intervalles.

Théorème 1 [1]

Les graphes d'intervalles sont parfaits.

Théorème 2 [1]

Un graphe G est d'intervalles si et seulement si G est triangulé et $\overline{G}$ est un graphe de comparabilité.

Une autre classe intéressante, contenant également les graphes triangulés, il s'agit de la classe des graphes faiblement triangulés, introduite par HAYWARD.

h. Graphes faiblement triangulés

Un graphe *G* est dit *faiblement triangulé* si ni *G*, ni $\overline{G}$, ne contiennent des cycles induits de longueur au moins 5. HAYWARD a montré que ces graphes sont parfaits et a donné un algorithme polynomial pour les reconnaître. Les problèmes d'optimisation dans cette classe sont résolus par HAYWARD, HOANG et MAFFRAY ainsi que par SPINRAD et SRITHARAN.

i. Graphes légèrement triangulés

Un graphe est dit *légèrement triangulé* s'il ne contient pas de cycles induits de longueur au moins 5 et si tout sous graphe contient un sommet dont le voisinage est une chaîne minimale P_4. Ces graphes furent introduits par MAIRE qui prouva leur perfection (ce sont des graphes de BERGE).

3.2.1.2. Les graphes parfaits BERGE explicites

a. Les graphes sans griffe

Une griffe est un K_{1-3}. Un graphe sans griffe est un graphe qui n'admet pas de K_{1-3} comme un sous graphe induit.

Théorème [1]

Tout graphe de BERGE sans griffe est parfait.

b. Les graphes sans diamant (ou K_4 / e)

Un graphe sans diamant est un graphe ne contenant pas de sous graphe isomorphe à K_4 / e (figure 1).

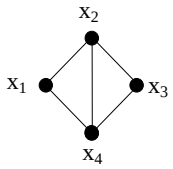


Figure. 1 : Un K_4 / e.

Théorème [1]

Tout graphe de BERGE sans K_4 / e est parfait.

Une première preuve a été proposée par K.R. PARTHASARATHY, G. RAVINDRA en 1979 mais malheureusement, elle a été remise en cause par A.TUCKER qui a montré qu'elle était incomplète et a démontré la conjecture de BERGE pour cette classe en proposant un algorithme polynomial de coloration des graphes parfaits sans diamant.

c. Les graphes sans taureau (ou sans bull)

Un taureau (ou bull) est le graphe suivant :

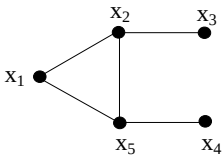


Figure. 2 : Un anneau.

Un graphe est dit sans taureau s'il n'admet pas de sous graphe isomorphe à un taureau.

Théorème [1]

Le théorème fort des graphes parfaits est vrai pour la classe des graphes sans taureau.

d. Les graphes sans dart (ou fléchette)

Un dart (ou fléchette) est le graphe suivant :

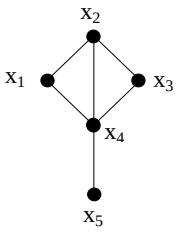


Figure. 3 : Un dart.

Théorème [1]

Tout graphe de BERGE sans dart est parfait.

e. Les graphes sans patte (ou sans paw)

Une patte (ou paw) est le graphe suivant :

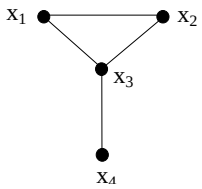


Figure. 4 : Une patte.

Théorème [1]

Tout graphe sans patte est parfait.

Chapitre 2

Problèmes d’affectation sous contraintes et métaheuristiques

Sommaire

Introduction

De nombreux problèmes sont définis en termes de contraintes (de temps, d'espace, ... ou plus généralement de ressources) :

- Les problèmes de planification et ordonnancement : planifier une production, gérer un trafic ferroviaire, ...
- Les problèmes d'affectation de ressources: établir un emploi du temps, allouer de l'espace mémoire, du temps CPU par un système d'exploitation, affecter du personnel à des tâches, des entrepôts à des marchandises, ...
- Les problèmes d'optimisation : optimiser des placements financiers, des découpes de bois, des routages de réseaux de télécommunication, ...

Ces différents problèmes sont des problèmes d'affectation sous contraintes, et ont la particularité commune d'envisager un grand nombre de combinaisons avant de trouver une solution.

Notre travail est axé sur l'étude heuristique des Problèmes d'Affectation Sous Contraintes (A.S.C) que nous avons approché par la coloration des sommets des graphes quelconques.

1. Problèmes d'affectation sous contraintes

La modélisation mathématique d'un problème combinatoire, indispensable à sa résolution rationnelle, est souvent rendue délicate par la grande variété de caractéristiques qui doivent être prise en compte ; le *noyau central* du problème peut généralement s'exprimer sous la forme de contraintes qui doivent absolument être satisfaites. Ainsi, il faut prendre en compte le fait qu'un modèle mathématique réalisé ne modélise pas totalement le problème réel, ce qui est dû à la faible précision des données, à l'absence ou à l'impossibilité de prendre en compte certaines d'entre elles. Les *problèmes d'affectation sous contraintes* offrent un avantage pour la représentation naturelle de larges classes de contraintes discrètes (problème d'emploi du temps, problème d'ordonnancement, …) [23].

1.1. Définition des problèmes d'affectation sous contraintes

Le cadre des *problèmes d'affectation sous contraintes* est rapidement apparu comme un cadre naturel pour représenter les problèmes de décision où les variables représenteront les choix à effectuer tandis que les domaines de valeurs des variables expriment pour chacun des choix les alternatives envisageables; les contraintes entre variables matérialisent les relations nécessaires et/ou souhaitables entre les choix. Cette classe de problèmes inclut notamment les problèmes de satisfaction de contraintes (*CSP* : Constraint Satisfaction Problems) qui est modélisé sous la forme de contraintes, les problèmes de satisfaction partielles (*MCSP :* Maximal Constraint Satisfaction Problems) et les problèmes d'optimisation sous contraintes (*CSOP :* Constraint Satisfaction Optimization Problems).

Les problèmes d'optimisation sous contraintes trouvent de nombreuses applications pratiques. Un tel problème peut se modéliser comme la minimisation (maximisation) de la valeur d'une fonction $f(x_1,...,\ x_n)$ sachant que les variables $x_1,...,\ x_n$ sont soumises à un ensemble de contraintes $C_j(x_1,...,\ x_n)$. Nous pouvons donc distinguer deux problématiques : l'une est la satisfaction qui consiste à trouver une affectation des valeurs aux variables $x_1,...,\ x_n$ vérifiant les contraintes $C_j(x_1,...,\ x_n)$ « solution réalisable » et l'autre est le problème d'optimisation, qui consiste à déterminer parmi ces affectations réalisables, celles qui minimisent (maximisent) la valeur de la fonction objectif « solution optimale ». Pour notre problème nous considèrerons la fonction coût comme étant le nombre de contraintes violées par une configuration solution [23], [6].

Définition 1

Un CSP est un triplet (X, D, C) *tel que*

- $X = \{X_1,\ X_2,...,X_n\}$ *est un ensemble de n variables.*
- *D est la fonction qui associe à chaque variable* X_i *son domaine* $D(X_i)$*, autrement dit, l'ensemble des valeurs que peut prendre* X_i*.*
- $C = \{C_1, C_2,..., C_m\}$ *est un ensemble de contraintes. Chaque contrainte* C_j *est une relation entre les valeurs que peuvent prendre simultanément des variables* [23].

Définition 2

Une affectation est un ensemble de couples variable/valeur

$A = \{X_1 \leftarrow v_1, X_2 \leftarrow v_2, ..., X_r \leftarrow v_r\}$ tel que $0 \leq r \leq n$ et $\forall i \in \{1,...,r\} v_i \in D(X_i)$.

Définition 3

Une affectation est partielle si elle ne concerne qu'une partie des variables de X, et totale si elle concerne toutes les variables de X .

Définition 4

Une affectation A est valide par rapport à un ensemble de contraintes C si pour toute contrainte C_i de C, la relation définie par C_i est vraie pour les valeurs des variables définies dans A.

Définition 5

Une solution d'un CSP est une affectation totale et valide.

Nous présenterons ci-dessous quelques exemples d'application des problèmes d'affectation sous contraintes.

1.2. Exemples de problème d'affectation sous contraintes

1.2.1. Exemple 1

Considérons le problème d'affectation de *n* ouvriers à *n* tâches. L'objectif est de trouver l'affectation univoque ayant un coût moindre :

Données :

- *n* tâches à accomplir ;
- *n* travailleurs disponibles ;
- c_{ij} = coût d'affectation du travailleur *j* à la tâche *i*, $\forall i, \quad j = 1, \ ..., \ n$.

Question :

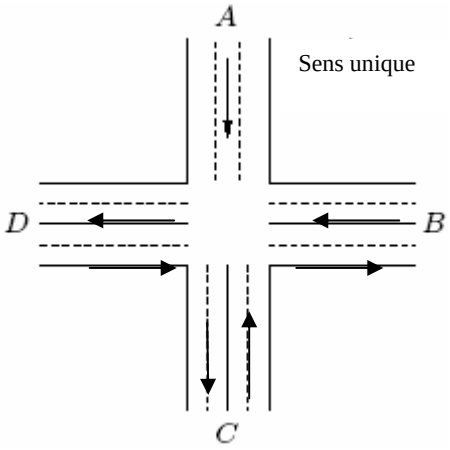
Affecter exactement une tâche *i* à chaque travailleur *j* de façon à minimiser le coût total $\sum_{j=1}^{n} c_{ij}$,

ce problème est « facile » malgré que le nombre de solutions est d'ordre *n*!.

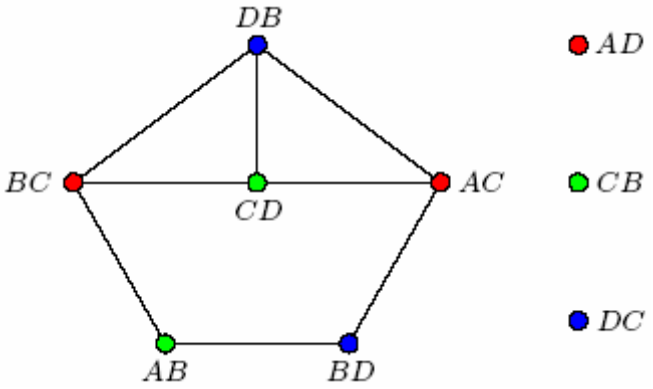
1.2.2. Optimiser le nombre de phases des feux tricolores d'un carrefour

On veut régler les feux d'un carrefour de telle façon qu'on puisse répartir les flux de circulation en groupes. Les flux du même groupe auront le feu vert simultanément (ce qui veut dire que les voitures ne peuvent pas se croiser). Pendant cette période (durée durant laquelle un feu a une couleur), tous les autres flux auront le feu rouge. Ce problème est un problème d'affectation sous contraintes. On peut donc lui appliquer les méthodes connues pour le résoudre. On se propose donc une instance de ce problème et nous tenterons de la résoudre par la coloration.

Une instance de ce problème

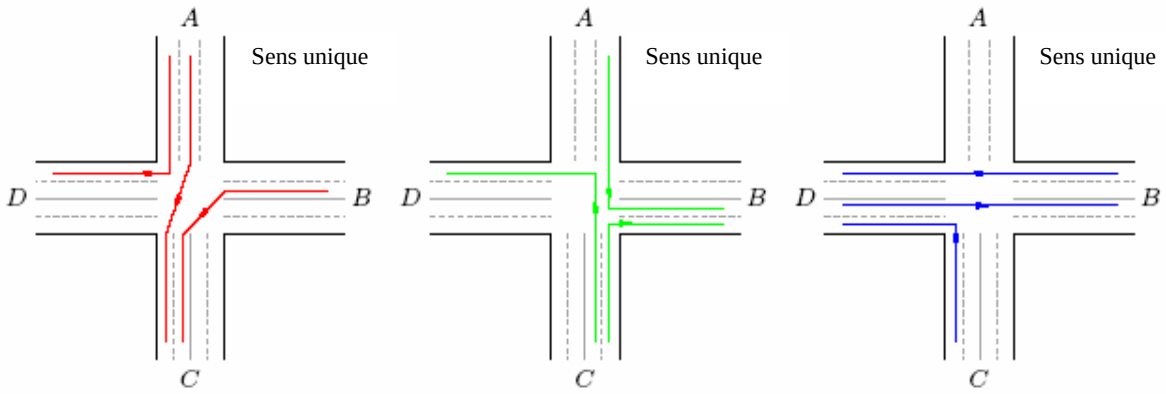


Le graphe G modélisant ce problème est obtenu en considérant les sommets comme étant le passage d'un sens à un autre (on peut aller de A à D donc AD est un sommet de G) et deux sommets sont adjacents dans G si les deux passages leurs correspondants se croisent (le passage de A à C se croise avec le passage de C à D, donc le sommet AC est adjacent à CD), on obtient ainsi le graphe de la Figure ci-dessous.

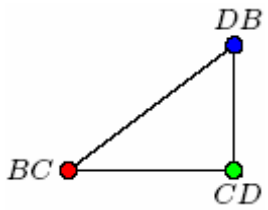


Dans cet exemple, la résolution revient à colorier les sommets de façon à attribuer des couleurs différentes aux sommets adjacents en utilisant un nombre minimum de couleurs. Ainsi le problème devient celui de la coloration des sommets d'un graphe donc d'affectation sous contraintes, où la contrainte est représentée par l'adjacence dans le graphe G.
On en déduit de ce qui précède une solution, qui utilise 3 couleurs :

De cette solution, on déduit les flux de voitures suivants :



Cette solution est optimale. En fait, il suffit de regarder le sous-graphe suivant du graphe G :



On voit qu'il possède une arête entre chaque paire de sommets. Donc les trois sommets sont adjacents deux à deux et il faut au minimum trois couleurs pour résoudre le problème posé. La solution en utilise exactement trois, il s'agit donc d'une solution optimale.

1.2.3. Problème de mise en boîtes (Bin packing)

Données :

- Une liste de n objets de poids w_i, $i = \overline{1,\ n}$;
- Un ensemble de boîtes, chacune de capacité donnée W.

Question :

Déterminer une répartition, affectation, réalisable des objets dans les boîtes de façon à minimiser le nombre de boîtes utilisées.

Une répartition réalisable signifie que la somme des poids des objets dans chacune des boîtes est inférieure ou égale à W. Il s'agit d'un problème « difficile ».

2. Les méthodes approchées

Résoudre donc exactement des problèmes d'optimisation combinatoire NP- difficiles est chose très ardu. En effet, pour de tels problèmes, les méthodes exactes requièrent un effort calculatoire qui croît exponentiellement avec la taille des instances du problème (explosion combinatoire) et, rapidement, les méthodes approchées[1] , heuristique et/ou métaheuristique, deviennent le quasi-unique moyen d'obtenir une bonne solution en un temps raisonnable [28]. Ces méthodes sont principalement basées sur des méthodes de recherche locale (méthodes de Descente, Tabou [21], Recuit Simulé [25]) ou des méthodes évolutives (Algorithmes Génétiques [24], Colonies De Fourmis [8],…).

Cependant, même pour les problèmes d'optimisation combinatoire les plus difficiles, les méthodes exactes demeurent très efficaces pour de petites tailles d'instances, la solution optimale étant trouvée et prouvée en un temps très court.

Dans la suite de notre travail, nous nous intéresserons uniquement aux méthodes, de la Recherche Tabou (RT) et des algorithmes basés sur le principe de Colonie de Fourmis(ACO).

2.1. Les voisinages et les transformations élémentaires

Le principe d'une méthode itérative consiste à partir d'une solution de départ x_0, on engendre une suite (finie) de solutions de proche en proche :

$$x_i \to x_{i+1} / f(x_{i+1}) < f(x_i), \forall i.$$

[1] Voir le chapitre1.

Pour mettre en œuvre cette méthode, il est impératif de définir la *transformation élémentaire* (ou *locale*) permettant d'engendrer la nouvelle solution x_{i+1} à partir de la solution courante x_i. Une transformation ne sera considérée comme élémentaire que si elle ne modifie que « faiblement » la solution courante.

Une transformation élémentaire ou le passage d'une solution x_i à une autre solution x_{i+1}, s'effectue par un « mouvement » $m(x_i)$. L'ensemble des solutions pouvant être obtenues à partir de x_i en appliquant une transformation élémentaire est nommé le « voisinage de x_i », soit $S(x_i)$.

2.2. Les métaheuristiques

Les métaheuristiques ont changé radicalement l'élaboration des heuristiques, par le fait, qu'on s'interroge d'abord sur les caractéristiques et les particularités du problème à résoudre avant de commencer à programmer une méthode spécifique. Les métaheuristiques ont en quelque sorte inversé le processus, la trame de la méthode de résolution étant fournie par la métaphore qui a inspiré la métaheuristique. Ayant une première heuristique, on cherche ensuite à l'améliorer en observant les faiblesses qu'elle présente une fois appliquée au problème à résoudre.

Une question à laquelle bien des personnes ont cherché à répondre est de savoir si telle ou telle métaheuristique est meilleure qu'une autre ou non. Jusqu'à présent la réponse à cette question n'est toujours pas obtenue, pour la simple raison que l'efficacité, des méthodes mises au point, dépend avant tout des adaptations de la trame de base aux spécificités du problème. Ainsi, plus la trame est riche et complexe, comme par exemple la recherche tabou, plus on a de possibilités d'amélioration d'une méthode de base. Par contre, il est difficile de sélectionner et d'assembler les bons principes de la métaheuristique. Elles sont classées en deux groupes : les méthodes à solution unique et les méthodes à population de solutions. Les méthodes itératives à solution unique sont toutes basées sur un algorithme de recherche de voisinage qui commence avec une solution initiale, puis l'améliore pas à pas en choisissant une nouvelle solution dans son voisinage [3], telles que les méthodes de descente, récuit simulé, la méthode tabou…etc. Cependant, 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 telles que les colonies de fourmis, les algorithmes génétiques etc…

2.2.1. Amélioration itérative (méthode de descente)

Cette méthode vise à déterminer une solution $s(x)$ dans le voisinage de la solution courante ***x***, telle que $f(s(x)) < f_{\min}$ ($f_{\min}$ désigne la valeur minimale courante de *f*). La méthode consiste à engendrer, à chaque itération, un *N*-échantillon, suivant un procédé aléatoire ou cyclique, ou suivant une loi de distribution uniforme, dans le voisinage de la solution courante ***x***. La fonction objectif *f* est évaluée en chaque point de l'échantillon, et la solution x' correspond à la plus petite valeur de f obtenue, $f(x') = f(s(x)) = \min_{1 \le i \le N} \left[f(s_i(x)) \right]$.

Cette nouvelle valeur $f(x')$ est comparée à la valeur minimale courante $f_{\min}$. Si elle est meilleure, cette valeur est enregistrée, ainsi que la solution correspondante, et réitéré.

Après avoir atteint un minimum local, cette procédure peut repartir d'un autre point pris au hasard.

2.2.2. La méthode du recuit simulé

Le recuit simulé trouve ses origines dans la thermodynamique. Cette méthode est issue d'une analogie entre le phénomène physique de refroidissement lent d'un corps en fusion, qui le conduit à un état solide, de basse énergie. Il faut abaisser lentement la température, en marquant des paliers suffisamment longs pour que le corps atteigne l'« équilibre thermodynamique » à chaque palier de température. Pour les matériaux, cette basse énergie se manifeste par l'obtention d'une structure régulière, comme dans les cristaux et l'acier [15].

L'analogie exploitée par le recuit simulé consiste à considérer une fonction *f* à minimiser comme fonction d'énergie, et une solution ***x*** peut être considérée comme un état donné de la matière dont ***f(x)*** est l'énergie. Le recuit simulé exploite généralement le critère défini par l'algorithme de Metropolis pour l'acceptation d'une solution obtenue par perturbation de la solution courante.

Pour une « température » T donnée, à partir d'une solution courante $\boldsymbol{x}$, on considère une transformation élémentaire qui changerait $\boldsymbol{x}$ en $\boldsymbol{s(x)}$. Si cette perturbation induit une diminution de la valeur de la fonction objectif, $\Delta f = f\left(s\left(x\right)\right) - f\left(x\right) < 0$, elle est acceptée. Dans le cas contraire, si $\Delta f = f\left(s\left(x\right)\right) - f\left(x\right) \geq 0$, la perturbation est acceptée avec une probabilité $p = \exp\frac{-\Delta f}{T}$.

Le paramètre de contrôle T est la « température » du système, qui influe sur la probabilité d'accepter une solution plus mauvaise. A une température élevée, la probabilité d'acceptation d'un mouvement quelconque tend vers 1 : presque tous les changements seront acceptés.

L'algorithme équivaut alors à une marche aléatoire dans l'espace des configurations. Cette température est diminuée lentement au fur à mesure du déroulement de l'algorithme pour simuler le processus de refroidissement des matériaux, et sa diminution est suffisamment lente pour que l'équilibre thermodynamique soit maintenu.

L'efficacité du recuit simulé dépend fortement du choix de ses paramètres de contrôle, dont le réglage reste très empirique.

2.2.3. La méthode de la recherche tabou

La Recherche Tabou (*RT*) est une métaheuristique originalement développée par GLOVER et indépendamment par HANSEN [22], sous l'appellation de "*steepest ascent mildest descent*". Cette méthode combine une procédure de recherche locale avec un certain nombre de règles et de mécanismes permettant à celle-ci de surmonter l'obstacle des optima locaux, tout en évitant de cycler. Elle a été appliquée avec succès pour résoudre de nombreux problèmes difficiles d'optimisation combinatoire : problèmes de routage de véhicule, problèmes d'affectation quadratique, problèmes d'ordonnancement, problèmes de coloration de graphes, [15], [6], …etc.

2.2.3.1. Principe de base

Dans une première phase, la méthode de recherche tabou peut être vue comme une généralisation des méthodes d'amélioration locale. En effet, en partant d'une solution quelconque $\boldsymbol{x}$ appartenant à l'ensemble de solutions $\boldsymbol{X}$, on se déplace vers une solution $\boldsymbol{s}(\boldsymbol{x})$

située dans le voisinage $S(\boldsymbol{x})$ de $\boldsymbol{x}$. Donc l'algorithme explore itérativement l'espace de solutions $\boldsymbol{X}$.

Afin de choisir le meilleur voisin $\boldsymbol{s}(\boldsymbol{x})$ dans $S(\boldsymbol{x})$, l'algorithme évalue la fonction objectif f en chaque point $\boldsymbol{s}(\boldsymbol{x})$, et retient le voisin qui améliore la valeur de la fonction objectif f, ou au pire celui qui la dégrade le moins.

L'originalité de la méthode de recherche tabou, par rapport aux méthodes locales, qui s'arrêtent dès qu'il n'y a plus de voisin $s(\boldsymbol{x})$ permettant d'améliorer la valeur de la fonction objectif f, réside dans le fait que l'on retient le meilleur voisin, même si celui-ci est plus mauvais que la solution d'où l'on vient. Ce critère autorisant les dégradations de la fonction objectif, évite à l'algorithme d'être piégé dans un minimum local. Mais il induit un risque de cyclage. En effet, lorsque l'algorithme a quitté un minimum quelconque par acceptation de la dégradation de la fonction objectif, il peut revenir sur ses pas, à l'itération suivante.

Pour régler ce problème, l'algorithme a besoin d'une mémoire pour conserver pendant un moment la trace des dernières meilleures solutions déjà visitées. Ces solutions sont déclarées *taboues*, d'où le nom de la méthode. Elles sont stockées dans une liste de longueur L donnée, appelée *liste tabou*. Une nouvelle solution n'est acceptée que si elle n'appartient pas à cette liste tabou. Ce critère d'acceptation d'une nouvelle solution évite le cyclage de l'algorithme, durant la visite d'un nombre de solutions au moins égal à la longueur de la liste tabou, et il dirige l'exploration de la méthode vers des régions du domaine de solutions non encore visitées.

La liste tabou est généralement gérée comme une liste « *circulaire* » : on élimine à chaque itération la solution tabou la plus ancienne, en la remplaçant par la nouvelle solution retenue. Mais le codage d'une telle liste est encombrant, car il faudrait garder en mémoire tous les éléments qui définissent une solution. Pour pallier cette contrainte, on remplace la liste tabou de solutions interdites par une liste de « *transformations interdites* », en interdisant la transformation inverse d'une transformation faite récemment.

2.2.3.2. Critère d'aspiration

Le remplacement de la liste tabou des solutions visitées par la liste des transformations élémentaires $\{\boldsymbol{x}, s(\boldsymbol{x})\}$ conduit non seulement à l'interdiction de revenir vers des solutions

précédentes, nous éviterons ainsi le cyclage court, mais aussi vers un ensemble de solutions dont plusieurs peuvent ne pas avoir été visitées jusqu'ici. Il est donc primordial de corriger cette limite et de trouver un moyen de lever l'interdiction de l'acceptation d'une transformation élémentaire $\{\boldsymbol{x}, s(\boldsymbol{x})\}$ déjà effectuée (donc appartenant à la liste tabou), sous un certain critère, appelé *critère d'aspiration*. Cette correction permet aussi de revenir à une solution déjà visitée et de redémarrer la recherche dans une autre direction.

Le critère d'aspiration le plus simple et le plus couramment utilisé consiste à tester si la solution produite de statut tabou présente un coût inférieur à celui de la meilleure solution trouvée jusqu'à présent. Si cette situation se produit, le statut tabou de la solution est levé. Ce critère est évidemment très sévère, il ne devrait pas être vérifié très souvent, donc il apporte peu de changements à la méthode. D'autres critères d'aspiration plus complexes peuvent être envisagés. L'inconvénient de recourir trop souvent à l'aspiration est qu'elle peut détruire, dans une certaine mesure, la protection offerte par la liste tabou vis-à-vis du cyclage.

2.2.3.3. Intensification

L'intensification consiste à approfondir la recherche dans certaines régions du domaine, identifiées comme susceptibles de contenir un optimum global. Cette intensification est appliquée périodiquement, et pour une durée limitée. Pour mieux intensifier la recherche dans une zone bien localisée, plusieurs stratégies sont proposées dans la littérature.

La plus simple consiste à retourner à l'une des meilleures solutions trouvée jusqu'à présent, puis de reprendre la recherche à partir de cette solution, en réduisant la longueur de la liste tabou pour un nombre limité d'itérations. Dans ce cas, on adapte la procédure de recherche tabou, en élargissant le voisinage de la solution courante (augmentation de la taille de l'échantillon $S(\boldsymbol{x})$), tout en diminuant le pas des transformations.

2.2.3.4. Diversification

La diversification permet à l'algorithme de bien explorer l'espace des solutions, et d'éviter que le processus de recherche ne soit trop localisé et laisse de grandes régions du domaine totalement inexplorées. La plus simple des stratégies de diversification consiste à interrompre périodiquement l'acheminement normal de la procédure tabou, et à la faire redémarrer à partir d'une autre solution, choisie aléatoirement, ou « intelligemment ». Une autre méthode consiste

à biaiser la fonction d'évaluation *f*, en introduisant un terme qui pénalise les transformations effectuées fréquemment, afin de favoriser des transformations nouvelles. Ce type de stratégie de diversification peut être utilisé de façon continue, sans interrompre la procédure de recherche tabou.

En résumé, nous dirons que la diversification et l'intensification sont des concepts complémentaires, qui enrichissent la méthode de recherche tabou et lui permettent d'atteindre une plus grande robustesse et efficacité.

2.2.4. Les colonies de fourmis (Ants Colony)

L'histoire de l'intelligence en essaim remonte à l'étude du comportement de fourmis à la recherche de nourriture au départ de leur nid, par GOSS, DENEUBOURG et leur équipe [9] et [10]. Les algorithmes inspirés des fourmis, appelés "Ant Algorithms", ont fait l'objet de nombreux travaux, qui en font un principe heuristique général pouvant être appliqué à de nombreux problèmes combinatoires. Marco Dorigo [11] est l'un des pionniers dans le développement des fourmis artificielles et c'est à lui que revient l'application, pour la première fois (dès 1992) de celles-ci à un problème combinatoire très célèbre, le problème du voyageur de commerce, dont l'objectif est de trouver le plus court chemin hamiltonien entre les différentes villes constituant les données du problème.

Cependant, l'optimisation par colonies de fourmis est une métaheuristique qui a déjà été utilisée avec succès pour résoudre des problèmes combinatoires difficiles (voyageur de commerce, coloration de graphes, affectation quadratique...) [24].

En se déplaçant du nid à la source de nourriture et vice-versa (ce qui, dans un premier temps, se fait essentiellement d'une façon aléatoire), les fourmis déposent au passage sur le sol une substance odorante appelée phéromone, ce qui a pour effet de créer une piste chimique. Les fourmis peuvent sentir ces phéromones qui ont un rôle de marqueur de chemin : quand les fourmis choisissent leur chemin, elles ont tendance à choisir la piste qui porte la plus forte concentration de phéromones (voir figure 1). Cela leur permet de retrouver le chemin vers leur nid lors du retour. D'autre part, les odeurs peuvent être utilisées par d'autres fourmis pour retrouver les sources de nourriture détectées par le reste de fourmis [27].

Une des propriétés de la phéromone est sa volatilité progressive. Un chemin emprunté par des fourmis, la phéromone est maintenue contrairement aux autres chemins plus long où la substance en question s'évapore de plus en plus et les fourmis deviennent aveugles (sans repère de guide).

Le système de fourmis, est une méthode d'optimisation basée sur ces observations proposées par DORIGO [12], [13] et [14].

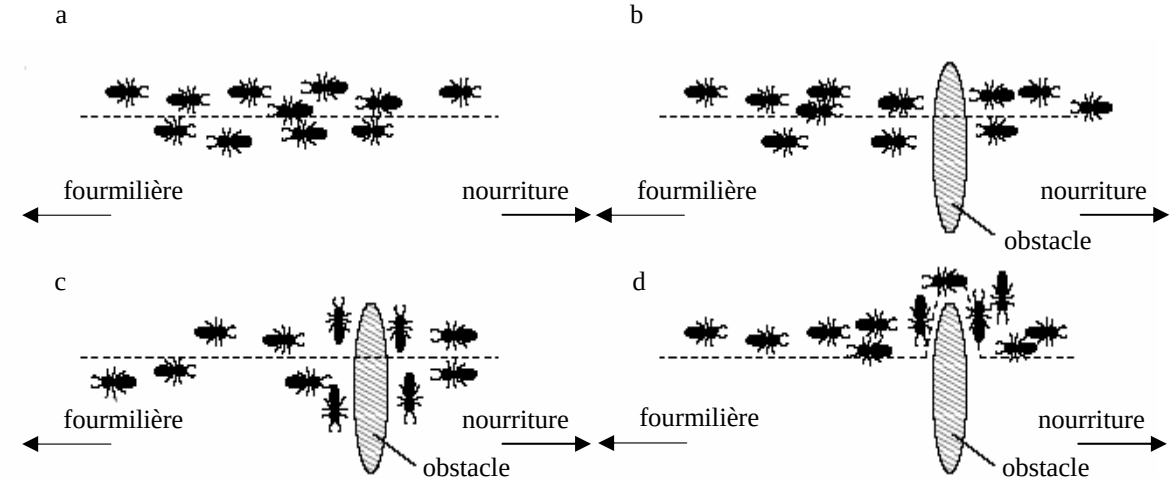


Figure. 1

- (a) Les fourmis suivent un chemin entre la fourmilière et la nourriture ;

- (b) Un obstacle apparaît sur le chemin ; les fourmis choisissent entre prendre à droite ou à gauche avec équiprobabilité ;

- (c) La phéromone s'évapore sur le chemin le plus long ;

- (d) Toutes les fourmis ont tendance à choisir le chemin le plus court.

2.2.4.1. Mode d'application de la méthode ACO pour notre problème

L'application de cette métaheuristique aux problèmes d'affectation sous contraintes est faite de la manière suivante :

Le problème d'A. S. C peut être modélisé sous la forme d'un graphe dont les sommets sont les valeurs et les fourmis artificielles cherchent des bons chemins. Les chemins trouvés sont ensuite évalués en fonction du nombre de contraintes qu'ils violent, une solution n'étant

valide que si elle respecte toutes les contraintes. Les chemins sont construits par les fourmis et correspondent à une affectation complète.

Il y a évidemment des différences entre une fourmi réelle et sa représentation virtuelle. Celle-ci a une mémoire qui lui permet de garder une trace de la partie de solution déjà trouvée et elle ne choisit pas sa trajectoire qu'en fonction de la phéromone mais aussi grâce à une visibilité locale. De plus la phéromone n'est déposée qu'après avoir atteint une solution.

Chapitre 3

Approches de résolution

Sommaire

Introduction

Dans ce travail, nous nous proposons d'apporter une contribution à la résolution du Problème de la Coloration des Sommets d'un Graphe quelconque (P.C.S.G) et par là les problèmes d'affectation sous contraintes, pour qui, une des modélisations est le PCSG. En effet, il suffit d'une part, de construire deux bijections, l'une entre les objets à affecter et l'ensemble des sommets des graphes et l'autre entre les affectations possibles et l'ensemble des différentes couleurs et d'autre part de considérer les contraintes comme étant la relation d'adjacence dans les graphes modélisant les problèmes d'affectation sous contraintes.

A titre d'exemple, on peut considérer un exemple où il est question d'une affectation de n candidats à n postes de façon que un candidat peut ne pas postuler pour tous les postes et qu'une fois qu'un poste est affecter pour un candidat, alors il ne peut être affecter à aucun autre candidat. Selon ce qui a été déjà énoncé ; les candidats sont les sommets du graphe modélisant ce problème, tels que de sommets sont adjacent si les candidats leur correspondant ont postuler pour le même poste, les postes sont les différentes couleurs et les contraintes sont les relations d'adjacence entre les sommets.

Les problèmes d'affectation sous contraintes s'adaptent à une grande partie des problèmes d'optimisation combinatoire, rencontrés en industrie, dans l'ordonnancement et autres applications pratiques et/ou opérationnelles, pour ne citer que les problèmes ; d'emploi du temps, Allocation des ressources, Placement et Localisation…etc.

La résolution des problèmes d'affectation sous contraintes consiste en la coloration des graphes les modélisant tout en respectant les contraintes de la non adjacence de deux sommets de même couleur.

1. Démarche proposée pour la résolution des problèmes d'A. S. C

Selon la modélisation énoncée ci-dessus, une solution pour un problème d'affectation sous contraintes est une configuration dont le nombre de contraintes violées est nul (chaque deux sommets adjacents ont deux couleurs différentes), si celui-ci diffère de zéro, notre travail consiste à le rendre nul tout en minimisant le nombre de couleurs utilisés pour colorier les sommets des graphes modélisant le problème d'affectation sous contraintes.

Pour un tel objectif notre démarche est constituée des étapes suivantes :

Etape 1 : *Construction d'une configuration solution* initiale (partielle ou totale)[1] *;*

Si la solution est *partielle* aller à l'*étape 2*

Etape 2 : *Application d'un algorithme basé sur le principe :*

De colonie de fourmis (Ant Colony Optimisation, ACO) ;

La recherche tabou.

La première étape consiste à l'élaboration d'une procédure de recherche d'une configuration initiale tandis que, la deuxième étape consiste à combiner cette procédure dans une méthode hybride intégrant le principe des colonies de fourmis avec la méthode de recherche tabou que nous décrirons après. Cependant, une large partie de notre travail est basée sur la première étape dont nous avons développé et élaboré une procédure de coloration que nous avons notée $P^{\oplus}$. Notre procédure de coloration $P^{\oplus}$ peut être décrite par ce qui suit :

1.1. Etape d'initialisation

Notre procédure nous permet d'obtenir une solution optimale selon les spécificités du graphe support sur lequel elle est appliquée.

1.1.1. Description de la procédure $P^{\oplus}$

Elle opère selon les étapes suivantes :

1ere étape : Détermination d'un sommet star, noté $x^* / d_G(x^*) = \max_{v \in V} d_G(v)$;

2eme étape : Après avoir choisit la couleur de plus petit indice, attribuée au sommet star, affecter celle-ci à tous les sommets constituant les extrémités terminales des chaînes minimales issues de x^* et de longueur paire ;

3eme étape : Repérer les voisins du sommet star x^* ;

4eme étape : Attribuer les couleurs de plus petits indices possibles autre que celle de x^*, sans que deux sommets adjacents dans le voisinage de x^*, aient la même couleur.

[1] Voir chapitre 1.

5eme étape : S'il existe un sommet de même couleur que celle de x^* constituant une extrémité d'une chaîne minimale de longueur paire entre celui-ci et un sommet non colorié ou voisin de x^* alors, choisir ce sommet comme le nouveau sommet star et aller en 2, si non aller en 6.

6eme étape : Choisir un sommet dans le voisinage de x^* d'indice de couleur le plus petit possible relié par une chaîne minimale de longueur paire avec un sommet non colorié ou voisin de celui-ci comme étant le nouveau sommet star et aller en 2.

Critère d'arrêt : tous les sommets sont coloriés.

1.1.2. Justification de la procédure

La procédure ainsi construite est convergente et finie.

La convergence : Sa convergence est assurée par construction, la procédure ainsi développer procure une solution partielle ou totale.

La finitude : A chaque itération nous cherchons à réduire le nombre de contraintes violées (ce nombre est fini et ne dépasse pas le nombre d'arêtes). Ce qui justifie sa finitude.

1.1.3. La complexité de la procédure $P^{\oplus}$

Nous passerons en revue les différentes étapes de notre procédure afin d'évaluer son temps d'exécution. Pour ce faire, nous attribuerons un coût en temps à chaque instruction, et nous compterons le nombre d'exécutions de chacune des instructions.

- La première étape de la procédure est faite de n opérations donc requiert un temps d'exécution d'ordre $O(n)$.

- La deuxième étape est faite d'au plus $n-1$ comparaisons, ce qui donne un ordre d'exécution $O(n)$. En effet, nous comparons le sommet star avec les $n-1$ sommets non coloriés à fin de détecter les sommets reliés par des chaînes minimales de longueur paire avec celui-ci.

- La troisième étape est linéaire, d'ordre $O(n)$.

- La quatrième étape est faite elle aussi de $n-m-1$ tel que *m* est le nombre de sommets coloriés à la deuxième étape.

- La cinquième étape est faite de $n-m-k-1$ tel que *k* est le nombre de sommets coloriés à la quatrième étape.

- En fin, la sixième étape nécessite $n-m-k-h-1$ tel que *h* est le nombre de sommets coloriés à la cinquième étape.

Il apparaît suite à ce qui précède que la procédure $P^{\oplus}$ requiert un temps d'exécution d'ordre $O(n)$, donc est polynomiale. Plus de éclaircissements sur $P^{\oplus}$ voir Annexe.

Nous proposons dans la suite la formulation algorithmique de cette procédure.

Pour la bonne compréhension de celle-ci, nous adoptons la notation suivante :

P_{xy} : La chaîne minimale de longueur paire reliant le sommet x au sommet y.

1.1.4. La formulation algorithmique de la procédure $P^{\oplus}$

{**Entrées** $G=(V,\ E)$ où V : L'ensemble des sommets du graphe ; $|V|=n$; E : L'ensemble des arêtes du graphe}.

{**Sortie** *une coloration de G*}.

Variables utilisées

T, T_1 : Les listes tabous contenant les sommets déjà coloriés ;

colsommet : Variable booléenne, tel que :

$$\text{colsommet} = \begin{cases} \text{Vrai} & \text{si tous les sommets sont coloriés;} \\ \text{Faux} & \text{sinon.} \end{cases}$$

Couleur (x) : Fonction retournant la couleur du sommet x.

Début

Étape d'initialisation (0) :

colsommet :=faux ;

$x^* \in G$ tel que $d_G(x^*) = \max_{v \in V} d_G(v)$;

Couleur (x^*) :=1 ;

$T := \{x^*\}$;

Étape d'itération (1) :

Tant que ($\left(x_j \notin T\right) \wedge \left(\left|T\right| \leq n\right)$ *faire*

Si $P_{x_j\, x^*}$ *alors*

Couleur (x_j) := 1

Sinon si $\left(x_j,\ x^*\right) \in E$ *alors*

Couleur (x_j) reçoit la plus petite couleur qui n'apparaît pas dans l'un de ses voisins ;

$T := T \cup x_j$;

$T_1 := \left\{x^*\right\}$;

Répéter

$x_i := \min_{v \in T / T_1} couleur(v); T_1 := T_1 \cup \{x_i\};$

Tant que $\left(x_j \notin T\right) \wedge \left(i \neq j\right)$ *faire*

Si $P_{x_i x_j}$ *alors*

Couleur (x_j) :=couleur (x_i)

Sinon si $\left(x_i,\ x_j\right) \in E$ *alors*

Couleur (x_j) := la plus petite couleur qui n'apparaît pas dans l'un de ses voisins ;

$T := T \cup x_j$;

Si $\left|T\right| = n$ *alors*

colsommet :=vraie ;

Jusqu'à colsommet = vraie ;

une coloration de $G := \max_{v \in V} couleur(v)$;

Fin.

Afin d'avoir une configuration initiale par la procédure $P^{\oplus}$, nous proposons la fonction d'évaluation suivante permettant de trouver le nombre de contraintes violées.

1.1.5. L'évaluation de l'exécution de $P^{\oplus}$

Une contrainte est violée s'il peut y avoir une corde reliant deux sommets, de deux chaînes de longueur minimale, à distance paire du sommet x^*.

1.1.5.1. La fonction d'évaluation

Début

Ordonner les sommets selon l'ordre croissant de leurs couleurs;

$A := \{\phi\}$; $B := \{\phi\}$; i:=1; cpt:=0;

tant que $|X| \neq \phi$ *ou* $|E| \neq \phi$ *faire*

pour j:=i+1 *à* n *faire*

si x_j est un sommet voisin de x_i *alors*

$A := A \cup \{x_j\}$; $E := E - \{[x_i, x_j]\}$;

fsi;

fait

pour j:=1 *à* $|A|$ *faire*

si les sommets x_i et x_j ont la même couleur *alors*

cpt:=cpt+1;

$B := B \cup \{x_j\}$;

fsi;

fait;

$X := X - \{x_i\}$; i:=i+1; $A := \{\phi\}$;

fait;

si $|B| = 0$ *alors*

la solution rencontrée est une solution réalisable

fsi;

Fin.

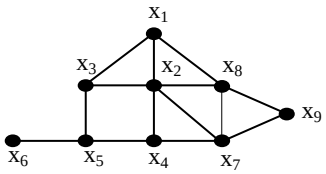
1.1.5.2. Justification de la fonction d'évaluation

Comme le nombre d'arêtes est fini, le nombre de sommets est aussi fini donc la fonction construite est finie; sa convergence est assurée par construction.

1.1.6. Essai de la méthode

Notre méthode a été programmée avec Delphi sous Windows et exécutée sur un Pentium VI. Nous présentons dans ce qui suit quelques instances, graphes, sur lesquelles nous avons appliqué notre méthode.

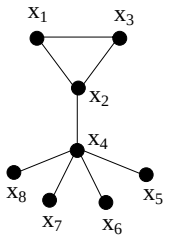
1. Soit le graphe G représenté dans la figure qui suit :



Le sommet x^* est le sommet x_2 (sommet d'initialisation de la procédure), donc x_2 reçoit la couleur 1. La chaîne minimale reliant x_2 au sommet x_9, respectivement x_5, est de longueur paire. Ainsi, les sommets x_9 et x_5 reçoivent la couleur du sommet x_2, la couleur 1. Cependant, les sommets x_1, x_3, x_4, x_7 et x_8 sont des sommets voisins au sommet x_2 donc on leur attribut la couleur de plus petit indice, non encore attribuée, tout en respectant les relations d'adjacence entre ces sommets, ceci nous amène à affecter la couleur 2 aux sommets x_3, x_4, et x_8 puisque ils sont deux à deux non adjacents entre eux et tous adjacents au sommet star x_2 et d'attribuer la couleur 3 au sommets x_1 et x_7.

Au bout de cette première itération on constate que le sommet x_6 n'est pas encore colorié donc d'après notre méthode, on choisit un sommet colorier par la couleur du premier sommet x^* considéré, le sommet x_2 et de réitérer la procédure tout en considérant le nouveau sommet x^* qui est dans ce cas le sommet x_5. Le sommet x_6 est voisin au sommet x_5 donc la couleur qu'elle lui sera affecter est la couleur 2.

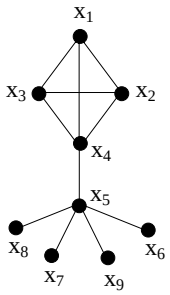
2. Considérons maintenant un autre graphe G qui est représenté par la figure ci-dessous :



Par notre procédure $P^{\oplus}$, un tel graphe est 2-coloriable du fait que les deux sommets x_1 et x_3 sont reliés au sommet x_4, qui est le sommet x^*, par deux chaînes minimales de parité paire donc ils reçoivent la même couleur que celle du sommet x_4 qui est la couleur 1, les sommets x_1 et x_3 sont reliés, $(x_1, x_3) \in E$, ce qui viole une contrainte, solution partielle. Les autres sommets de ce graphe sont des sommets voisins au sommet x_4 sans qu'ils soient

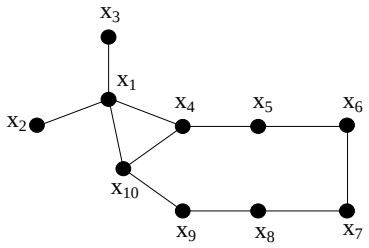
reliés entre eux, donc ils seront coloriés par la couleur 2, mais le graphe G est 3-coloriable.

3. Soit un graphe G donné par la figure suivante :



Par notre procédure $P^{\oplus}$, G est 2-coloriable du fait que, les sommets x_6, x_7, x_8 et x_9 sont des sommets pendants et voisins au sommet x^* (sommet x_5) donc ils seront coloriés avec la couleur 2, le sommet x_4 sera colorié avec la couleur 2 car il appartient au voisinage de x_5 tandis que les autres sommets x_1, x_2, x_3 seront coloriés avec la couleur 1 puisqu'ils représentent les extrémités des chaînes minimales, de parité paire, qui sont issues du sommets x_5. Mais, le graphe G est 4-coloriable et ceci car il contient une clique, maximale, de cardinalité 4. Ce qui fait, le nombre de contraintes violées est de 3, tout le travail de notre méthode consiste à rendre ce nombre nul.

4. Soit le graphe G représenté par la figure ci-dessous :



Le graphe G contient un trou comme sous graphe induit, donc G n'est pas parfait. Cependant, un tel graphe est coloriable par notre procédure $P^{\oplus}$ et ceci car les sommets x_5, x_7 et x_9 sont des extrémités des chaînes minimales, de parité paire, issues du sommet x_1 qui est le sommet x^* donc ils seront coloriés avec la couleur 1. En effet, les sommets x_2, x_3, x_4 et x_{10} sont des sommets voisins au sommet x_1 et les sommets x_4 et x_{10} sont reliés par une arête donc ils reçoivent deux couleurs différentes 2 et 3 respectivement et les sommets x_3 et x_2 seront coloriés avec la couleur 2. Les sommets x_6 et x_8 seront coloriés avec la couleur 2 du fait qu'ils sont voisins des sommets coloriés avec la couleur 1.

1.1.7. Performance de $P^{\oplus}$

Etant donné un graphe G, dont les sommets représentent des objets, il s'agit d'attribuer un ou plusieurs nombres entiers (qui représentent les couleurs) à chaque objet en satisfaisant un certain nombre de contraintes tout en essayant à utiliser un nombre minimum d'entiers. En effet, le nombre de couleurs utilisé est très petit du fait que la violation des contraintes est permise. Cependant, une solution totale par notre procédure $P^{\oplus}$ est une solution ne violant aucune contrainte donc si le nombre de contraintes violées, par notre procédure, est nul alors la solution obtenue converge vers la solution optimale.

De plus notre procédure, $P^{\oplus}$, est performante et procure des solutions de bonnes qualités vue que notre classe des graphes parfaits, G^*, est parfaitement coloriable par celle-ci (qui fait l'objet du chap. 4) et que G^* contient des classes connues des graphes parfaits, polynomialement coloriable, telles que : les graphes biparti, les graphes de Berge multiparti, les graphes de Berge sans griffe et les graphes de Berge sans patte.

1.2. Etape d'amélioration

Afin d'améliorer la solution obtenue, par la procédure $P^{\oplus}$, nous cherchons donc à réduire à chaque itération le nombre de contraintes violées (voir chapitre 2), et ceci pour les graphes qui ne sont pas coloriable de manière polynomiale par la procédure $P^{\oplus}$ (relatif à la deuxième étape). Cependant, nous utilisons notre procédure $P^{\oplus}$ pour initialiser la solution de départ que nous améliorons dans cette étape par une méthode basée sur le principe des colonies de fourmis (ACO) et de la recherche tabou (RT), dans le but de satisfaire toutes les contraintes.

1.2.1. Amélioration par la méthode basée sur le principe de colonie de fourmis

Le préalable nécessaire à l'application du principe ACO au Problème de Coloration des Sommets d'un Graphe (PCSG), est l'encodage de celui-ci en un problème de recherche de plus court chemin d'un graphe de telle sorte qu'un chemin traduit une partition en stable. Donc le problème d'affectation sous contraintes est modélisé sous la forme d'un graphe dont les sommets sont les valeurs ou les objets à affecter et les fourmis artificielles cherchent les chemins optimaux, partitions en stables, ou les affectations possibles. Les chemins trouvés sont ensuite évalués en fonction du nombre de contraintes violées, une solution n'étant valide

que si elle respecte toutes les contraintes. Les chemins sont construits par les fourmis et correspondent à une affectation complète ou totale.

1.2.1.1. Adaptation du principe d'ACO

La mise en œuvre de la méthode des fourmis pour la résolution du problème de la coloration des sommets d'un graphe fait intervenir les paramètres suivants [18], [26]:

f : le nombre de fourmis;

w_{ij} : distance, en nombre d'arêtes, entre le sommet i et le sommet j (la longueur de la chaîne reliant ces deux sommets);

M_d^i : la longueur de la plus grande chaîne à partir du sommet i;

S_{ij} : mesure la visibilité du sommet j depuis le sommet i (en terme de la longueur de la chaîne qui les reliée) : $S_{ij} = \left(M_d^i - w_{ij}\right) / M_d^i$;

ℓ : coefficient d'évaporation $0 \le \ell \le 1$;

F : la cardinalité de chaque « tournée » (partition) ;

P_{av} : probabilité de déplacement aveugle ;

K : le nombre de sommets qui sont reliés par une chaîne de longueur ≥ 2 ;

P_{ls} : la probabilité de lancer une recherche locale sur une solution ;

Les variables

σ_{ij} : la quantité de phéromone sur la chaîne reliant sommet i au sommet j ;

$\Delta\sigma_{ij}$: l'ajout de phéromone ;

L^{μ} : coût total de la "tournée", l'évaluation, en terme de la fonction objectif ;

T^{μ} : la liste tabou de la fourmi μ (ensemble des sommets déjà traités par la fourmi μ)

Ω_i^{μ} : l'ensemble des **k** sommets qui sont reliés avec le sommet i par une chaîne de longueur ≥ 2 et non traités par la fourmi μ ;

ψ_i^{μ} : l'ensemble des **k** « meilleurs sommets » (en terme de phéromone) et non traités par la fourmi μ [26];

La solution de départ nous a permis de calculer la trace initiale de phéromone. A chaque arête $\{i, j\}$ de G' une valeur σ_{ij} définissant la quantité de phéromone. Cette valeur, peut être interprétée comme une longueur sur les arrêtes du graphe. Elle est dynamique et évolue après chaque passage de fourmis.

La mise à jour de la phéromone, au terme d'une colonie de f fourmis à travers le graphe $G' = (X,\ E')$, si les sommets i et j sont dans X_i (partition en stable) alors $\{i,\ j\} \in E'$, les phénomènes d'évaporation et de dépôt de la phéromone sur les arêtes du graphe G', sont mis en œuvre.

Sur chaque arête $\{i,\ j\}$ de E' le taux de phéromone est réduit : $\sigma_{ij} = \ell\sigma_{ij}$ si $\sigma_{ij} > \Phi_{\min}$.

$\Phi_{\min}$: taux minimum de phéromone, il permit de laisser une chance minimale à chaque arête (paire de sommets) d'être choisie [26].

Le taux de phéromone pour un chemin est renforcé sur toutes les arêtes reliant ses sommets (dans E'). Autrement dit, l'ensemble des sommets i et j tels que: $i, j \in X_i$ (X_i représente une partition en stable).

$$\sigma_{ij} = \begin{cases} \sigma_{ij} + \Delta\sigma_{ij} \ si\ i, j \in X_i \ et\ \sigma_{ij} < \Phi_{\max} \\ \sigma_{ij} \qquad\qquad \sin on. \end{cases} \quad \text{Avec}\ \Delta\sigma_{ij} = \frac{W_{ij} \times F - 1}{F \times (L^{\mu} + 1)}.$$

$\Phi_{\max}$ représente le taux maximum de phéromone.

En complément de $\Phi_{\min}$, il permet de laisser une chance à chaque paire de sommets d'être empruntée par une fourmis [26].

Au cours des itérations, la phéromone va s'évaporer un peu sur toutes les arêtes et va se renforcer de manière significative sur les arêtes (dans E') reliant les meilleurs sommets (en terme de, la longueur de la chaîne qui les reliés) de manière à concentrer les recherches des fourmis sur ces parties du graphe. Enfin, si un candidat au renforcement à une évaluation trop élevée, alors l'opération de dépôt de phéromone est inutile et sera ignorée [26].

Chaque solution construite par une fourmi est évaluée à l'aide de la fonction d'évaluation précédente en considérant le graphe G comme étant le sous graphe engendré par les sommets traités par cette fourmi.

En un sommet i, la fourmi se dirige soit en tenant compte des traces de phéromone qui sont déposées par les autres membres de la colonie qui la précèdent, soit de façon aveugle.

A chaque itération, s'opère le choix aléatoire d'une variable impliquée dans une contrainte non satisfaite. Pour cette variable, la fourmi est aveugle avec une probabilité P_{av}.

L'idée à retenir, concernant le choix de la variable suivante pour le déplacement aveugle, est de choisir le sommet j au hasard parmi les **k** sommets qui lui sont reliés par une chaîne de longueur ≥ 2 et non traités par cette fourmi ; elle tient en compte les traces de phéromone avec la probabilité ($1-P_{av}$) et elle choisit alors le prochain sommet j avec la probabilité P_{ij} [26], définie comme suit : $P_{ij} = \begin{cases} \sigma_{ij}\left(1-S_{ij}\right) \Big/ \sum_{q \in \psi_i^{\mu}} \sigma_{ij}\left(1-S_{ij}\right) \ si \ \ j \in \psi_i^{\mu}; \\ 0 \ \text{sinon.} \end{cases}$

Chaque solution trouvée par une fourmi subie une procédure d'amélioration en faisant des transformations, recherche locale (la solution testée est une solution voisine de la solution courante), avec une probabilité P_{ls}.

Le choix opéré au niveau de la transformation, perturbation, influe grandement sur la qualité des solutions procurées. Nous retenons pour la suite, les différentes transformations admissibles :

- Supprimer un sommet de la partition, déplacer un sommet x_i de l'ensemble X_i (la tournée i) à un autre ensemble X_j, $X_i' := X_i - \{x_i\}; X_j' := X_j \cup \{x_i\};$
- Ajouter un sommet à la partition, $X_i' := X_i \cup \{x_j\}, X_j' := X_j - \{x_i\};$
- Permuter deux sommets entre deux partition, $X_i' := X_i \cup \{x_j\} - \{x_i\}, X_j' := X_j \cup \{x_i\}$;

  Dans le cas de dégénérescence[1], nous faisons changer le nombre chromatique du graphe, en augmentant le nombre de partition en stable (couleurs), pour cela nous prenons un sommet x_i impliqué dans une contrainte non satisfaite et on le déplace de l'ensemble X_i vers l'ensemble X_{k+1}, $X_{k+1} := \{x_i\}$; et nous itérons le processus d'optimisation.

Une fois que toutes les fourmis d'une colonie ont traversé le graphe, nous classons les solutions obtenues.

[1] : La solution reste inchangée au bout de (n+i) itérations $\forall i \in IN^*$.

En terme de fonction d'évaluation, nous définissons une ***fourmi élitiste*** comme étant la meilleure solution retrouvée depuis le début de l'algorithme. L'ensemble des solutions initiales comprend en fait f_e fourmis élitistes et $f - f_e$ non élitiste [26].

Notre but sera donc de trouver un ensemble de règles pour déterminer les fourmis élitistes et les non élitistes. De ce fait, nous définissons la fonction « *éval* » à partir de la fonction coût par une transformation telle que : $éval(\mu_i) = m - f(\mu_i)$; dont *m* représente le nombre d'arêtes. Les étapes à suivre sont comme suit :

- Calculer la fonction $éval(\mu_i)$ pour chaque fourmi ;

- Calculer l'*éval* totale, *EV*, $EV = \sum_{i=1}^{f} éval(\mu_i)$;

- Calculer la probabilité p_i pour chaque fourmi μ_i tel que $p_i = éval(\mu_i)/EV$;

- Nous choisissons les fourmis élitistes et les non élitistes, en utilisant : $f_e = \left\{ \mu_i \ / \ p_i \geq \frac{\sum_{i=1}^{f} p_i}{f} \right\}$; $f_{\bar{e}} = \left\{ \mu_i \ / \ p_i < \frac{\sum_{i=1}^{f} p_i}{f} \right\}$.

Afin de privilégier les sommets qui a priori semblent mener à une solution plus prometteuse.

Lorsque la fourmi est élitiste, on cherche s'il existe une solution de S' voisine de la solution courante S telle que $Z(S') < Z(S)$, on se déplace alors de S vers S' qui devient ainsi la nouvelle solution courante pour cette fourmi, si la fourmi et non élitiste, on se déplace de S vers S' dans les deux cas ; amélioration ou bien dégradation.

Il apparaît clairement que, $Z(S) = 0$, si et seulement si S représente une coloration réalisable du graphe *G*. Notre objectif ultime sera donc de trouver une telle solution S. On peut donc fixer $Z(S) = 0$ et arrêter la recherche si l'on visite une telle solution.

Les principales équations et paramètres de la méthode, appliqués pour tenter la résolution du problème d'affectation sous contraintes, ont aidé à la construction de l'ossature de notre méthode ;
Plus de éclaircissements sur l'adaptation du principe d'ACO, voir Annexe.

1.2.1.2. Ossature basé sur le principe de colonie de fourmis [18]

{**Entrées** K : Le nombre de couleurs ; Maxit: Le nombre maximum d'itérations au delà duquel la solution ne peut être améliorée ; f : Le nombre de fourmis par colonie ; sol : solution de départ ; T_μ : la liste tabou de la fourmi μ ; T_j :une partition en stable, correspond à un chemin admissible}.

{**Sortie** une coloration : le nombre chromatique}.

Début

Etape (1):

Soltrouvée := faux; i:=0;

Tant que non soltrouvée *ou* $i \leq \max it$ *faire*

Pour j:=1 *à* f *faire*

Construire un chemin admissible T_j ;

Mettre à jour la liste tabou;

Si la fourmi est élitiste *alors*

Si évaluation (T_j)<évaluation(sol) *alors*

asol:= T_j;

Fsi;

Sinon sol:= T_j;

fsi ;

Fait

Si évaluation (sol)=0 *alors*

Soltrouvée := vrai;

Fsi;

Si non soltrouvée *alors*

Renforcer la phéromone sur les meilleurs chemins;

Faire évaporer la phéromone sur tous les autres chemins.

Fsi;

Fait

Si non soltrouvée *alors*

K:=k+1;

aller à l'étape (1);

Fsi;

Fin.

1.2.2. Amélioration par la méthode RT

Cette méthode combine une procédure de recherche locale avec un certain nombre de règles et de mécanismes permettant à celle-ci de surmonter l'obstacle des optima locaux, tout en évitant de cycler [15], [6].

1.2.2.1. Principe de base de la méthode RT

Elle consiste à se donner une solution de départ sur laquelle nous effectuerons des perturbations tabous et définir ainsi un voisinage de solutions, résultant de ces perturbations, dans lequel nous choisirons la meilleure et à laquelle nous appliquerons le même procédé défini précédemment pour définir un autre voisinage et nous itérerons le processus jusqu'à ce que nous atteindrons la solution adéquate que nous sauvegarderons en mémoire. L'application de telles perturbations taboues est interdite à toute solution courante.

a. Les tabous

Les tabous sont une manière de représenter la mémoire du cheminement effectué pour diriger l'exploration vers des régions non visitées.

La manière la plus simple de définir les tabous est de conserver une liste T. Nous gérons cette liste comme une liste circulaire : nous éliminons le plus vieux tabou et nous insérons la nouvelle solution [15], [6].

Nous pouvons alors définir les tabous en fonction des « transformations » permettant de passer d'une solution à une autre. Ainsi, dans le problème de la coloration, si l'on exclut x de V_i, on s'interdit de retransférer x dans V_i tant que cette opération est un tabou.

b. Exception aux interdictions

Il est possible de violer une interdiction lorsqu'un mouvement interdit permet d'obtenir la meilleure solution enregistrée.

1.2.2.2. Adaptation de la méthode R.T

Par cette méthode nous donnerons une résolution heuristique pour le problème d'affectation sous contraintes qui est approché par le problème de la coloration des sommets d'un graphe quelconque.

Soit $G = (V, E)$ un graphe, modélisant notre problème, avec V l'ensemble des sommets et E l'ensemble des arêtes. La stratégie consiste à obtenir une partition des sommets de celui-ci en des sous ensembles stables[1] et d'affecter à chaque stable V_i une couleur *i*.

Comme le problème de la recherche d'une telle partition revient à la recherche des stables maximaux de G ce qui constitue un problème NP-Complet [19]. Cependant nous sommes contraints d'initialiser notre méthode par une partition donner par la procédure $P^{\oplus}$ (l'existence dans la partition de sous ensemble non stable est permise).

Soit $(V_i)_{i\in I}$, $i = \overline{1,\ k}$ une telle partition de départ. Nous affecterons à chaque V_i la couleur C_i, $i = \overline{1,\ k}$, s'il existe un $i_0 \in I$ tel que V_{i_0} contient des paires (x, y) de façon que chaque paire est une arête de G alors nous effectuerons des transformations tabous consistant à déplacer l'un des sommets d'une même paire à un autre sous ensemble V_j tel que $i \neq j$ de façon que $d(x,z) \geq 2$ ou $d(y,z) \geq 2,\ \forall z \in V_j$. Nous itérerons le processus jusqu'à ce que aucun V_i ne contient de telles paires, ce qui nous donne la k-coloration de G ou d'aboutir à l'impossibilité de tels déplacements ce qui nous conduira à définir un autre sous ensemble V_{k+1} dont nous mettrons l'un des sommets d'une telle paire tout en excluant de telles paires dans V_{k+1} et nous itérons le procédé jusqu'à l'obtention d'une partition de G en *k* stable.

Plus de éclaircissements sur l'adaptation de la méthode TR, voir Annexe.

1.2.2.3 Ossature de la méthode RT

Définition des variables

s : La solution actuelle ;

s' : La solution obtenue à partir de s (solution voisine) ;

$N(s,\ h)$: L'espace de solutions voisines à s (l'ensemble des s') ;

s^* : La meilleure solution courante.

Début

Étape 1: Générer une solution initiale s par la procédure $P^{\oplus}$;

Appliquer $s^* := s$ et $h := 0$;

[1] Un stable est formé par un sous ensemble de sommets de V ne contenant aucune arête.

Étape 2 : Appliquer $h := h + 1$ et générer un sous-ensemble de solutions en $N(s,\ h)$ pour que :

- Les mouvements tabous ne soient pas choisis ;
- Un des critères d'aspiration soit applicable.

Étape 3 : Choisir la meilleure solution s' parmi l'ensemble de solutions voisines $N(s,\ h)$

Appliquer s : = meilleur s' ;

Étape 4 : Si $f(s) \leq f(s^*)$, alors nous avons trouvé une meilleure solution

Appliquer $s^* := s$;

Étape 5 : Mettre à jour la liste T et les critères d'aspiration ;

Étape 6 : Si une condition d'arrêt (*) est atteinte, stop.

Sinon, retour à Étape 2.

(*) : $f(s) = 0$; le nombre de contraintes violées est nul.

Fin.0

Chapitre 4

Caractérisation d'une Nouvelle Classe des Graphes Parfaits

Sommaire

Introduction

Dans ce chapitre, nous présenterons une nouvelle classe de graphes parfaits appelés G^*. Nous démontrerons le théorème fort des graphes parfaits pour cette classe. La coloration des graphes d'une telle classe se fait à l'aide de la procédure $P^{\oplus}$, présentée dans le chapitre 3, opérant en un temps polynomial.

L'objectif assigné à notre travail, au départ était d'apporté des éléments de résolution aux problèmes d'affectation sous contraintes, ce qui nous a conduit au développement d'une procédure de coloration polynomiale d'un graphe quelconque, procédure $P^{\oplus}$, suivi d'une propriété notée P^*.

L'analyse de l'ensembles des graphes coloriable d'une manière polynomiale par $P^{\oplus}$, nous a permis de le scinder en deux sous ensembles : le premier sous ensemble, vérifiant le théorème fort des graphes parfaits et la propriété P^* et l'autre sous ensemble non caractérisé.

Le premier sous ensemble que nous avons nommé G^*, inclut d'une manière stricte des classes connues des graphes parfaits telles que : les graphes biparti et les graphes de Berge : multiparti, sans griffe et sans patte.

Nous donnerons ci-dessous un cas particulier, presque trivial de la résolution des problèmes d'affectation sous contraintes.

Remarque

Si les graphes G_i modélisant les problèmes d'A. S. C sont des anti-trous respectivement des trous alors les solutions de telles instances de ces problèmes sont obtenues de façon linéaire. Du fait que la coloration de tels graphes est polynomiale [4].

1. Caractérisation de la nouvelle classe

Pour un graphe G connexe, notons par x^*, un sommet de G tel que : $d_G\left(x^*\right) = \underset{v\in V}{Max}\, d_G\left(v\right)$

1.1. Propriété P^*

Nous définissons la propriété P^* Par la non existence de paires de chaînes minimales, de longueur ≥ 2, de même parité, issues du sommet star choisi, ayant une arête reliant ses extrémités terminales ; de sorte que le sommet x^* choisi n'appartient à aucun voisinage d'un cycle élémentaire impair.

Nous définissons ainsi, la classe de graphes G^* comme suit :

1.2. Définition de la classe G^*

Un graphe *G* appartient à la classe G^* s'il vérifie la propriété P^* **.**

Proposition 1

Si $G \in G^*$ alors G parfait.

Preuve

Nous utilisons pour la démonstration de la perfection des graphes de la classe G^* le théorème fort des graphes parfaits [7], stipulant qu'un graphe est parfait si et seulement si il est de Berge, c.à.d n'admet ni de trou ni d'anti-trou impair de longueur ≥ 5.

Démontrons d'abord que G n'admet pas de trou impair de longueur ≥ 5.

Supposons que $G \in G^*$ et que G admet un trou impair de longueur ≥ 5. Nous distinguons alors trois cas :

1ier cas : x^* appartient à un trou impair de longueur ≥ 5.

Soit C_{x^*} un tel trou, c.à.d $C_{x^*} = x^* e_1 x_1 \ldots x_{p-1} e_p x^*$. Figure 1.

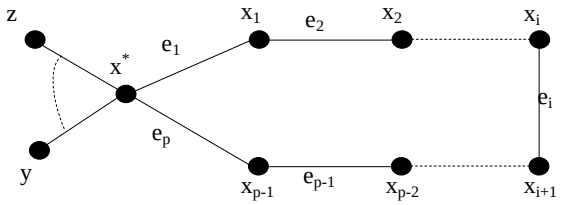


Figure. 1 : Le sommet star appartient au trou C_{x^*} **.**

Nous désignons par P_{xy} la chaîne minimale reliant le sommet x au sommet y.

Si les sommets x_i et x_{i+1} sont tous les deux dans le trou C_{x^*}, alors ceux-ci sont nécessairement reliés à x^* par deux chaînes minimales, de parité paire, distinctes, notées $P_{x^* x_{i+1}}$, $P_{x^* x_i}$ et qui sont séparées par l'arête (x_i, x_{i+1}).

Autrement dit, $\exists\, i \in I / I = \{1, \ldots, p-1\}$ tel que $(x_i, x_{i+1}) \in E$ avec $x_i, x_{i+1} \in C_{x^*}$.

Comme C_{x^*} est impair et $u=(x_i,x_{i+1})\in E$ et $P_{x^*x_{i+1}}$, $P_{x^*x_i}$ sont de même parité paire, et $C_{x^*}=P_{x^*x_{i+1}}\cup(x_i,x_{i+1})\cup P_{x^*x_i}$. Ainsi $u=(x_i,x_{i+1})\in E$ relie deux chaînes $P_{x^*x_{i+1}}$ et $P_{x^*x_i}$ de même parité implique que $G\notin G^*$, contradiction avec l'hypothèse de départ.

2eme cas : x^* n'est voisin à aucun sommet d'un trou impair de longueur ≥ 5.

Considérons maintenant le trou $C_{xp}=C_{2k+1}, k\geq 2$, avec $x_p\in V$, tel que $C_{x_p}=(x_p,e_1,x_1,...,x_{p-1},e_p,x_p)$, Figure 2.

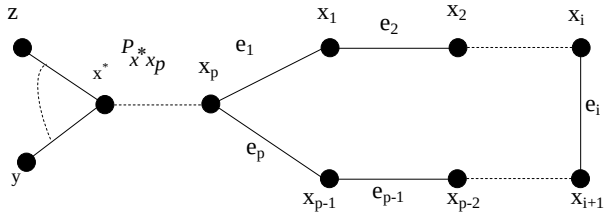


Figure. 2 : Le sommet star n'appartient pas au trou C_z.

Alors, $\exists\, i\in I/I=1,...,p-1$ tel que $(x_i,x_{i+1})\in E$ avec $x_i,x_{i+1}\in C_z$.

Ainsi, $P_{z\,x_i}$ et $P_{z\,x_{i+1}}$ sont de même parité paire car (x_i,x_{i+1}) est une arête du trou C_z.

Rappelons d'abord que $P_{x^*\,z}$ est la chaîne minimale qui relie le sommet star choisi au sommet z.

Par conséquent ; $P_{x^*x_i}=P_{x^*z}\cup P_{zx_i}$ et $P_{x^*x_{i+1}}=P_{x^*z}\cup P_{zx_{i+1}}$ avec $(x_i,x_{i+1})\in E$.

Comme les deux chaînes P_{zx_i} et $P_{zx_{i+1}}$ ont la même parité paire, donc $P_{x^*x_i}$ et $P_{x^*x_{i+1}}$ doivent être de même parité, sachant que $(x_i,x_{i+1})\in E$.

Donc la propriété P^* n'est pas vérifiée, ceci implique que $G\notin G^*$.

3eme cas : $x^*\notin C_{2k+1}$ et $\exists$ au moins un sommet $x\in C_{2k+1}$ tel que $(x^*,x)\in E$.

Il est trivial que $G\notin G^*$, et ceci car, le sommet star choisi est incident direct à un trou impair de longueur ≥ 5.

Des cas précédents on déduit que, si $G\in G^*$ alors G est sans trou impair de longueur ≥ 5.

Démontrons maintenant que si $G\in G^*$ alors G n'admet pas d'anti-trou impair de longueur ≥ 5.

Par l'absurde, supposons que $G \in G^*$ et G admet un anti-trou impair de longueur ≥ 5.

Soit $\overline{C}_1$ un tel anti-trou, notons par *m* le nombre d'arêtes dans celui-ci. Nous distinguons alors trois cas :

1ier cas : Supposons que le sommet star appartient à $\overline{C}_1$. Figure 3.

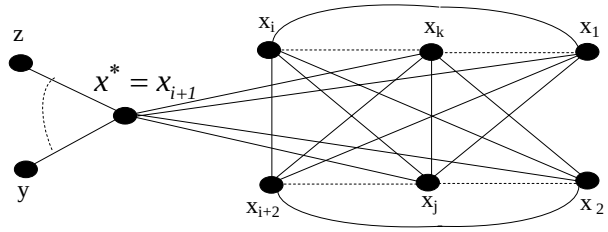


Figure. 3 : Le sommet star appartient à $\overline{C}_1$.

Tel que : $\begin{cases} k \neq j, j+1 \text{ et } j-1; \\ j \neq k+1 \text{ et } k-1. \end{cases}$

Rappelons que si $\overline{C}_1$ est un anti-trou alors $\forall x \in \overline{C_1}, d_{\overline{C}_1}(x) = m-3.$

Autrement dit, $\exists x_i, x_{i+1} \in V_1$ tel que $\left(x^*, x_{i+1}\right) \notin E_1$ et $\left(x^*, x_i\right) \notin E_1$ avec $\left(x_i, x_{i+1}\right) \in E_1.$

Il est clair que la chaîne minimale qui relie le sommet x^* choisi aux sommets x_i et x_{i+2} est de longueur ≥ 2, car x^* n'est pas voisin à ceux-ci.

Donc $\forall x \in \overline{C}_1 / \{x_i, x_{i+2}\}, \left(x^*, x\right) \in \overline{C}_1.$

Ainsi, $\exists x_1, x_2 \in \overline{C}_1$ tel que $\begin{cases} \left(x^*, x_1\right) \in \overline{C}_1; \\ \left(x^*, x_2\right) \in \overline{C}_1; \\ \left(x_i, x_1\right) \in \overline{C}_1; \\ \left(x_{i+2}, x_2\right) \in \overline{C}_1. \end{cases}$

D'où les deux chaînes minimales reliant le sommet x^* aux sommets x_i et x_{i+2} sont de longueur exactement égale à 2.

et ceci car $P_{x^* x_i} = (x^* e_1 x_1 e_2 x_i)$ et $P_{x^* x_{i+2}} = (x^* e_1' x_2 e_2' x_{i+2})$. *Contradiction*, car la propriété P^* n'est pas vérifiée.

2eme cas : $x^* \notin \overline{C}_1$ et $\exists$ au moins un sommet $x_{i+4} \in \overline{C}_1$, tel que $(x_{i+4}, x^*) \in \overline{C_1}$.

1. Si $\overline{C}_1 = \overline{C}_5$;

   $\overline{C}_5$ est un trou impair de longueur ≥ 5. Or, C_5 n'apparient pas à G^*.

2. Si $\overline{C}_1 \neq \overline{C}_5$;

   $\overline{C_1}$ est un anti-trou donc, $(x_i, x_{i+2}) \in \overline{C}_1, (x_i, x_{i+4}) \in \overline{C_1}$ et $(x_{i+2}, x_{i+4}) \in \overline{C_1}$, d'où vient que :

   $P_{x^* x_i} = (x^* e_1 x_1 e_2 x_i)$ est de longueur paire (egale à 2) et $P_{x^* x_i+2} = (x^* e_1' x_2 e_2' x_{i+2})$ l'est aussi à savoir que $(x_i, x_{i+2}) \in \overline{C_1}$.

   Ainsi l'existence de deux chaînes minimales de même parité, issues du x^*, ayant une arête reliant ses extrémités. De ce fait, la propriété P^* n'est pas vérifiée, donc $G \in G^*$.

3eme cas : x^* n'appartient pas à $\overline{C_1}$ et aucun sommet de $\overline{C_1}$ n'est voisin à x^*. Illustrer dans la figure suivante :

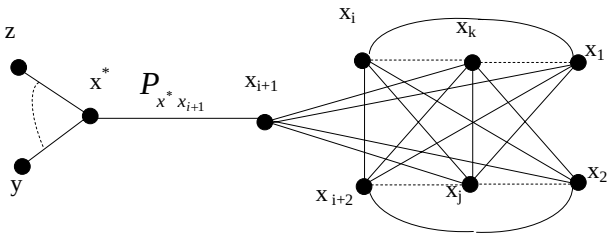


Figure. 4 : Le sommet star n'appartient pas au voisinage de $\overline{C_1}$.

Notons par $P_{x^* x_{i+1}}$ la chaîne minimale qui relie le sommet x^* au sommet x_{i+1} avec $x_{i+1} \in \overline{C}_1$;

1. Si $\overline{C}_1 \neq \overline{C}_5$ alors $\exists x_i, x_{i+2} \in \overline{C_1}$ de sorte que :

$$\begin{cases} (x_i, x_{i+2}) \in \overline{C}_1 ; \\ (x_{i+1}, x_i) \notin \overline{C}_1 ; \text{ car } x_i, x_{i+1}, x_{i+2} \text{ des sommets de } \overline{C}_1. \\ (x_{i+1}, x_{i+2}) \notin \overline{C}_1 \end{cases}$$

   D'où $P_{x_{i+1} x_i}$ et $P_{x_{i+1} x_i+2}$ sont de longueur égale ≥ 2.

Ainsi, les deux chaînes $P_{x_{i+1}x_i} = x_{i+1}\, e_1\, x_1\, e_2\, x_i$ et $P_{x_{i+1}x_{i+2}} = x_{i+1}\, e'_1\, x_2\, e'_2\, x_{i+2}$ sont de parité (égale à 2) tel que $(x_i, x_{i+2}) \in \overline{C_1}$.

Donc $P_{x^* x_i} = P_{x^* x_{i+1}} \cup P_{x_{i+1}x_i}$ et $P_{x^* x_{i+2}} = P_{x^* x_{i+1}} \cup P_{x_{i+1}x_{i+2}}$.

Comme $P_{x_{i+1}x_i}$ et $P_{x_{i+1}x_{i+2}}$ sont de même parité alors, $P_{x^* x_i}$ et $P_{x^* x_{i+2}}$ doivent être de même parité tel que $(x_i, x_{i+2}) \in \overline{C_1}$, ce qui contredit la non existence de deux chaînes de même parité ayant une arête reliant ses extrémités donc $G \notin G^*$.

2. Si $\overline{C}_1 = \overline{C}_5$ alors $\overline{C}_5$ est un $\overline{C}_5$. *Ce qui contredit l'hypothèse précédente.*

D'où, si $G \in G^*$ alors G est sans anti-trou impair de longueur ≥ 5. Ceci termine la démonstration de la perfection de la classe des graphes de G^*.

Résultat

Nous déduisons que si G vérifie la propriété P^* alors G est parfait et la propriété de Berge est implicite.

1.3. Justification de la coloriabilité optimale de la classe G^* par $P^{\oplus}$

La coloration des graphes contenant des paires de chaînes minimales, de parité paire, issues du sommet star choisit, ayant une arête reliant ses extrémités finales, par notre procédure procure des solutions heuristiques du fait de la violation des contraintes pour de tels graphes. Hors cette condition, $P^{\oplus}$ donne des solutions qui convergent vers la solution optimale. Cependant, notre classe G^* est définie par la non existence de telles paires de chaînes et par d'autres sans que le sommet x^* ne soit voisin d'aucun sommet appartenant à un voisinage d'un cycle élémentaire impair. Donc la première classe des graphe qui sont coloriable par $P^{\oplus}$ est plus large que celle de G^*, ce qui prouve que G^* est coloriable de manière polynomiale par $P^{\oplus}$.

2. La classe G^* et les autres classes des graphes parfaits

Notre analyse nous a permis d'identifier que certaines classes de graphes parfaits sont contenues strictement dans la classe G^* et d'autres non.

2.1. Les classes strictement incluses dans la classe G^*

La classe des graphes biparti, et de Berge multiparti, sans patte, et sans griffe sont inclues strictement dans G^*.

Preuve

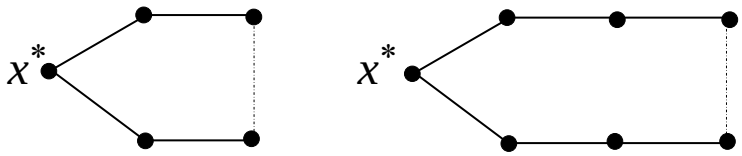
1. La classe des graphes biparti est dans la classe des graphes de G^*

Un graphe biparti est un graphe ne contenant aucun cycle impair.

Preuve

La classe des graphes biparti est incluse strictement dans la classe G^* et ceci en se basant sur la définition des graphes biparti, donc tous les cycles de G sont de longueur paire. Autrement dit, $\forall x,\ y \in V$, toutes les chaînes reliant le sommet x au sommet y ont la même parité paire.

S'il existe deux chaînes minimales de même parité (paire ou impaire), voir la figure ci-dessous, issues d'un sommet x de G, G biparti, tel que ses deux extrémités seront reliées par une arête, alors dans ce cas le graphe G doit avoir un cycle impair.



De ce fait, un graphe biparti interdit l'existence de telles paires de chaînes pour n'importe quel sommet de G, donc le sommet x^*. Ainsi, la propriété P^* est vérifiée pour un graphe biparti. D'où la classe des graphes biparti est incluse dans la classe des graphes de G^*.

Considérons maintenant le graphe complet K_3. Chaque sommet de K_3 peut être considéré comme étant un sommet x^*. Dans K_3 le sommet x^* choisi, étant le sommet x_1, est un sommet voisin des sommets x_2 et x_3. Il est évident qu'un tel graphe, K_3, vérifie la propriété P^*, tous les sommets sont voisin de celui-ci, donc K_3 appartient à G^* sans qu'il soit biparti et ceci du fait que le graphe G contient un cycle impair (de longueur égale à 3).

Il s'ensuit, tous les graphes biparti vérifient la propriété P^* et qu'il existe au moins un graphe G appartenant à G^* (le K_3 est un exemple) et n'appartenant pas à la classe des graphes biparti.

Conclusion 1

Nous concluons alors que la classe des graphes biparti est inclue strictement dans la classe G^*.

2. La classe des graphes de Berge multi parti est dans la classe des graphes de G^*

Un graphe de Berge multi parti est un graphe ne contenant aucun cycle impair.

Preuve

Un graphe G est de Berge multi parti si et seulement si G ne contenant pas de cycle impair, donc toutes les chaînes reliant le sommet x au sommet y ont la même parité paire, ainsi toutes les chaînes minimales issues du sommet x^* ont la même parité paire. Ce qui est dû à la non existence de deux chaînes minimales de même parité qui sont reliées par une arête. Par conséquent, la classe des graphes de Berge multi biparti est incluse dans la classe G^*.

Par ailleurs, la classe des graphes de Berge multi parti n'admet pas le graphe G qui est représenté par un $K_{1,3}$ comme un sous graphe induit, or que, $K_{1,3}$ appartient à G^*. Ce qui justifier l'inclusion stricte.

Conclusion 2

La classe des graphes de Berge multi parti est incluse strictement dans la classe G^*.

3. La classe des graphes de Berge sans patte est dans la classe des graphes de G^*

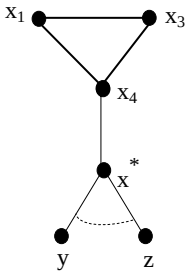
Un graphe de Berge sans patte est un graphe ne contenant pas de patte.

Preuve

Etant donné un graphe G parfait et un sommet x^* de G.

Supposons que G est un graphe de Berge sans patte et G n'appartenant pas à G^*. Donc, supposons que n'appartenant pas à G^*, pour cela nous distinguons deux cas possibles :

1^ier cas : x^* est voisin à un seul sommet d'un cycle élémentaire impair donc il est de longueur 3, car *G* est parfait, donc ce cas il existe deux chaînes minimales de même parité (de longueur 2), issues du x^* , ayant une arête reliant ses extrémités $\left(G \notin G^*\right)$.

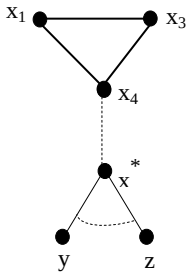


Le sommet x^* est voisin d'un sommet de K_3

Donc G contient une patte. Ce qui contredit l'hypothèse que G est de Berge sans patte.

2^eme cas : x^* n'appartient à aucun voisinage d'un cycle élémentaire impair.

G n'est pas dans G^*, donc il existe au moins deux chaînes minimales de même parité, issues du sommet x^* choisi, à distance ≥ 2, reliant par une arête ses extrémités. L'existence de telles chaînes signifie l'existence d'un cycle élémentaire impair donc de longueur 3, sinon G n'est pas parfait, tel que x^* n'est voisin d'aucun sommet de celui-ci.



Le sommet x^* n'est voisin d'aucun sommet d'un K_3

Ainsi, la chaîne minimale reliant le sommet x^* aux sommets de K_3 est de longueur ≥ 2 donc il existe au moins un sommet *x* de la chaîne reliant x^* aux sommets de K_3, qui est voisin à au moins un sommet de K_3. D'où l'existence d'une patte. Contradiction.

De ces deux cas, nous déduisons que la classe des graphes sans patte est incluse dans la classe des graphes de G^*.

Considérons maintenant un graphe G qui est représenté par une patte, celui-ci appartient à G^* sans qu'il soit de Berge sans patte. Ce qui justifie l'inclusion stricte de la classe G^*.

Conclusion 3

La classe des graphes de Berge sans patte est incluse strictement dans la classe G^*.

4. La classe des graphes sans griffe est dans la classe des graphes de G^*

Un graphe de Berge sans griffe est un graphe ne contenant pas de griffe.

Preuve

Cette classe de graphes est incluse dans la classe des graphes de Berge sans patte et comme cette dernière est inclue strictement dans la classe G^*, donc la classe des graphes de Berge sans griffe est incluse, strictement, dans la classe G^*.

Conclusion 4

La classe des graphes de Berge sans griffe est incluse strictement dans la classe G^*.

Les conditions ci-dessus démontrent la proposition précédente.

2. 2. Comparaison avec d'autres classes de graphes parfaits

2.2.1. La classe des graphes i-triangulé diffère de la classe des graphes de G^*

Un graphe est i-triangulé si tout les cycle impair de longueur au moins cinq possède deux cordes qui ne se croisent pas.

Preuve

Soit G un graphe, tel que G contient un cycle impair de longueur égale à 5 admettant deux cordes qui se croisent, donc n'importe quel sommet x^* choisi dans G la propriété P^* est vérifiée $G \in G^*$, il est clair qu'un tel graphe n'appartient à la classe des graphes i-triangulé. Donc $G \in G^*$ et $G \notin$ i-triangulé. Figure1.

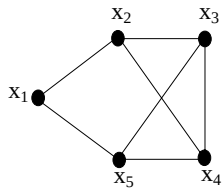


Figure. 1 : Cycle de longueur égale à 5 avec deux cordes qui se croisent.

Soit maintenant un graphe G contenant un seul cycle impair de longueur égale 3 tel que le sommet x^* choisi est voisin à au moins un de ses sommets, dans ce cas G n'appartient pas à G^*, or G est i-triangulé. Donc $G \notin G^*$ et $G \in$ i-triangulé. Figure 2.

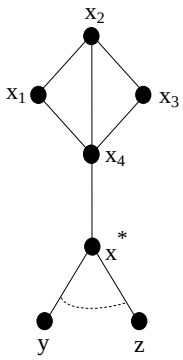


Figure. 2 : Un graphe qui admet une patte comme un sous graphe induit.

Ainsi, $\begin{cases} \exists G \in G^* \text{ et } G \notin \text{i-triangulé} \\ \text{et} \\ \exists G \notin G^* \text{ et } G \in \text{i-triangulé} \end{cases}$ démontre que ces deux classes sont différentes.

2.2.2. La classe des graphes Triangulé diffère de la classe des graphes de G^*

Un graphe triangulé est un graphe dont tout cycle de longueur supérieur ou égale à quatre admet une corde.

Preuve

Considérons le graphe G comme étant un trou pair. Donc toutes les chaînes reliant le sommet x_i au sommet x_j tel que $i \neq j$ ont la même parité. Alors G ne doit y pas avoir deux chaînes minimales de même parité qui sont reliées par une arête, donc la propriété P^* est vérifiée ($G \in G^*$) mais G n'est pas dans la classe des graphes triangulés. Figure 3.

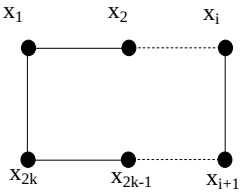


Figure. 3 : Un trou pair.

Considérons maintenant le cas où le graphe G contient un seul cycle élémentaire impair donc de longueur égale 3, de sorte que le sommet x^* choisi est voisin à un sommet de celui-ci et G sans trou pair de longueur ≥ 4. Il est clair que G est dans la classe des graphes triangulé, néanmoins G n'est pas dans G^*. Figure 2.

2.2.3. La classe des graphes de Meyniel diffère de la classe des graphes de G^*

Un graphe est de Meyniel si tout les cycle impair de longueur au moins cinq possède deux cordes.

Preuve

Soit G un graphe tel que G contient un cycle impair de longueur égale à 5 possédant une seule corde. Donc il est évident que G n'est pas de Meyniel. N'importe quel sommet x^* choisi dans G la propriété P^* est vérifiée, donc G est dans G^* et G n'est pas de Meyniel. Figure 4.

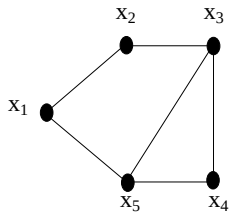


Figure. 4 : Cycle de longueur égale à 5 avec une corde.

Etant donné un graphe G contenant un seul cycle élémentaire impair donc de longueur 3, sinon G n'est pas parfait, et un sommet x^* de G. supposons que x^* n'appartient pas à ce cycle, pour un tel graphe la propriété P^* n'est pas vérifiée. Cependant, G est de Meyniel (voir Figure 2). Donc G est de Meyniel et G n'est pas de G^*. Ce qui démontre la proposition précédente.

2.2.4. La classe des graphes d'Intervalle diffère de la classe des graphes de G^*

Un graphe G est d'intervalle si et seulement si G ne contient aucun trou.

Preuve

Etant donné un graphe G qui est représenté par un trou pair, un tel graphe n'est pas d'intervalle et ceci en se basant sur la définition des graphes d'intervalle, tandis que la propriété P^* est vérifié dans G, autrement dit ; G appartient à G^* et G n'appartient pas à G^* (voir Figure : 3).

Cependant, il existe un graphe G dont le sommet x^* est voisin à un sommet appartenant à un cycle impair de longueur 3, à savoir que, G ne possède pas un trou pair, donc G est d'intervalle, mais il n'est pas dans G^* (voir Figure : 2).

2.2.5. La classe des graphes de Parité diffère de la classe des graphes de G^*

Un graphe est de parité si pour toute paire de sommets non adjacents, toute chaîne minimale les reliant est de longueur paire.

Preuve

Etant donné un graphe G qui contient un seul cycle impair de longueur égale à 5 possédant une corde. Dans ce graphe, il existe deux sommets non adjacents, x et y qui sont reliés par deux chaîne de parités différentes, donc G n'est pas de parité, cependant, G appartient à G^* (voir la Figure 4).

Considérons maintenant un graphe G dont il contient un seul cycle impair K_3, où le sommet x^* est voisin à un sommet de K_3, il est évident que G n'appartient pas à G^* tandis que G est de parité (voir Figure : 2).

2.2.6. La classe des graphes de Berge Sans anneau diffère de la classe des graphes de G^*

Preuve

Considérons un graphe G qui est représenté par un anneau. Pour n'importe quel sommet x^* choisi, la propriété P^* est vérifiée, donc G appartient à G^* sans qu'il soit de Berge sans anneau (voir la Figure 5).

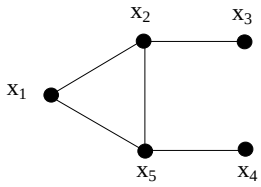


Figure. 5 : Un anneau.

Soit maintenant un graphe G, où le sommet x^* est voisin à un sommet d'un cycle élémentaire impair donc de longueur 3 ainsi, G n'admet pas un anneau comme sous graphe induit, d'où G appartient à la classe des graphes sans anneau tandis qu'il n'appartient pas à G^* (voir la Figure 2).

2.2.7. La classe des graphes de Berge sans diamant diffère de la classe des graphes de G^*

La classe des graphes de Berge Sans diamant diffère de la classe des graphes G^*.

Preuve

Etant donné un graphe G qui est représenté par un diamant, donc le sommet x^* choisi est voisin aux autres sommets de G, ce qui conduit à la non existence de deux chaîne

minimales, de même parité, issues du sommet x^* ayant une arête reliant ses extrémités. Pour un tel graphe la propriété P^* est vérifiée, donc G appartient à la classe G^* tandis qu'il n'appartient pas aux graphes de Berge sans diamant Figure 6.

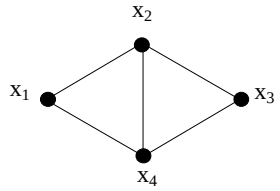


Figure. 6 : Un diamant.

Considérons maintenant, le graphe G dont le sommet x^* est voisin à un sommet d'un cycle de longueur 3. le graphe G étant dans la classe des graphes de Berge sans diamant, néanmoins G n'est pas dans la classe des graphes de G^* (voir la Figure 2). Ce qui justifié la différence entre ces deux classe.

2.2.8. Les graphes faiblement triangulé diffère de la classe des graphes de G^*

Un graphe faiblement triangulé s'il ne contient pas de trou ni d'anti-trou d e longueur supérieur ou égale à cinq.

Preuve

Considérons un graphe G qui est représenté par un trou pair de longueur égal à 6, celui-ci appartient à la classe G^* mais non aux faiblement triangulés, ceci est dû à la non existence de deux sommets dans G reliés par deux chaînes de parité différente, autrement dit : Pour chaque deux sommets x et y les chaînes les reliant sont toutes de même parité, ce qui induit l'exclusion d'une arête entre chaque deux chaînes minimales de même parité incidente à un sommet x^* quelconque de G. Figure 7. Donc $G \in G^*$ et $G \notin$ faiblement traingulé.

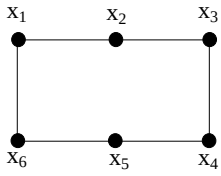


Figure. 7 : Un trou de longueur 6.

Par ailleurs, le graphe G dont les voisins du sommet x^* contient un sommet appartenant à un K_3 est faiblement triangulé mais n'appartient pas à G^* (voir la Figure 2), donc résulte la différence entre les deux classes des G^* et des faiblement triangulés.

2.2.9. Les graphe de Berge sans dart diffère de la classe des graphes de G^*

La classe des graphes de Berge sans dart diffère de la classe des graphes G^*.

Preuve

Etant donnée un graphe G représenté par un dart, le sommet x^* est voisin aux sommets de G, ce qui interdit l'existence de deux chaînes minimales, issues du sommet x^*, de longueur ≥ 2, donc la propriété P^* est vérifiée. Par conséquent, G appartient à G^* mais non aux graphes de Berge sans dart. Figure 8.

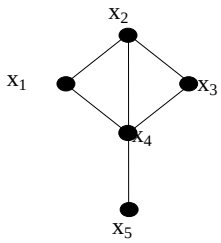


Figure. 8 : Un dart.

D'autre part, le graphe G dont le sommet star est voisin à un sommet appartenant à K_3, sans qu'il soit incident à un diamant, est de Berge sans dart tandis qu'il n'est pas dans la classe G^* (voir la Figure 2). Ceci justifie la différence entre ces deux dernières classes.

Dans notre démonstration nous avons opté à l'utilisation du théorème fort des graphes parfaits stipulant que tout graphe de Berge est parfait. Cependant, nous avons montré la non existence de trou ou d'anti-trou (le *ou* est exclusif) pour un graphe appartenant à notre classe.

Notre démarche a été motivée d'une part, du fait que ce théorème a été démontré en 2002 [7]. D'autre part, par l'inexistence d'une classe de graphe parfait connue contenant notre classe.

Chapitre 5

Exemple d'application et implémentation des résultats

Sommaire

1. Problème proposé

Nous supposons dans ce qui suit une modélisation et résolution d'une situation pouvant se poser pour un centre de recherche en ce qui concerne le problème de disposition et d'affectation de ses produits chimiques dans sa banque de stockage.

1.1. Description du problème supposé

Nous supposons qu'un centre de recherche en chimie moléculaire et industrielle est composé de plusieurs laboratoires. Chaque laboratoire utilise de façon aléatoire plusieurs produits chimiques, que nous supposons dangereux, certains de ces produits sont partagés entre les différents laboratoires, tandis que d'autres sont spécifiques à quelques utilisateurs uniquement. Nous nous plaçons dans le cas où la manipulation de tels produits nécessite une grande prudence, ceci est dû à des risques de plusieurs dangers pouvant être le résultat d'une erreur de mesure, mais souvent d'un mélange de produits incompatibles (contre-indiqués). Cependant, les conditions de stockage de ces derniers ne sont pas une chose simple. En effet, un mélange d'un produit avec un autre peut provoquer un incendie ou une explosion, de graves maladies, des pollutions fatales et beaucoup d'autres dégâts, souvent inestimables. De ceci, découle certaines conditions de stockage ou de conservation. Ainsi, une grande banque est mise à la disposition du centre pour une telle tache.

Cependant, nous supposons que la diversité des produits et des propriétés les régissant font appelle à des conditions très spécifiques de stockage. Supposons d'une part, que certaines de ces conditions sont relatives à la température et d'autres à l'interaction entre ces différents produits, alors que d'autres sont relatives au même temps à la température et à ces différentes interactions et d'autre part, une famille de ces produits est constituée des produits devant être dans des températures bien déterminées, une autres de ceux devant être à l'abri de la lumière et la dernière famille que nous considérons est celle constituée des produits devant être dans un isolement total ou partiel avec d'autres produits.

La satisfaction de telles conditions nécessite des moyens très conséquents et onéreux pour le centre, les responsables le savent bien qu'ils ne peuvent négliger aucune de ces conditions et pour l'exclusion de tout risque la satisfaction de celles-ci devient obligatoire. De ce fait, il est donc naturel de penser à des dispositions optimales de ces différents produits dans les différents coins adéquats de la banque et ceci en utilisant

un minimum de compartiments coûteux[1], tout en respectant les différentes conditions déjà citées.

D'une part, supposons que la banque en question est constituée de plusieurs compartiments, chacun de ceux-ci contient un nombre bien déterminé d'étagères, deux compartiments peuvent avoir bien entendu un même nombre d'étagères et chacune de ces dernières est faites de niveaux. Certains compartiments répondent à des conditions de température, d'autre à l'isolements des différents produits aux interactions risquées et d'autres encore au maintient de certains produits à l'abri de la lumière.

Et d'autre part, comme déjà cité, les produits sont d'une nature complexe, au point ou certains produits ne peuvent même pas être dans un même compartiment avec d'autres, certains d'autres ne peuvent pas être dans un même niveau, voir même un même étagère, pendant que d'autres ne peuvent se trouvés côte à côte.

Ainsi tout revient donc à chercher une affectation optimale de ces produits dans les compartiments, dans les étagères, dans les nivaux et encore une meilleure disposition de ceux-ci dans chaque niveau, tout en respectant les conditions de stockage dictées par leurs propriétés, éviter tout risque de danger ou de décomposition des produits et minimiser les frais de stockage pour le centre.

1.2. Approche de modélisation

La modélisation de ce problème peut se faire par le problème d'affectation sous contraintes et ceci en établissant des lien entre les contraintes intervenant dans l'affectation de ces produits et les propriétés des graphes, pouvant se définir et se résoudre par l'approche de la coloration adoptée le long de cette thèse.

1.2.1. Définition des contraintes

En considérant deux produits p et q, ne pouvant être dans un même compartiment, niveau, ou étagère ou côte à côte, comme étant deux sommets d'un graphe, alors cette contrainte peut être vue comme étant la contrainte d'adjacence, déjà citée dans ce travail.

La contrainte de la minimisation des compartiments, des étagères utilisées peut être vue comme étant le nombre chromatique d'un certain graphe G. En fin la contrainte

[1] Un compartiment est dit coûteux si les frais de son utilisation s'ont chers.

d'affectation de chaque sous ensemble de produits dans le compartiment idéal n'est autre que les sous ensembles de sommets de même couleur d'un graphe modélisant cette contrainte.

1.3. Approche de résolution

La modélisation et la résolution de ce problème peut se faire principalement en quatre grandes parties (bloque de graphes).

Dans la première partie, on cherche le nombre de compartiments minimum ainsi que l'affectation optimale, parmi les différentes affectations possibles, des produits dans ceux-ci. Il est à signaler qu'il peut exister deux sortes de contraintes d'affectation des produits dans les compartiments, la première est que certains produits doivent impérativement être dans des compartiment leurs offrant les bonnes conditions de stockage, tandis que la deuxième est celle de trouver le bon compartiment, en nombre d'étagères, qui contiendra le sous ensemble de produits pouvant être à l'intérieur de celui-ci. Ces deux dernières s'ajoutent à la contrainte de minimisation du nombre de compartiments à utiliser.

Dans la seconde partie, on s'intéressera à la recherche d'une affectation des produits, pouvant être dans un même compartiment, dans les différentes étagères de celui-ci et ceci pour chaque compartiment.

La troisième partie traite le problème d'affectation des produits, pouvant se trouver dans une même étagère dans les différents niveaux de celle-ci, un travail que nous ferons pour chaque sous ensemble de produits pouvant être dans la même étagère.

En fin, dans la quatrième et dernière partie, nous traiterons, pour chaque niveau prévu pour une utilisation, de ce qui est de la disposition des produits dans un même niveau.

1.3.1. Partie I

Cherchons en premier le nombre minimum de compartiments à utiliser. Pour se faire considérons l'ensemble de tous les produits, possédé par le centre et une bijection f, qui fait correspondre à chaque produit un point du plan que nous considérons comme sommets d'un graphe qu'on notera $G_C(E,V_C)$. Deux sommets de G_C sont reliés entre eux si et seulement si les deux produits correspondant ne peuvent être dans le même compartiment. Cependant le nombre minimum recherché est exactement le nombre minimum de couleur nécessaire pour colorier les sommets de G_C sans que deux

sommet de même couleur ne soit adjacents, Min (des compartiments)$=\gamma(G_C)$. De ceci vient que tous les produits dont les sommets correspondant ont une même couleur peuvent se placer dans un même compartiment si d'autres contraintes ne s'opposent pas (voir les autres conditions de stockage).

Cherchons maintenant pour chaque sorte de compartiments le nombre minimum à utiliser de ceux-ci (les compartiments sont repartis selon les conditions de stockage qu'ils peuvent offrir aux produits, ici on ne s'intéressera qu'aux trois sortes déjà citées) et pour chaque sorte faisant ce qui suit :

Considérons le sous ensemble de produits nécessitant le traitement qu'offre les compartiments, d'une des trois sortes. Parmi les produits de ce sous ensemble il existe certainement ceux qui ne peuvent être dans le même compartiment. Pour parer à ce problème considérons un graphe qu'on notera G_{Si}, dont l'ensemble des sommets est en bijection avec les produits du sous ensemble exigeant un traitement, noté i et deux sommets de G_{Si} sont adjacents si les produits leurs correspondant ne peuvent pas être dans le même compartiment, ainsi le nombre minimum recherché pour la sorte en question correspond au nombre chromatique de $G_{Si}\ (\gamma\ (G_{Si}))$.

Une fois avoir fait ce travail pour chaque sorte de compartiments, on obtiendra une affectation optimale de chaque sous ensemble de produits pouvant être dans le même compartiment et ceci par le choix des sommets de même couleur du graphe modélisant la contrainte et nous obtiendront :

$\gamma(G_C)=\gamma(G_{S1})+\gamma(G_{S2})+\gamma(G_{S3})+...$ où G_{Si} est le graphe modélisant la recherche du nombre de compartiments offrant le traitement i.

Une fois que ces sous ensembles de produits sont obtenus, attribuons à chaque produit p le numéro du compartiment dont il se trouve, de sorte que chaque produits existant dans le centre sera noté par p_i avec $i=\overline{1,\gamma(G_C)}$.

1.3.2. Partie II

Dans cette partie nous chercherons une affectation optimale des produits d'un même compartiment dans les différentes étagères, vu que certains produits ne peuvent être dans une même étagère. La modélisation de cette contrainte s'est faite, pour chaque compartiment, à l'aide de $\gamma(G_C)$ graphes distincts, noté G_{ECi} avec $i=\overline{1,\gamma(G_C)}$. En

effet, pour chaque produits pouvant être dans un même compartiment, on associe un ensemble de sommets, de sorte que chaque deux sous ensembles de produits pouvant être respectivement dans les compartiments i et j, notés C_i et C_j $i \neq j$ avec i et j dans $\{1,2,...\gamma(G_C)\}^2$, on fait correspondre deux ensembles de sommets distincts X_i et X_j. Deux sommets d'un même ensemble $X_i, i = \overline{1,\gamma(G_C)}$, sont adjacents dans G_{ECi} si et seulement si les deus produits leur correspondant ne peuvent pas être dans la même étagère. Chercher une meilleure disposition des produits du compartiment C_i dans les différentes étagères de celui-ci revient exactement à la recherche d'une coloration optimale du graphe G_{ECi}. En effet, une fois les sommets de ce graphe sont coloriés, de façon optimale, il ne reste qu'a considérer chaque produits dont les sommets correspondant dans G_{ECi} sont coloriés par une même couleurs comme étant le sous ensemble de produits, des produits du compartiment i, pouvant être sur la même étagère. Ainsi pour chaque compartiment i il faut $\gamma(G_{ECi})$, avec $i = \overline{1,\gamma(G_C)}$, étagères différentes pour stocker les produits pouvant être dans ce compartiment. A chaque produit du centre noté dans la première partie par p_i où i est le numéro du compartiment, attribuant un autre indice j pour que chaque produit du centre sera noté par p_{ij}, où j est le numéro de l'étagère du compartiment i, $i = \overline{1,\gamma(G_C)}$ et $j = \overline{1,\gamma(G_{ECi})}$.

1.3.3. Partie III

Pour chaque sous ensemble de produits pouvant être sur la même étagère faisant ce qui suit pour trouver une meilleure disposition de ces derniers sur les différents niveaux de cette étagère. Nous décrirons ici un travail qui se fera sur une des étagère j d'un compartiment i, mais cette procédure se répétera pour les $\gamma(G_C)$ compartiments et pour chacune des $\gamma(G_{ECi})$ étagères d'un compartiment i. ainsi celle-ci se fera $\sum_{i=1}^{\gamma(G_C)} \gamma(G_{ECi})$ fois. Considérons donc un sous ensemble de produits d'une même étagère j et faisant correspondre à chacun de ceux-ci un sommet, ceci de façon bijective. Deux sommets seront adjacents s'ils ne peuvent être stocké dans un même niveau de cette étagère, de ceci on obtiendra un graphe qu'on notera G_{Nj} dont la

coloration nous donne une disposition de ces produits sur les différents niveaux. En effet, il suffit de placer tous les produits, dont les sommets correspondant dans $G_{N\,j}$ sont de même couleur, sur un même niveau de l'étagère j. Attribuons un autre indice, pour chaque produits du centre déjà noté $p_{i\,j}$, k correspondant au niveau de l'étagère j du compartiment i Avec $i = \overline{1,\gamma\left(G_C\right)}$, $j = \overline{1,\gamma\left(G_{EC\,i}\right)}$ et $k = \overline{1,\gamma\left(G_{N\,j}\right)}$.

1.3.4. Partie IV

Cherchons en fin la disposition des produits de chaque niveau, en tachant d'éviter tout frottement interdit entre ceux-ci.

Comme dans la partie précédente la procédure décrite ici se fera pour chaque niveau de chaque étagère et dans chaque compartiment.

Associons à chaque produit d'un même niveau un sommet, et deux sommets de seront reliés si les deux produits correspondant ne peuvent être côte à côte sur le niveau k de l'étagère j du compartiment i. Ainsi on obtient un graphe, noté $G_{PN\,k}$, dont la coloration de ses sommets nous donne la position des produits dans le niveau k. Cependant, il suffit de placer côte à côte dans le même niveau tous les produits dont les sommets correspondant dans $G_{PN\,k}$ sont coloriés par une même couleur. Attribuant un autre indice t, dit de position à chacun des produits, déjà notés dans les parties précédentes, du centre, cet indice est déterminé par rapport un point de produit de ce niveau, fixé comme repère, par exemple un produit de grande utilisations peut jouer ce rôle et t est pair si le produit est à gauche de ce repère, impair sinon, multiple de 3 s'il est à gauche est avant et multiple de 2 sinon ...Cependant, chaque produit du centre se trouve notés par $p_{i\,j\,k\,t}$ (t peut être aussi défini comme étant autre chose).

Par ceci, nous venons de donner des situations envisageables pour le problème de la disposition des produits chimiques dans une banque de stockage d'un centre de recherche, en évitant tout risque de danger et en minimisant les frais dus au stockage. En ce qui concerne la résolution (la coloration des graphes modélisant les situations en question) nous adoptons l'approche développée dans cette thèse, à savoir la coloration par $P^{\oplus}$, dans le cas d'un graphe G^{*}, ou par notre méthode hybride intégrant cette procédure.

Nous tenons à signaler que les notations des produits utilisées dans cet exemple peuvent de plus servir comme système de référence de la gestion[1] des produits dans la banque pour éviter tout encombrement ou confusion en cas de besoin.

Remarque

Le nombre de compartiment minimum ne dépasse pas le nombre de conditions de stockage mis à part les conditions d'adjacences.

2. Utilisation de notre méthode

Soit I le nombre minimum du compartiment nécessaire pour stocker N produits, J_i le nombre minimum d'étagère dans chaque compartiment i et K_{j_i} le nombre minimum de niveaux à définir pour chaque étagère j du compartiment i.

Considérons le cas où l'unité de stockage dispose de $N = 40$ produits de différents types tels que :

Remarque

Les produits ne figurant pas dans le tableau sont ceux qui s'adaptent à toutes les conditions (sans contraintes).

Q_1 produits de type x_1 ;

Q_2 produits de type x_2 ;

...

Q_{40} produits de type x_{40}.

Le centre envisagé cherche à trouver une disposition optimale de ces produits, tout en respectant les conditions de stockages déjà cités, dans des compartiments, étagères et niveaux que doit posséder l'unité de stockage du centre de recherche.

Pour parer à tout danger ou pourrissement des produits, on suppose les conditions spécifiques de stockage de ces derniers énoncées dans le tableau suivant :

[1] : La mise à jour (suppression et ajout).

	abri de la lumière forte.	abri de la lumière faible.	abri de la lumière moyenne.	Température élevée.	Température faible.	Température moyenne.
x_1			X			
x_2						X
x_3		X			X	
x_4		X				
x_5	X					X
x_6					X	
x_7			X			X
x_8		X		X		
x_9	X					
x_{10}	X				X	
x_{11}				X		
x_{12}	X			X		
x_{14}	X				X	
x_{15}	X					X
x_{16}		X		X		
x_{17}	X			X		
x_{18}						X
x_{19}	X				X	

x_{20}	X					
x_{21}			X			X
x_{22}	X			X		
x_{23}				X		
x_{24}						X
x_{25}	X			X		
x_{26}	X					X
x_{27}		X				
x_{28}	X				X	
x_{30}		X				
x_{31}			X			X
x_{32}	X			X		
x_{33}					X	
x_{34}				X		
x_{35}		X		X		
x_{37}	X					
x_{38}						X
x_{39}	X				X	
x_{40}	X			X		

2.1. Recherche du nombre minimum de compartiments

On a $I \leq 6$.

Pour chaque type de produits associons un sommet x_i et deux sommets x_i et x_j sont reliés si et seulement si x_i ne peut pas être dans le même compartiment que x_j.

On voit bien que la solution est optimale du fait qu'elle est obtenue avec $P^{\oplus}$, puisque le graphe est dans la classe des graphe parfaitement coloriable avec $P^{\oplus}$.

Après l'exécution de notre logiciel on a abouti à $\gamma(G_C) = 6$, ce qui nous donne le nombre minimum de compartiment nécessaire que doit possédé l'unité de stockage du centre. D'après notre programme, la répartition des différents types de produits dans les compartiments ce fait de la manière suivante :

Compartiment 1	**Compartiment 2**	**Compartiment 3**	**Compartiment 4**
x_3, x_4, x_6, x_{27}, x_{30}, x_{33}.	x_1, x_2, x_7, x_1, x_{21}, x_{24}, x_{31}, x_{38}.	x_5, x_9, x_{15}, x_{20}, x_{26}, x_{37}.	x_8, x_{11}, x_{16}, x_{23}, x_{34}, x_{35}.
Compartiment 5	**Compartiment 6**		
x_{10}, x_{14}, x_{19}, x_{28}, x_{39}.	x_{12}, x_{17}, x_{22}, x_{25}, x_{32}, x_{40}.		

Tandis que, les autres types de produits (x_{13}, x_{29}, x_{36}) peuvent se placer dans n'importe quel compartiment.

Nous présenterons ici les sous ensembles des produits ne pouvant pas être sur les même étagères de chacun des compartiments, niveaux de chacun des étagères ainsi leurs interactions interdites :

	Contre-indication Etagère	**Contre-indication Niveau**	**Contre-indication Côte à côte**
x_1	x_2, x_7, x_{12}, x_{24}.	x_{21}, x_{18}, x_{16}.	x_2, x_7, x_{11}, x_{38}.

x_2	$x_1, x_{18}, x_{21}, x_{31}, x_{38}$.	x_7, x_9.	x_1, x_7, x_{40}.
x_3	x_{33}, x_6, x_{20}.	x_1, x_6.	x_{16}, x_{25}.
x_4	x_{11}, x_{27}, x_{30}.	x_{24}, x_{27}.	x_{26}, x_{30}, x_{33}.
x_5	$x_9, x_{17}, x_{26}, x_{35}$.	x_{10}, x_{14}, x_{20}.	x_{26}, x_{28}.
x_6	$x_3, x_{12}, x_{15}, x_{27}$.	x_3, x_{33}.	x_{19}, x_{23}.
x_7	x_1, x_{24}.	x_2, x_{24}.	x_1, x_2.
x_8	x_{17}, x_{35}.	x_{26}, x_{38}.	x_{27}, x_{39}.
x_9	$x_5, x_{20}, x_{26}, x_{31}$.	x_2, x_{37}.	x_{20}, x_{31}, x_{35}.
x_{10}	$x_{14}, x_{19}, x_{21}, x_{25}$.	x_5, x_{40}.	x_{18}, x_{40}.
x_{11}	x_4, x_{16}, x_{34}.	x_{20}.	x_1, x_{22}, x_{32}.
x_{12}	x_1, x_6, x_{32}.	x_{17}, x_{22}, x_{25}.	x_{14}, x_{34}, x_{39}.
x_{13}			
x_{14}	x_{10}, x_{33}, x_{39}.	x_5, x_{19}.	x_{21}, x_{33}, x_{39}.
x_{15}	x_{16}, x_{35}.		x_{35}.
x_{16}	x_{11}, x_{23}.	x_1, x_{34}, x_{35}.	x_3, x_{25}, x_{38}.
x_{17}	x_5, x_8, x_{40}.	x_{12}, x_{25}.	$x_{19}, x_{21},, x_{28}$.
x_{18}	x_2, x_{28}.	x_1, x_{31}.	x_{10}, x_{27}.
x_{19}	x_{10}, x_{22}, x_{39}.	x_{14}.	x_6, x_{17}.
x_{20}	x_3, x_9, x_{37}.	x_5, x_{11}, x_{40}.	x_9, x_{37}.
x_{21}	$x_2, x_{10}, x_{25}, x_{38}$.	x_1, x_{31}.	x_{17}.
x_{22}	x_{19}, x_{40}.	x_{12}, x_{25}, x_{39}.	x_{11}, x_{32}.

x_{23}	x_{16}, x_{28}.	x_{32}.	x_6.
x_{24}	x_1, x_7, x_{30}.	x_4, x_7.	x_{30}.
x_{25}	x_{10}, x_{21}, x_{32}.	x_{12}, x_{17}, x_{22}.	x_3, x_{16}.
x_{26}	x_5, x_9, x_{37}.	x_8, x_{31}.	x_4, x_5.
x_{27}	x_4, x_6.	x_4, x_{30}.	x_8, x_{18}.
x_{28}	x_{18}, x_{23}.		x_5, x_{17}.
x_{29}			
x_{30}	x_4, x_{24}, x_{33}.	x_3, x_{27}.	x_4, x_{24}.
x_{31}	x_2, x_9.	x_{18}, x_{21}, x_{26}.	x_9.
x_{32}	x_{12}, x_{25}.	x_{23}, x_{39}.	x_{11}, x_{22}.
x_{33}	x_3, x_{14}, x_{30}.	x_6.	x_4, x_{14}.
x_{34}	x_{11}.	x_{16}, x_{35}.	x_{14}.
x_{35}	x_5, x_8, x_{15}.		x_9, x_{15}.
x_{36}			
x_{37}	x_{20}, x_{26}.	x_9.	x_{20}.
x_{38}	x_2, x_{21}.		x_1, x_{16}.
x_{39}	x_{14}, x_{19}.	x_{22}, x_{32}.	x_8, x_{12}, x_{14}.
x_{40}	x_{17}, x_{22}.	x_{10}, x_{20}.	x_2, x_{40}.

2.2. Recherche des nombres minimaux des étagères

On itère le même procédé que celui utilisé pour la recherche du nombre minimum de compartiment tout en considérant les sommets des G_{E_i} comme étant les types des

produits qui peuvent être dans le même compartiment i, et deux sommets seront reliés si et seulement si, ils ne peuvent pas être sur le même étagère.

Les solutions pour $G_{E_1}, G_{E_3}, G_{E_4}, G_{E_5}$ et G_{E_6} sont optimales du fait qu'elles sont obtenues lors de l'application de $P^{\oplus}$, ce qui nous donne :

$$\gamma\left(G_{E_1}\right)=2, \gamma\left(G_{E_3}\right)=3, \gamma\left(G_{E_4}\right)=2, \gamma\left(G_{E_5}\right)=2, et\ \gamma\left(G_{E_6}\right)=2.$$

Donc le nombre minimum d'étagères dans le compartiment 1 est 2 ;

le nombre minimum d'étagères dans le compartiment 3 est 3 ;

le nombre minimum d'étagères dans le compartiment 4 est 2 ;

le nombre minimum d'étagères dans le compartiment 5 est 2 ;

le nombre minimum d'étagères dans le compartiment 6 est 2.

Le graphe G_{E2} n'appartient pas à la classe G^* du fait que $P^{\oplus}$ viole une contrainte ce qui nous contraint d'utiliser la méthode tabou et ACO pour rendre nul le nombre de contraintes violées.

Après ces trois exécutions on a abouti à $\gamma\left(G_{E_{C1}}\right)=3$, qui représente le nombre minimum d'étagères nécessaires, dans le deuxième compartiment, que doit posséder l'unité de stockage du centre.

Nous présentons ici les dispositions possibles des produits, de différents types, dans les différentes étagères de chacun des compartiments.

Compartiment 1		
	Etagère 1	Etagère 2
Produits	x_3, x_{27}, x_{30}.	x_4, x_6, x_{33}.

Compartiment 2			
	Etagère 1	Etagère 2	Etagère 3
Produits	x_2, x_{24}.	$x_1, x_{18}, x_{21}, x_{31}$.	x_7, x_{38}.

Compartiment 3			
	Etagère 1	Etagère 2	Etagère 3
Produits	$X_{9,}$ x_{37}.	$x_{5,}$ x_{20}.	x_{26}.

Compartiment 4		
	Etagère 1	Etagère 2
Produits	$x_{8,}$ $x_{11,}$ x_{23}.	$x_{16,}$ $x_{34,}$ x_{35}.

Compartiment 5		
	Etagère 1	Etagère 2
Produits	$x_{10,}$ x_{39}.	$x_{14,}$ x_{19}.

Compartiment 6		
	Etagère 1	Etagère 2
Produits	$x_{32,}$ x_{40}.	$x_{12,}$ $x_{17,}$ $x_{22,}$ x_{25}.

2.3. Recherche des nombres minimaux des niveaux dans les étagères de chaque compartiment

En affectant à chaque type un sommet et en considérant les sous graphes définis par les sous ensembles de types de produits, d'un même étagère de sorte que deux sommets sont adjacents si et seulement si, les deux types leurs correspondant ne peuvent pas être dans le même niveau.

En répétant les étapes précédentes, on a abouti à :

$$\gamma\left(G_{N_{11}}\right)=2,\gamma\left(G_{N_{12}}\right)=2,\gamma\left(G_{N_{21}}\right)=1,\gamma\left(G_{N_{22}}\right)=2,\gamma\left(G_{N_{23}}\right)=1,\gamma\left(G_{N_{31}}\right)=2,\gamma\left(G_{N_{32}}\right)=2,$$
$$\gamma\left(G_{N_{33}}\right)=1,\gamma\left(G_{N_{41}}\right)=2,\gamma\left(G_{N_{42}}\right)=3,\gamma\left(G_{N_{51}}\right)=1,\gamma\left(G_{N_{52}}\right)=2,\gamma\left(G_{N_{61}}\right)=1,\gamma\left(G_{N_{62}}\right)=3.$$

La répartition des différents types de produits dans les étagères de chacun des compartiments est faite de la manière suivante :

Compartiment 1			
Etagère 1		**Etagère 2**	
Niveau 1	**Niveau 2**	**Niveau 1**	**Niveau 2**
$x_{3,}$ x_{27}.	x_{30}.	$x_{6,}$ x_{4}.	x_{33}.

Compartiment 2			
Etagère 1	**Etagère 2**		**Etagère 3**
Niveau 1	**Niveau 1**	**Niveau 2**	**Niveau 1**
$X_2.x_{24}$.	x_1, x_{31}.	x_{18}, x_{21}.	x_7, x_{38}.

Compartiment 3				
Etagère 1		**Etagère 2**		**Etagère 3**
Niveau 1	**Niveau 2**	**Niveau 1**	**Niveau 2**	**Niveau 1**
x_9.	x_{37}	x_5.	X_{20}	x_{26}.

Compartiment 4			
Etagère 1	**Etagère 2**		
Niveau 1	**Niveau 1**	**Niveau 2**	**Niveau3**
x_8, x_{11}, x_{23}.	x_{16}.	x_{34}.	x_{35}.

Compartiment 5		
Etagère 1	**Etagère 2**	
Niveau 1	**Niveau 1**	**Niveau 2**
x_{10}, x_{39}.	x_{14}.	x_{19}.

Compartiment 6			
Etagère 1	**Etagère 2**		
Niveau 1	**Niveau 1**	**Niveau 2**	**Niveau 3**
x_{32}, x_{40}.	x_{12}.	x_{22}, x_{17}.	x_{25}.

2.4. Recherche de l'emplacement des produits

Il reste à définir la position des différents types de produits sur chaque niveau k des étagères j dans chaque compartiment i.

Pour ceci, considérons les graphes engendrés par les sommets de même niveau et deux sommets cette fois ci sont reliés s'ils ne peuvent pas être côte à côte. Ceci nous a permis de définir le nombre minimum de compartiments, d'étagères ainsi que de niveaux et les positions des produits.

Après classement de ces derniers on peut adopté le système de référence des produits comme déjà cités, et on propose le tableau de référence suivant :

Produits	**Référence**	**Signification**
x_1	P_{2211}	Le type x_1 est dans le compartiment 2, étagère 2, niveau 1 à gauche.
x_2	P_{2111}	Le type x_2 est dans le compartiment 2, étagère 1, niveau 1 à gauche.
x_3	P_{1111}	Le type x_3 est dans le compartiment 1, étagère 1, niveau 1 à gauche.
x_4	P_{1211}	Le type x_4 est dans le compartiment 1, étagère 2, niveau 1 à gauche.
x_5	P_{321}	Le type x_5 est dans le compartiment 3, étagère 2, niveau 1.

x_6	P_{1212}	Le type x_6 est dans le compartiment 1, étagère 2, niveau 1 à droite.
x_7	P_{2311}	Le type x_7 est dans le compartiment 2, étagère 3, niveau 1 à gauche.
x_8	P_{4111}	Le type x_8 est dans le compartiment 4, étagère 1, niveau 1 à gauche.
x_9	P_{311}	Le type x_9 est dans le compartiment 3, étagère 1, niveau 1.
x_{10}	P_{5111}	Le type x_{10} est dans le compartiment 5, étagère 1, niveau 1 à gauche.
x_{11}	P_{4113}	Le type x_{11} est dans le compartiment 4, étagère 1, niveau 1 au centre.
x_{12}	P_{621}	Le type x_{12} est dans le compartiment 6, étagère 2, niveau 1.
x_{13}	P_{ijk}	Le type x_{13} est dans le compartiment i, étagère j, niveau k.
x_{14}	P_{521}	Le type x_{14} est dans le compartiment 5, étagère 2, niveau 1.
x_{15}	$P_{ijk} / \ i \neq 4$	Le type x_{15} est dans le compartiment i, étagère j, niveau k.
x_{16}	P_{421}	Le type x_{16} est dans le compartiment 4, étagère 2, niveau 1.
x_{17}	P_{6222}	Le type x_{17} est dans le compartiment 6, étagère 2, niveau 2 à droite.
x_{18}	P_{2221}	Le type x_{18} est dans le compartiment 2, étagère 2, niveau 2 à gauche.
x_{19}	P_{522}	Le type x_{19} est dans le compartiment 5, étagère 2, niveau 2.
x_{20}	P_{322}	Le type x_{20} est dans le compartiment 3, étagère 2, niveau 2.
x_{21}	P_{2222}	Le type x_{21} est dans le compartiment 2, étagère 2, niveau 2

		à droite.
x_{22}	P_{6221}	Le type x_{22} est dans le compartiment 6, étagère 2, niveau 2 à gauche.
x_{23}	P_{4112}	Le type x_{23} est dans le compartiment 4, étagère 1, niveau 1 à droite.
x_{24}	P_{2112}	Le type x_{24} est dans le compartiment 2, étagère 1, niveau 1 à droite.
x_{25}	P_{623}	Le type x_{25} est dans le compartiment 6, étagère 2, niveau 3.
x_{26}	P_{331}	Le type x_{26} est dans le compartiment 3, étagère 3, niveau 1.
x_{27}	P_{1112}	Le type x_{27} est dans le compartiment 1, étagère 1, niveau 1 à droite.
x_{28}	$P_{ijk} / i \neq 2,4$	Le type x_{28} est dans le compartiment i, étagère j, niveau k.
x_{29}	P_{ijk}	Le type x_{29} est dans le compartiment i, étagère j, niveau k.
x_{30}	P_{112}	Le type x_{30} est dans le compartiment 1, étagère 1, niveau 2.
x_{31}	P_{2212}	Le type x_{31} est dans le compartiment 2, étagère 2, niveau 1 à droite.
x_{32}	P_{6111}	Le type x_{32} est dans le compartiment 6, étagère 1, niveau 1 à gauche.
x_{33}	P_{122}	Le type x_{33} est dans le compartiment 1, étagère 2, niveau 2.
x_{34}	P_{422}	Le type x_{34} est dans le compartiment 4, étagère 2, niveau 2.
x_{35}	P_{423}	Le type x_{35} est dans le compartiment 4, étagère 2, niveau 3.
x_{36}	P_{ijk}	Le type x_{36} est dans le compartiment i, étagère j, niveau k.
x_{37}	P_{312}	Le type x_{37} est dans le compartiment 3, étagère 1, niveau 2.

x_{38}	P_{2312}	Le type x_{38} est dans le compartiment 2, étagère 3, niveau 1 à droite.
x_{39}	P_{5112}	Le type x_{39} est dans le compartiment 5, étagère 1, niveau 1 à droie.
x_{40}	P_{6112}	Le type x_{40} est dans le compartiment 6, étagère 1, niveau 1 à droite.

Conclusion générale

Conclusion

Arrivé au terme de notre étude, il y a lieu de faire un bilan du travail fait.

De nombreux algorithmes et méthodes de résolution exactes ont été développés suite aux travaux de recherche entrepris sur la coloration des sommets d'un graphe. Ces algorithmes concernent certaines classes complètement caractérisées pour lesquelles la solution optimale est totalement déterminée par l'application de la méthode adéquate. La NP- Complétude avérée de certains problèmes modélisables par l'approche de la coloration, fait que le quasi-unique moyen d'obtenir une solution est le recours à des méthodes approchées (heuristiques). Dans notre mémoire, nous avons présenté une contribution à la résolution d'une large classe des problèmes d'affectations sous contraintes. Cette contribution consiste en l'implémentation d'une nouvelle procédure, nommée $P^{\oplus}$ et de son hybridation avec des métaheuristiques en l'occurrence la méthode tabou et la méthode basée sur le principe de colonies de fourmis. L'outil obtenu s'est avéré efficace dans la majeure partie des instances testées.

Les problèmes d'affectations sous contraintes s'adaptent à une grande partie des problèmes d'optimisation combinatoire, rencontrés en industrie, dans l'ordonnancement et autres applications pratiques et/ou opérationnelles, pour ne citer que les problèmes ; d'emploi du temps, Allocation des ressources, Placement et Localisation …etc. Cependant, nous avons proposé dans notre mémoire d'apporter des éléments de résolution aux ces problèmes à travers la coloration des sommets d'un graphe quelconque. Ce qui nous a conduit à proposer une résolution heuristique du Problème de la Coloration des Sommets d'un Graphe quelconque (P.C.S.G) et par là les problèmes d'affectation sous contraintes.

Le problème de la coloration des sommets d'un graphe quelconque est un problème connu dans le milieu de l'optimisation combinatoire comme étant un problème NP-Complet, la solution exacte de celui-ci est restreinte aux classes des graphes parfaits dont nous avons ajouter une nouvelle classe de celles-ci à celles déjà existantes qui sont les graphes de la classe G^* et qui sont coloriable de manière polynomiale par

notre procédure $P^{\oplus}$. Cette dernière s'est avérée efficace pour un certain nombre de graphes appartenant ou non à notre classe G^{*}.

La deuxième partie de notre travail a consisté à vérifier l'inclusion stricte de certain classes connues telles que classes : biparti et de Berge ; multi parti, sans griffe et sans patte.

Les graphes, pour lesquels $P^{\oplus}$ ne produit pas la solution réalisable, nous avons développé un algorithme intégrant $P^{\oplus}$ à une procédure d'adaptation du principe ACO et RT pour l'amélioration de la solution courante.

La dernière partie de notre travail a été consacrée à l'étude d'un exemple pratique envisageable des problèmes d'affectation sous contraintes consistant en l'affectation de plusieurs produits chimiques dans une banque de stockage d'un laboratoire de recherche. Les contraintes d'une telle affectation sont entres autres :

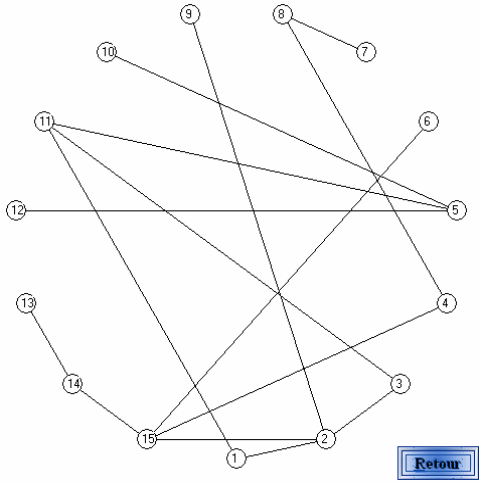
- Eviter les dangers qui peuvent survenir suite à des interactions entres les différents produits ;
- Respecter les conditions de stockage des produits pour éviter le pourrissement des produits :
  - Abri de lumière ;
  - Abri de chaleur ;
  - Basse température.

Tout en minimisant le nombre de compartiments dans la banque de stockage, d'étagères dans chaque compartiment et le nombre de niveaux de chaque étagère.

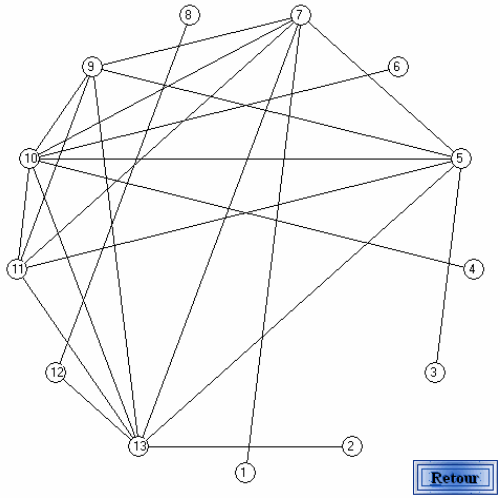
La simulation de notre méthode a donné lieu à un logiciel que nous avons nommé « Coloration » et qui a été programmer en langage de programmation Delphi 5 sous l'environnement Windows orienté objet. Notre logiciel permet de colorier un graphe G quelconque de manière quasi-optimale.

Annexe

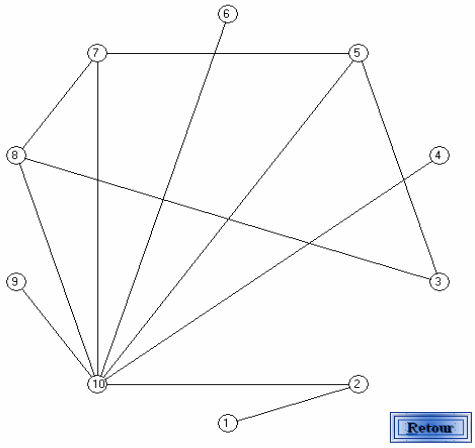
1. Instances d'application de notre méthode



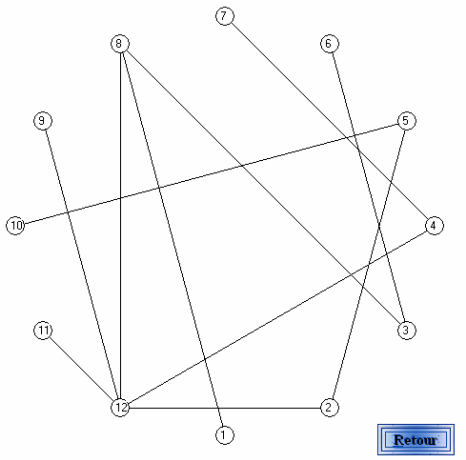
Le nombre de contraintes violées est 0 et le nombre chromatique est 2 autrement dit, la solution trouvée est optimale.



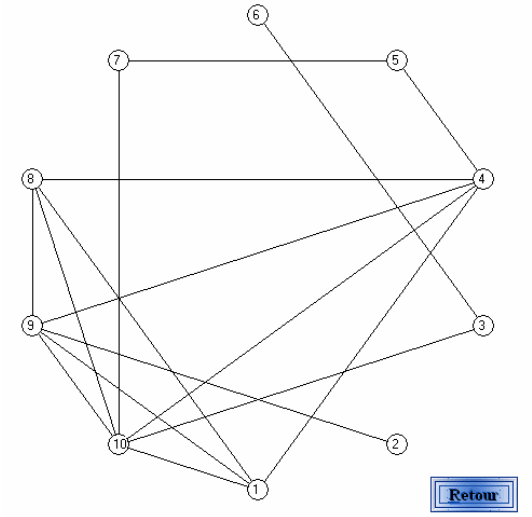
Le nombre de contraintes violées est 0 et le nombre chromatique est 4 autrement dit, la solution trouvée est optimale.



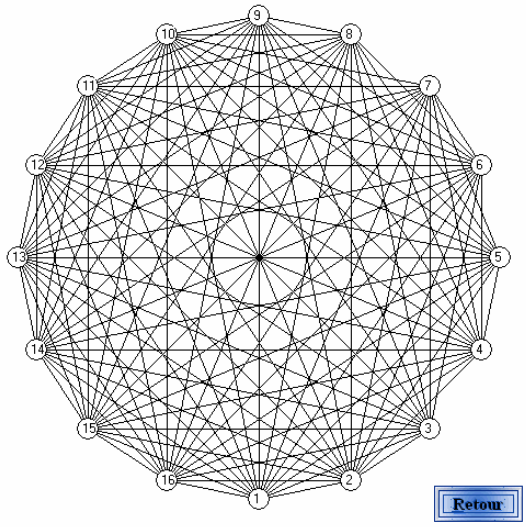
Le nombre de contraintes violées est 0 et le nombre chromatique est 3 autrement dit, la solution trouvée est optimale.



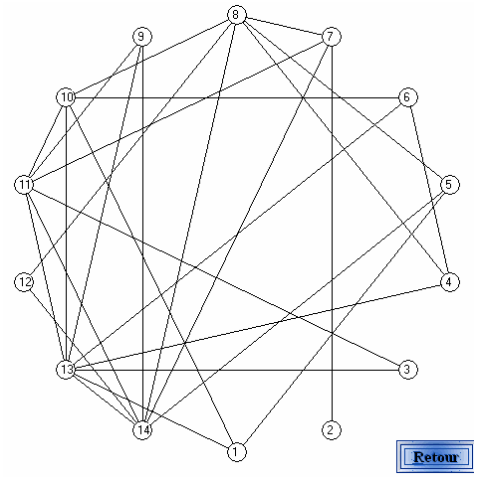
Le nombre de contraintes violées est 0 et le nombre chromatique est 2 autrement dit, la solution trouvée est optimale.



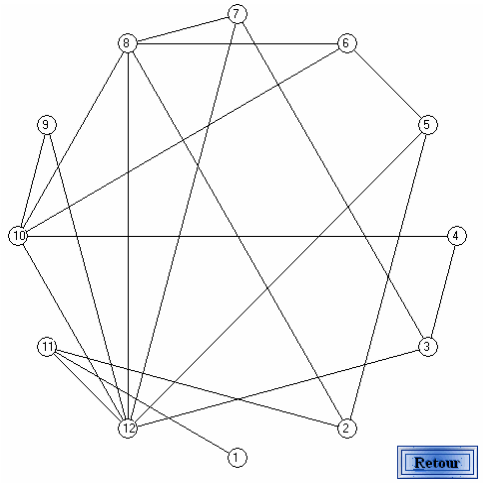
Le nombre de contraintes violées est 0 et le nombre chromatique est 5 autrement dit, la solution trouvée est optimale.



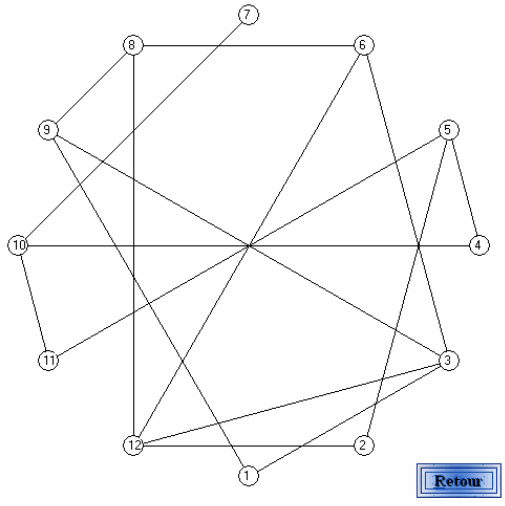
Le nombre de contraintes violées est 0 et le nombre chromatique est 16 autrement dit, la solution trouvée est optimale.



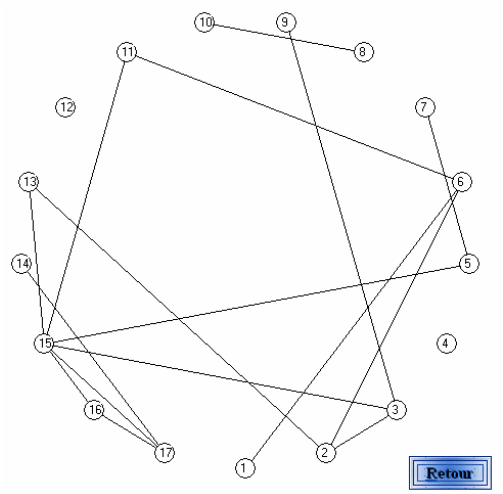
Le nombre de contraintes violées est 0 et le nombre chromatique est 4 autrement dit, la solution trouvée est optimale.



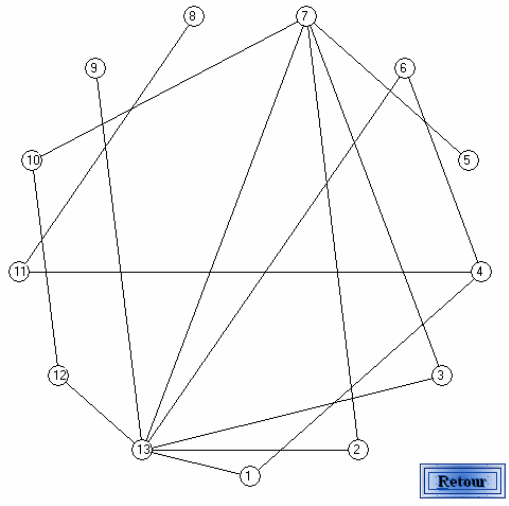
Le nombre de contraintes violées est 0 et le nombre chromatique est 3 autrement dit, la solution trouvée est optimale.



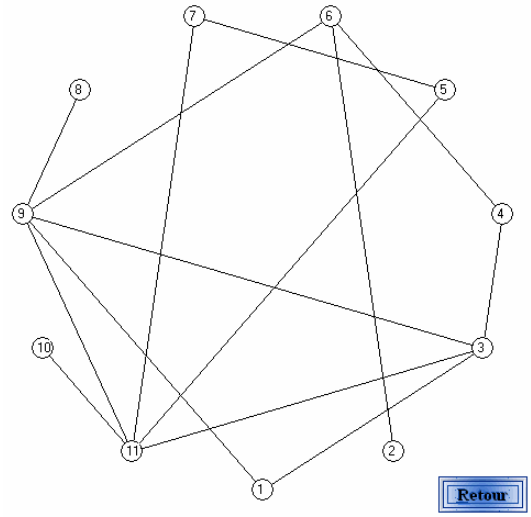
Le nombre de contraintes violées est 0 et le nombre chromatique est 3 autrement dit, la solution trouvée est optimale.



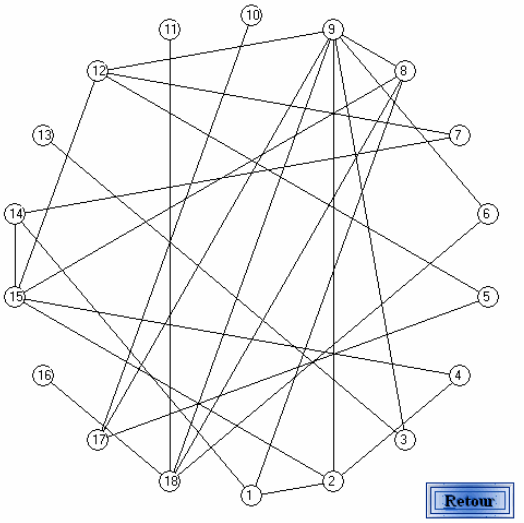
Lors de l'exécution de notre logiciel on a trouvé 4 composantes connexes et le nombre chromatique donné par notre procédure est 3 avec violation d'une seule contrainte.



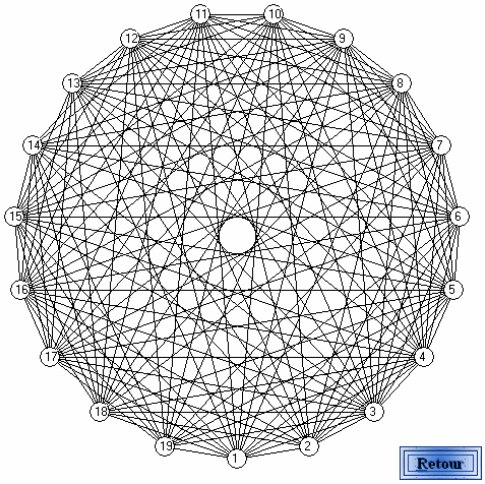
Le nombre de contraintes violées est 0 et le nombre chromatique est 4 autrement dit, la solution trouvée est optimale.



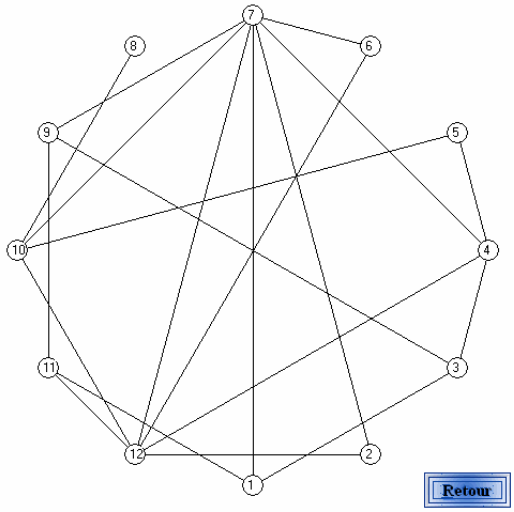
Le nombre de contraintes violées est 0 et le nombre chromatique est 3 autrement dit, la solution trouvée est optimale.



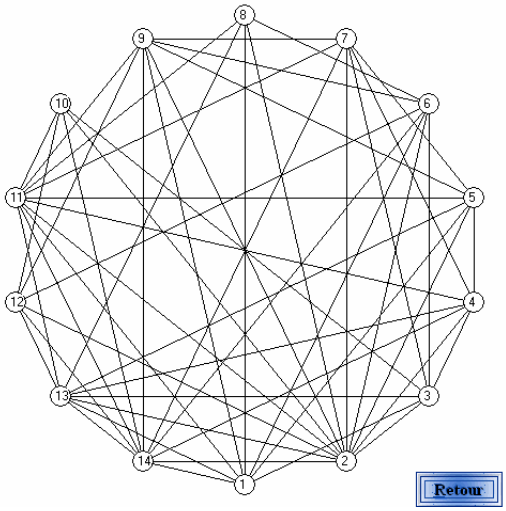
Le nombre de contraintes violées est 0 et le nombre chromatique est 3 autrement dit, la solution trouvée est optimale.



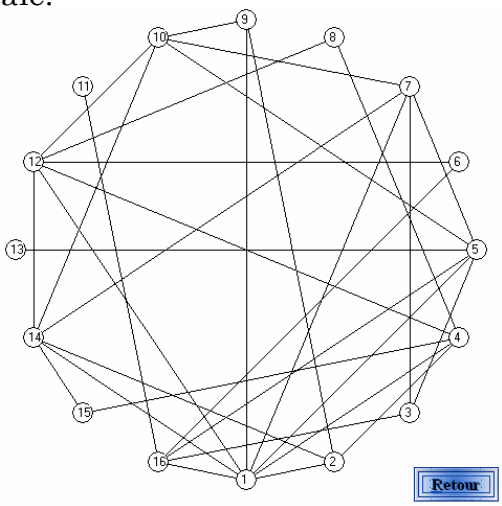
Le nombre de contraintes violées est 0 et le nombre chromatique est 16 autrement dit, la solution trouvée est optimale.



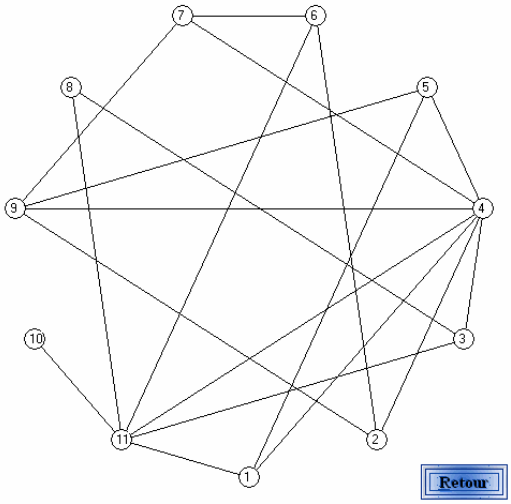
Le nombre de contraintes violées est 0 et le nombre chromatique est 3 autrement dit, la solution trouvée est optimale.



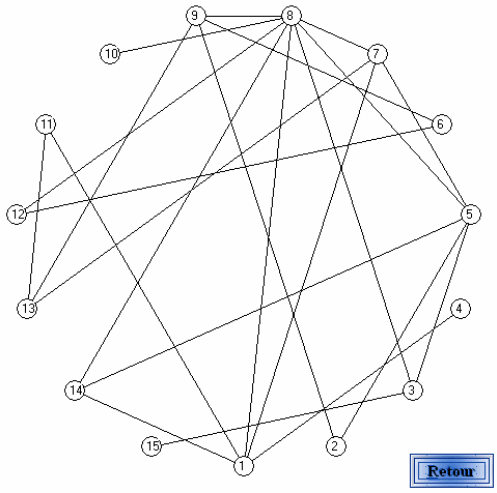
Le nombre de contraintes violées est 0 et le nombre chromatique est 7 autrement dit, la solution trouvée est optimale.



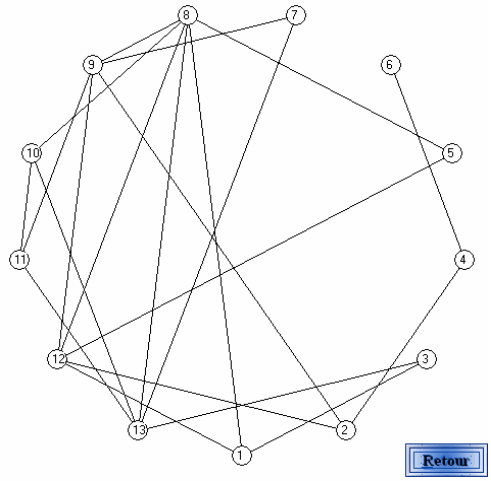
Le nombre de contraintes violées est 0 et le nombre chromatique est 3 autrement dit, la solution trouvée est optimale.



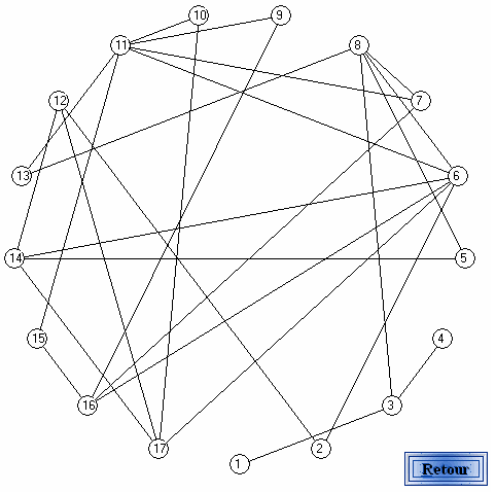
Le nombre de contraintes violées est 0 et le nombre chromatique est 4 autrement dit, la solution trouvée est optimale.



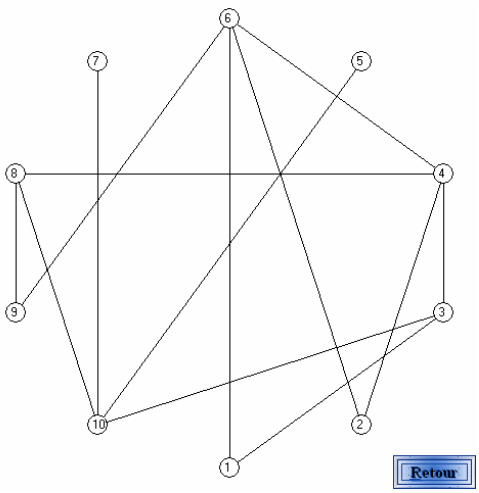
Lors de l'exécution de notre logiciel on a trouvé que ce graphe est 4 coloriable et que le nombre de contraintes violées par notre procédure est 1, du fait que les deux sommets x_{13} et x_{11} sont reliés avec le sommet star qui est ici le sommet x_8 par une chaîne de longueur paire tel que (x_{11}, x_{13}) est une arête dans E, auquel cas nous utilisons notre méthode hybride qui nous a permis d'annuler le nombre de contraintes violées et d'avoir un graphe 4 coloriable.



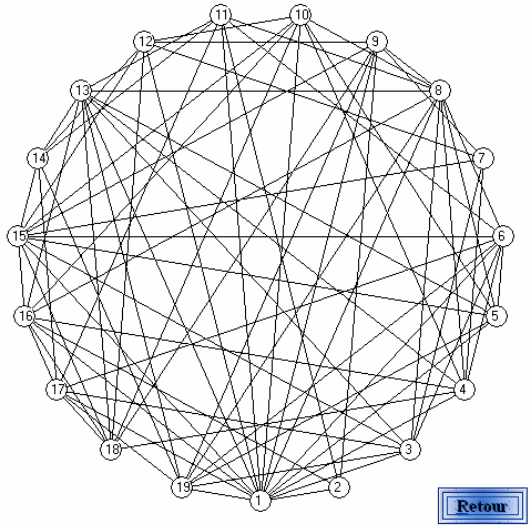
Le nombre de contraintes violées est 0 et le nombre chromatique est 3 autrement dit, la solution trouvée est optimale.



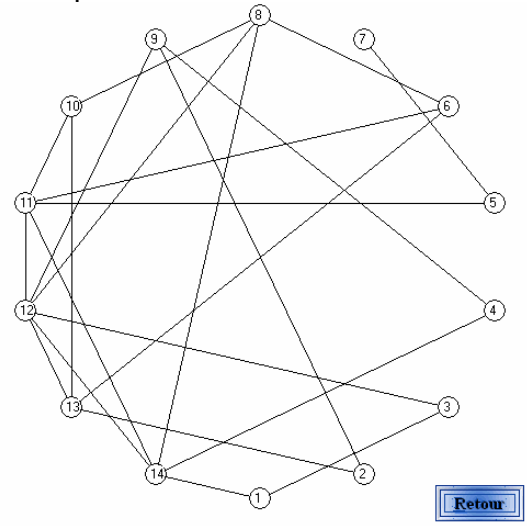
Le nombre de contraintes violées est 0 et le nombre chromatique est 3 autrement dit, la solution trouvée est optimale.



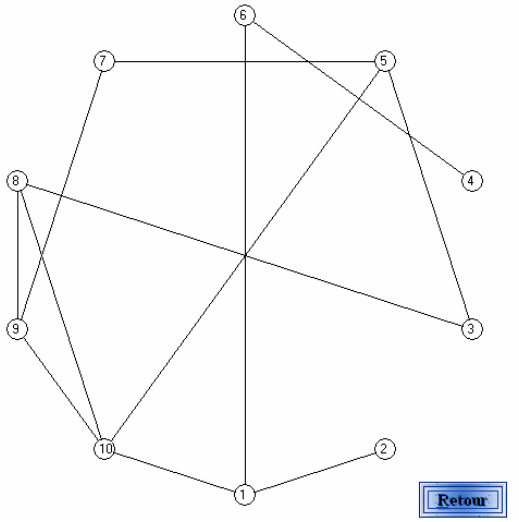
Le nombre de contraintes violées est 0 et le nombre chromatique est 3 autrement dit, la solution trouvée est optimale.



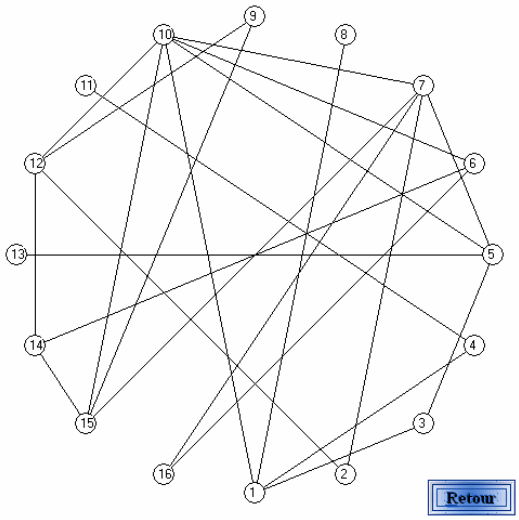
Le nombre de contraintes violées est 0 et le nombre chromatique est 6 autrement dit, la solution trouvée est optimale.



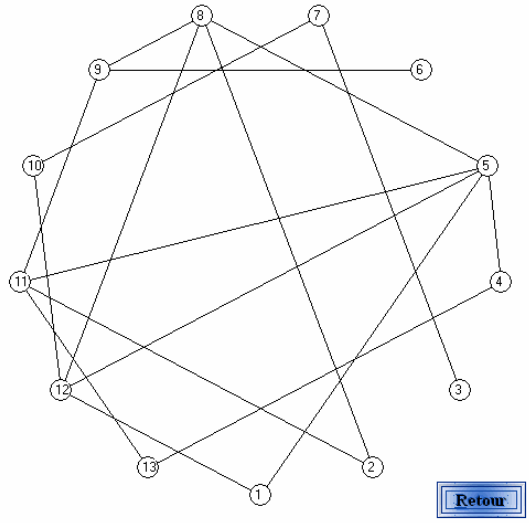
Le nombre de contraintes violées est 0 et le nombre chromatique est 3 autrement dit, la solution trouvée est optimale.



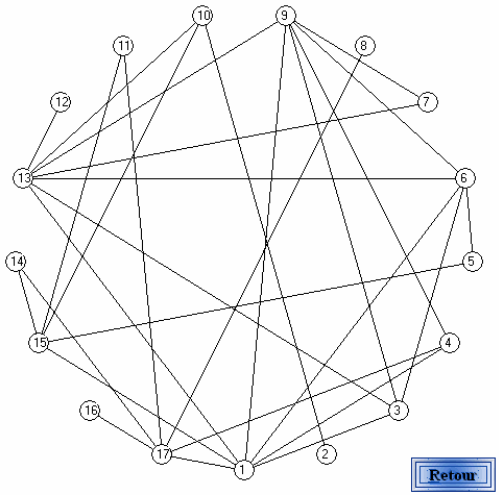
Le nombre de contraintes violées est 0 et le nombre chromatique est 3 autrement dit, la solution trouvée est optimale.



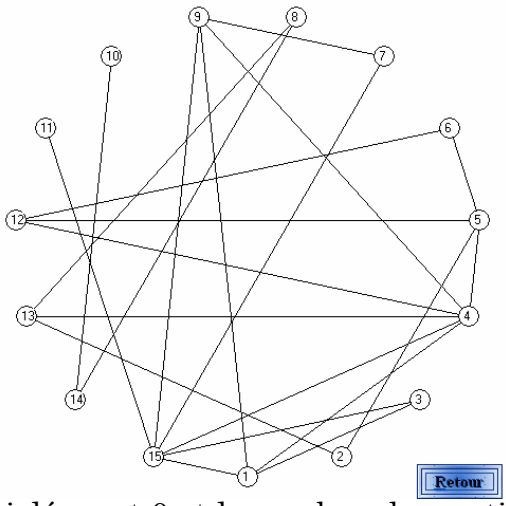
Le nombre de contraintes violées est 0 et le nombre chromatique est 3 autrement dit, la solution trouvée est optimale.



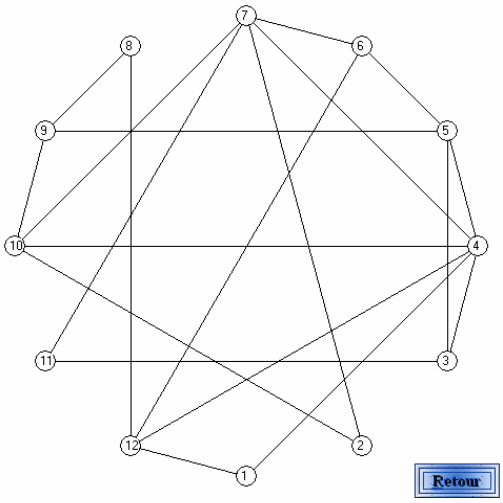
Le nombre de contraintes violées est 0 et le nombre chromatique est 3 autrement dit, la solution trouvée est optimale.



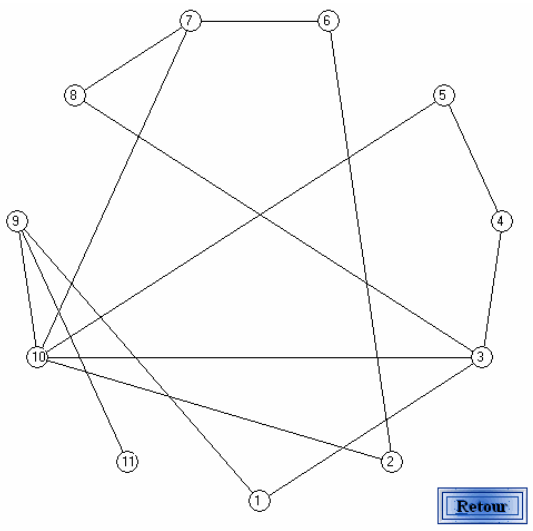
Le nombre de contraintes violées est 0 et le nombre chromatique est 5 autrement dit, la solution trouvée est optimale.



Le nombre de contraintes violées est 0 et le nombre chromatique est 4 autrement dit, la solution trouvée est optimale.



Lors de l'exécution de notre logiciel on a trouvé que ce graphe est 3 coloriable et que le nombre de contraintes violées par notre procédure est 1, du fait que les deux sommets x_8 et x_9 sont reliés avec le sommet star qui est ici le sommet x_4 par une chaîne de longueur paire tel que $(x_8, x_9) \in E$. Auquel cas nous utilisons notre méthode hybride, tout en suivant les organigrammes donnés précédemment, qui nous a permis d'annuler le nombre de contraintes violées et d'avoir un graphe 4 coloriable.



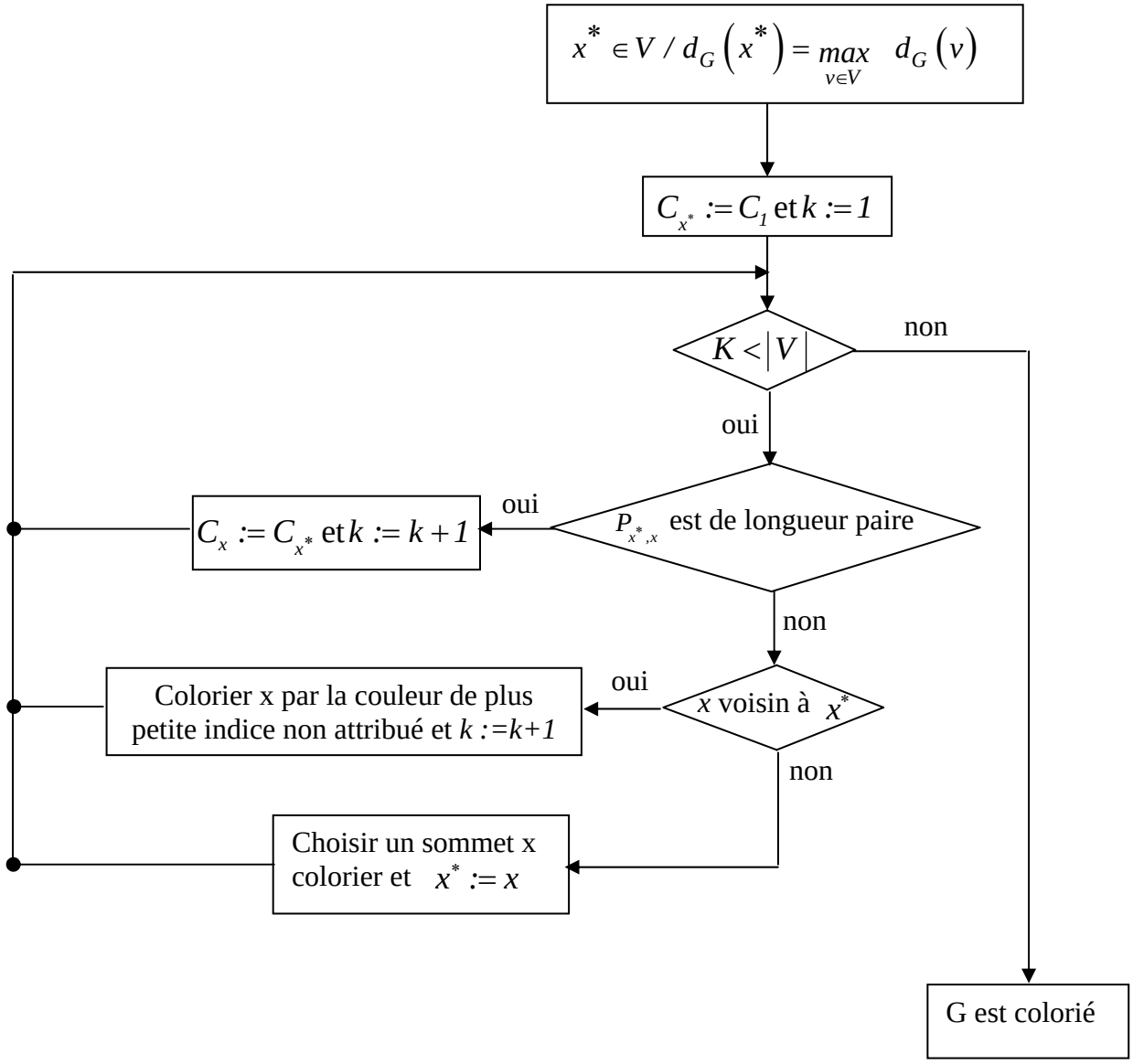
Le nombre de contraintes violées est 0 et le nombre chromatique est 2 autrement dit, la solution trouvée est optimale.

Nous présenterons dans ce qui suit certaines instances de graphes pour lesquelles la taille est un peu grande.

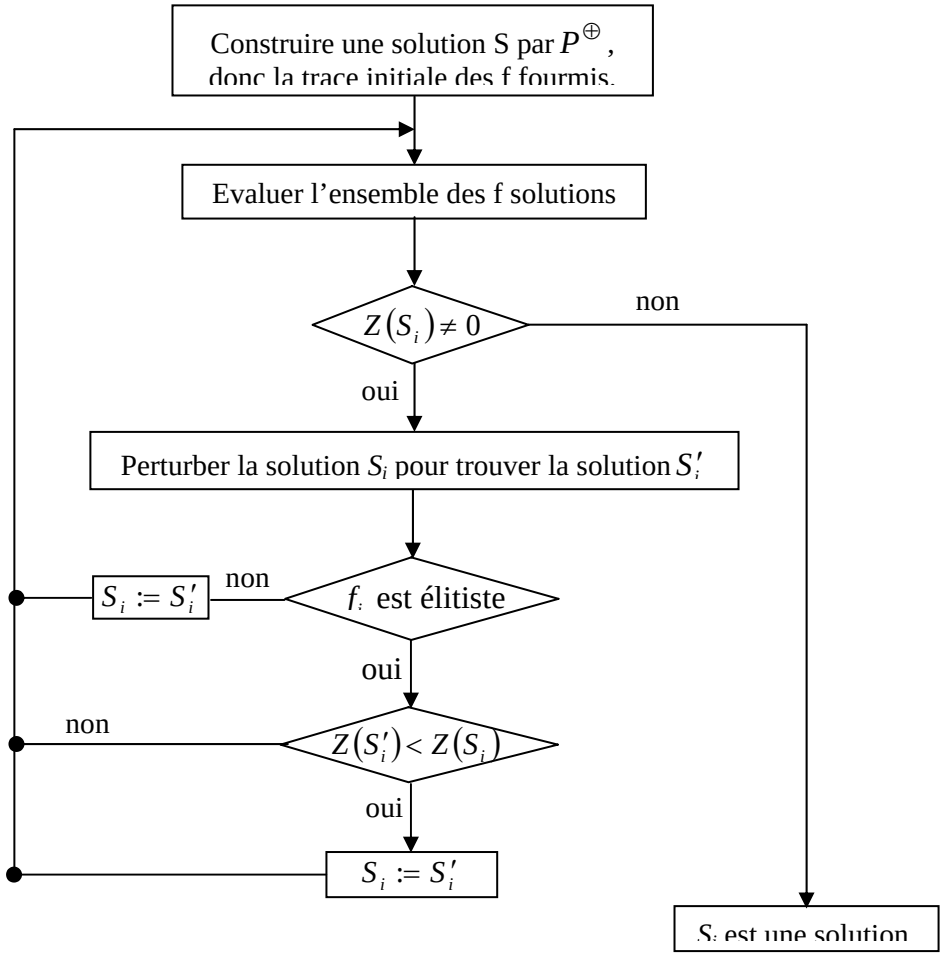
(Sommets, arêtes)	Solution donnée par P^+		Solution améliorée par la méthode TR	Solution améliorée par la méthode ACO
	Nombre de couleurs utilisées	Nombre de contraintes violées		
(300, 22408)	115	0	Rien à signaler	Rien à signaler
(100, 2439)	42	2	Non améliorée	Non améliorée
(400, 39842)	186	0	Rien à signaler	Rien à signaler
(200, 9963)	63	0	Rien à signaler	Rien à signaler
(359, 32018)	152	0	Rien à signaler	Rien à signaler
(426, 45185)	197	1	198	198
(500, 62167)	173	0	Rien à signaler	Rien à signaler

(550, 75572)	213	0	Rien à signaler	Rien à signaler
(433, 46806)	201	0	Rien à signaler	Rien à signaler
(114, 3273)	47	0	Rien à signaler	Rien à signaler
(280, 19523)	105	0	Rien à signaler	Rien à signaler
(379, 35639)	169	0	Rien à signaler	Rien à signaler
(463, 53600)	146	3	Non améliorée	Non améliorée
(1000, 250124)	433	0	Rien à signaler	Rien à signaler

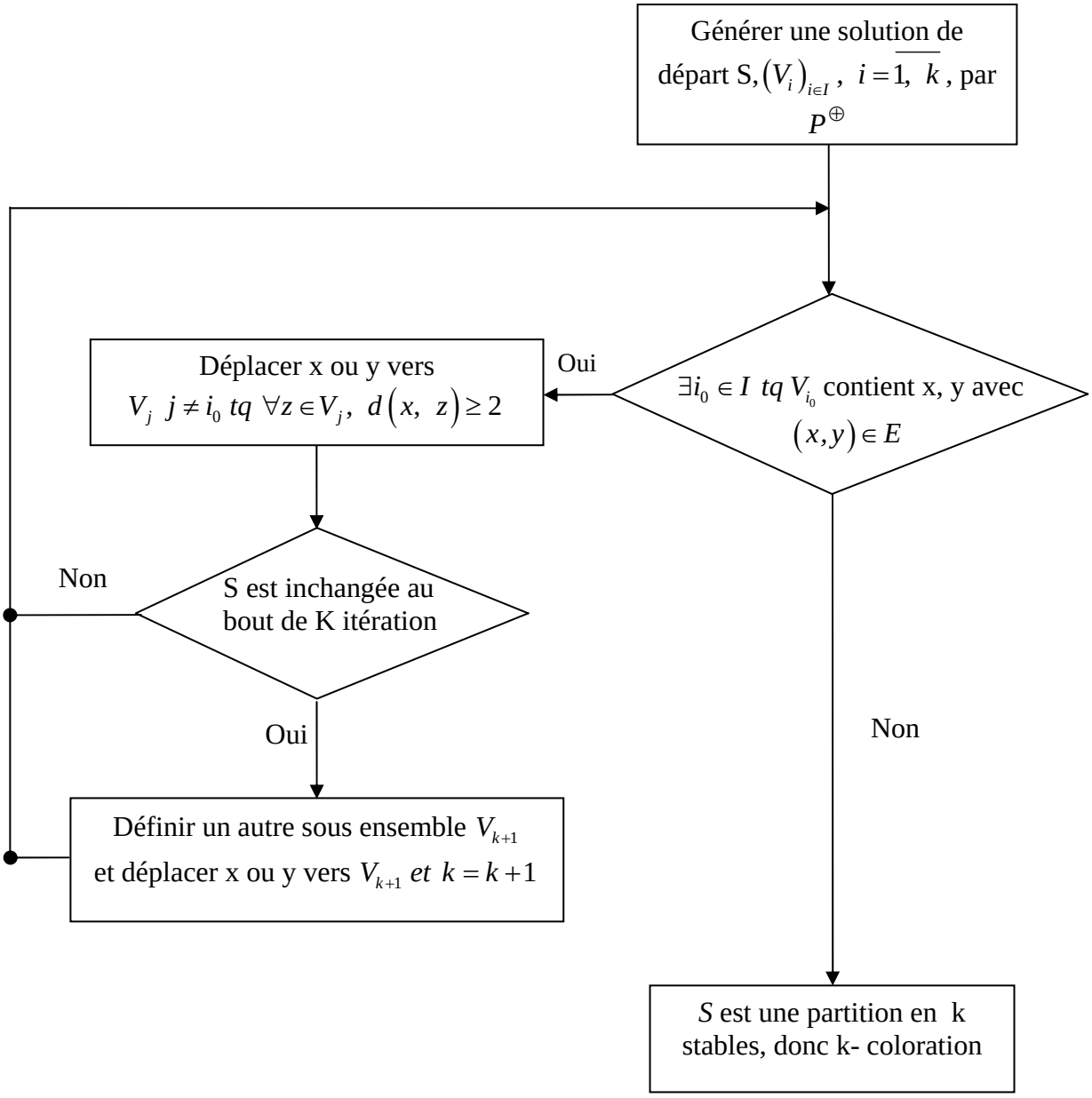
2. Organigramme de notre méthode $P^{\oplus}$



3. Organigramme de l'adaptation du principe de la méthode ACO



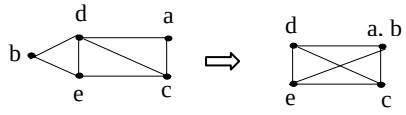
4. Organigramme de l'adaptation de la méthode TR

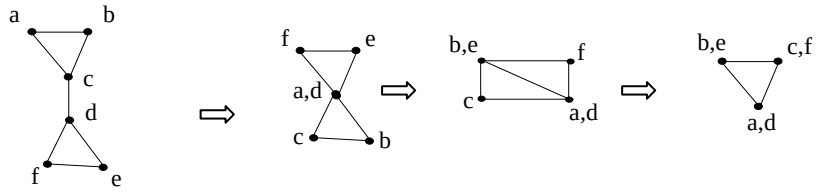


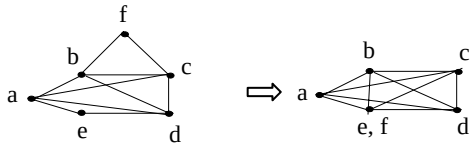
5. Comparaison entre la coloration par $P^{\oplus}$ et la coloration par contraction

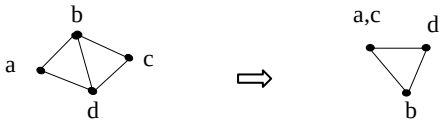
Instances / Coloration	instance 1	Instance 2	Instance 3	Instance 4
Contraction	G non colorié	G est colorié	G non colorié	G est colorié
Par $P^{\oplus}$	G est colorié	G non colorié	G est colorié	G est colorié

Contraction des graphes des instances précédentes









Bibiographie

[1] H. AIT HADDADENE, *Sur quelques structures de graphes parfaits,* Thèse d'Etat, USTHB 2000.

[2] J. AYAS et M.A. VIAU, *la recherche tabou,* 2004.

[3] V. BACHELET, Métah*euristiques parallèles hybrides : Application au QAP*. *PhD,* these, USTL LIFL France, 1999.

[4] C. BERGE, *Graphes et hypergraphes*, Editions Gauthier - villars, Bordas, Paris198.

[5] V. BARICHARD, *Approches hybrides pour les problèmes multiobjectifs*, Thèse de Doctorat, Ecole Doctorale d'Angers, 2003.

[6] CERT-ONERA, *Equipe d'optimisation combinatoire*, Département d'Etudes et de Recherches en Informatique.

[7] M. CHUDNOVSKY, N. ROBERTSON, P.D. SEYMOUR and R. THOMAS, *Progress on perfect graphs*, Mathematics Programming.

[8] A. COLORNI, M. DORIGO, V. MANEZZIO, and M. TRUBIAN, *Ant system for job-shop scheduling.* Belgian Journal of Operations Research, Statistics and Computer Science, 34, 1994.

[9] J.L. DENEUBOURG, J.M. PASTEELS, and J.C. VERHAEGHE, *Probabilistic behaviour in ants : a strategy of errors ? Journal of Theoretical Biology*, 105 :259–271, 1983.

[10] J.L DENEUBOURG and S. GOSS, *Collective patterns and decision-making. Ethology and Evolution,* pages 295–311, 1989.

[11] M. DORIGO AND L. M. GAMBARDELLA, *Ant colony system: A cooperative learning approach to the traveling salesman problem*, IEEE Transactions on Evolutionary Computation, 1(1), 1997.

[12] M. DORIGO, V. MANIEZZO, and A. COLORNI, *Positive feedback as a search strategy.* Technical report, Technical Report 91016, Dipartimento di Elettronica e Informatica, P olitecnico di Milano, italie, 1991.

[13] M. DORIGO, V. MANIEZZO, and A. COLORNI, *Optimization by a colony of cooperating agents.* IEEE Transactions on Systems, Man, and Cybernetics, Part B : Cybernetics, 26(1) :29–41, 1996.

[14] M. DORIGO, *Learning and Natural Algorithms* (in Italian). PhD thesis, DEI, Politecnico di Milano,Italy, 1992.

[15] C. DUHAMEL, *Un cadre formel pour les méthodes par amélioration itérative, Application deux problèmes d'optimisation dans les réseaux*, Docteur d'Université, Spécialité Informatique, Université Blaise Pascal - Clermont-Ferrand II 2001.

[16] ENCYCLOPÉDIE DES MATHEMATIQUES. Bordas, Paris, 1973.

[17] B. FABIAN, et S. BERNARD, *Coloration des sommets d'un graphe*, TP Fortran 1ere licence 2001-2002.

[18] J.GABORIAU, *Heuristiques pour le calcul de modèles stables optimisation par colonies de fourmis et recherche locale*, DEA Informatique co-habileté entre Université de NANTES et Université d'ANGERS, Septembre 2003.

[19] M. R. GAREY, D. S. JOHNSON, *Computers and intractability, a guide to the theory of* NP-*Completeness*, W.H. Freeman and Company, 1979.

[20] P. GASTIN, *Complexité*, LIAFA, Université Paris 7, UMR CNRS 7089.

[21] F. GLOVER and M. LAGUNA, *Kluwer Academic Publishers*, 1997. [ving et un]

[22] F. GLOVER, *Tabu search : part I*, ORSA Journal on Computing, Vol. 1 (3), 1989, p. 190-206.

[23] J. HAO, P. GALINIER et M. HABIB, *Métaheuristique pour l'optimisation combinatoire et l'affectation sous contraintes*.

[24] J. HOLLAND, *Adaptation in natural and artificial systems*, University of Michigan Press, Ann Arbor, 1975.

[25] S. KIRKPATRICK, C. D. GELATT, and M. P. VECCHI, *Optimization by simulated annealing*. Science, 220, 1983.

[26] P.LACOMME, *Optimisation par colonies de fourmis pour les problèmes sur arcs*, 4ème conférence francophone de modélisation et simulation, MOSIM.03- du 23 au 25 avril 2003- Toulouse, France.

[27] P. NICOLAS, F. SAUBION et I. STEPHAN, *Optimisation par colonies de fourmis pour la programmation logique étendue*, Université d'Angers.

[28] M. PALPANT, *Conception d'une métaheuristique et application au problème d'ordonnancement de projet à moyens limites*, sous la direction de Christan Artigues et Philippe Michelon, Laboratoire d'informatique d'Avigon, 15 juin 2001.

[29] M PREISSMANN, *graphes et optimisation combinatoire*, Séminaire, Mathématiques discrètes et applications - Modèles de calcul.

[30] I. RUSU, *Graphes parfaits : étude structurelle et algorithmiques de coloration,*

Thèse D'état, Université de Paris-Sud, 1994.

[31] Rencontres avec la recherche opérationnelle, Journal du club RO, 2005.

[32] B. SADI, *Théorie des graphes, complexité algorithmique*, cours de Mathématique, Institut d'Informatique de l'Université de Tizi-Ouzou, 1993.

[33] M. SAKAROVITCH, *Techniques mathématiques de la recherche opérationnelle, Optimisation combinatoire*, Université scientifique et médicale, Institut national polytechnique de Grenoble, 1983.

[34] É. SOPENA, *Éléments de théorie des graphes*, l'I.U.T de Technologie de l'Université Bordeaux 1.

[35] F. VIVIEN, *Algorithmique avancée*, IUP 2, 24 avril 2002.

[36] M.YAGOUNI, *Approche neuromimétique et le recuit simulé pour la résolution des problèmes complexes de l'optimisation combinatoire, cas du problème de voyageur de commerce***,** thèse de magistère, USTHB 1999.