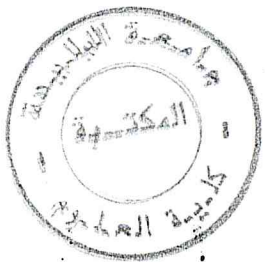
REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE

MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE SCIENTIFIQUE

UNIVERSITE SAAD DAHLEB BLIDA
FACULTE DES SCIENCES
INSTITUT D'INFORMATIQUE

MEMOIRE DE FIN D'ETUDES POUR L'OBTENTION DU DIPLOME D'INGENIEUR D'ETAT EN INFORMATIQUE

Option : Intelligence Artificielle



Thème :

Conception d'un algorithme d'optimisation multi-objectif et application dans un réseau de télécommunication

Réalisé par :

M^r Adel BALI

M^r Adel CHEMLAL

Encadré par:

Dr Mohamed Tounsi
Prince Sultan University,
Ryadh, Arabie Saoudite

Dr M^me Oukid Khouas
Université de Blida, Algérie

2003-2004

Remerciements

Dieu merci de nous avoir aidé à la réalisation de ce travail malgré les difficultés rencontrées

Nous remercions nos parents qui ont veillé sur nous depuis notre naissance, ainsi que toute la famille

Nous remercions notre promoteur Dr Mohamed Tounsi de nous avoir encadré et aidé dans ce travail malgré la contrainte de distance.

Nous remercions notre co-promotrice Dr Mme Oukid Khouas pour son suivi et ses conseils qui nous ont été très utiles.

Nous remercions notre directrice Mme Bensettiti pour son accueil

Nous remercions tous nos enseignants depuis notre premier jour à l'école à ce jour

Nous remercions tous ceux qui ont contribué de près ou de loin dans ce travail

C. Adel et B. Adel

DEDICACES

Je dédie ce travail a ma mère
A ma famille qui a été a mes cotés à tout moment

A mes cousins et cousines

A tous mes amis que j ai connu

A tous ce qui ont contribué de prés ou de loin dans ce modeste travail

B. Adel

DEDICACES

A mes très chers parents
qui ont été à mes cotés durant tout mes moments
difficiles et qui le sont toujours, et qui attendaient ce moment
depuis longtemps

A mes sœurs et frères qui n'ont pas hésité une seconde à me
soutenir dans toutes les circonstances, et sans lesquels ce travail
n'aurait pas existé

A tous mes amis que j'ai connus et que je n'oublierai jamais

A tous ceux qui ont contribué de près ou de loin dans ce travail

A tous, je dédie ce modeste travail

C. Adel

SOMMAIRE

Introduction générale :

On assiste de nos jours, à un énorme développement dans les différents secteurs de l'industrie (énergie, télécommunication, transport, santé,...etc.) exposant des problèmes complexes qui mettent en jeu des moyens financiers très importants, et auxquels les décideurs doivent faire face en cherchant des solutions optimales. L'optimisation classique consiste à minimiser un critère ou le maximiser, ou encore le maintenir dans un certain état. Cette optimisation est connue sous le terme d'*optimisation monocritère* ou *mono-objectif*. Mais actuellement, on assiste à des problèmes plus complexes introduisant plusieurs critères conflictuels (contradictoires), pour lesquels l'optimisation mono-objectif n'est plus efficace. L'*optimisation multicritère* (*multi-objectif*) s'intéresse à la résolution de ce type de problèmes. Elle a été utilisée initialement en économie durant le 19^{ieme} siècle [BER.01].

La résolution d'un problème multi-objectif ne fournit pas une solution unique mais un ensemble de solutions connu sous le nom de *l'ensemble Pareto Optimal*. Ceci résulte du fait qu'une solution peut être optimale sur un certain critère et peut ne pas l'être sur d'autres. Cette propriété impose une multitude de solutions qui optimisent chacune un critère ou plus et qui n'optimise pas les autres. Cette propriété est donnée par la notion de *non dominance* entre les solutions possibles.

Plusieurs travaux ont été réalisés dans le cadre de l'optimisation multi-objectif, comportant différentes approches de résolution telles que *les approches de transformation d'un problème multi-objectif en un problème mono-objectif* [H&M], englobant les critères existants en un seul critère à minimiser ou à maximiser. Cette approche présente des insuffisances au niveau des problèmes à critères fortement indépendants, pour lesquels la fusion est presque irréalisable. Une autre approche consiste à traiter séparément les critères et propose un compromis entre les solutions à la fin de la résolution. Cette approche est connue sous le nom de *l'approche non pareto*. Dans la réalité, cette approche est peu utilisée à cause des résultats non compromettants qu'elle fournit. L'approche la plus courante est l'approche Pareto, qui repose sur la génération de l'ensemble Pareto optimal déjà cité. Elle est fondée sur la notion de non dominance entre les solutions. Elle garantie l'indépendance entre les critères et traite chacun d'eux séparément. Cette approche comporte essentiellement deux types de méthodes : les méthodes exactes (programmation dynamique et A* multi-objectif [B&C.91]) et les méthodes approchées basées sur les métaheuristiques (recuit simulé, algorithmes génétiques [GOL.89] et recherche tabou [B&H.99]). Ces méthodes sont concernées par

différents types de problèmes tels que les problèmes de réseaux de télécommunication, problème du plus court chemin, pilotage automatique des avions... etc.

Notre travail rentre dans le cadre des méthodes d'optimisation exactes. Il constitue une suite du travail fait par BRADLEY S. STEWART et CHELSEA C. WHITE qui ont proposé l'idée de l'algorithme MOA* (Multi-Objective A*) littérairement et dans un état brut. Cet algorithme repose sur la recherche des chemins optimaux dans un graphe d'état. Nous avons présenté durant cette recherche, des améliorations sur cet algorithme en introduisant des heuristiques sur le traitement des nœuds pour réduire le nombre d'itérations de l'algorithme (inévitablement le temps d'exécution), sans détérioration des solutions trouvées. Nous avons également combiné l'algorithme MOA* avec une métaheuristique à savoir la méthode *TABOU*, sans pour autant affecter la propriété d'exactitude de cet algorithme. Les performances données par l'algorithme MOA*, dans ses différentes versions (avec heuristiques et métaheuristique), sont montrées à travers une étude comparative de ses différentes versions faite sur des exemples de test. Ces résultats montrent que les performances de chaque version dépendent du type du problème à résoudre (type du graphe d'état). Ce travail est doté d'une application sur un réseau cellulaire de télécommunication, présenté par France Télécom, dans une zone géographique où un nombre de sites potentiels a été défini. La résolution de ce problème revient à distribuer des antennes sur un minimum de sites potentiels et maximiser le trafic écoulé sur le réseau, tout en garantissant la couverture totale de la zone géographique par le réseau.

Organisation du mémoire

Le mémoire est organisé en 6 chapitres

Dans le premier chapitre, nous avons introduit les problèmes d'optimisation, les différentes méthodes d'optimisation, les heuristiques, et les métaheuristiques les plus courantes, ainsi que leur classification. Nous avons donné une attention particulière à l'algorithme A*, qui constitue la base de l'algorithme A* multi-objectif.

Le deuxième chapitre aborde l'aspect mono et multi objectif des problèmes d'optimisation, et ce en montrant, par le biais d'un exemple, l'incapacité des méthodes d'optimisation mono-objectif à résoudre une classe de problèmes qui est la classe des problèmes multi-objectif. Nous avons également introduit la notion d'ensemble Pareto, qui est une notion primordiale dans ce genre de problèmes, et les différentes approches de résolution des problèmes multi-objectif.

Dans le troisième chapitre, nous avons montré le passage du A* classique au A* multi-objectif (MOA*). Le MOA* a été présenté comme algorithme de résolution de problèmes multi-objectif : son principe, son fonctionnement, les données dont il a besoin pour son bon fonctionnement et la terminologie qui lui est associée. Son fonctionnement a été montré par un schéma illustrant son déroulement sur un exemple.

Le quatrième chapitre porte sur les améliorations apportées au MOA* : les différentes heuristiques appliquées et la métaheuristique de recherche Tabou. Ces trois versions font l'objet de comparaison avec le MOA* à l'état initial par des exemples bien choisis pour montrer le type de problèmes pour lequel chaque version apporte des améliorations par rapport aux autres.

Le chapitre cinq aborde la modélisation multicritère du problème du réseau cellulaire proposé par France Télécom, ainsi que sa modélisation en graphe d'état pour être résolu par les différentes versions de L'algorithme MOA*. Les différents résultats sont donnés dans un tableau comparatif.

Enfin le dernier chapitre présente la conclusion et les perspectives de ce travail.

Chapitre1

Méthodes d'optimisation et heuristiques

1. Méthodes de résolution et méthodes d'optimisation

La plupart des problèmes informatiques actuels sont formalisés sous forme d'un espace d'objets en interaction dans lequel une solution doit être trouvée. Le type des objets cherchés varient d'un problème à l'autre ; dans certains problèmes, l'objectif est simplement de présenter un objet satisfaisant un ensemble de critères (exemple : problème de l'échiquier) [PEA.90], mais pour d'autres, comme le problème du voyageur de commerce (PVC), en outre de la satisfaction des contraintes, l'objet trouvé doit posséder des qualités inégalées par aucun autre objet dans l'espace de possibilités. En revanche dans les autres problèmes, on voulait trouver n'importe quel objet qualifié avec aussi peu d'efforts que possible, une tache que nous appelons '*satisfaction*' (originellement inventée par H.Simon, 1956).

Les méthodes d'optimisation donc sont des méthodes à deux fonctions : l'une est de trouver les solutions possibles ; et l'autre consiste à trouver celle qui est la plus optimale. Ces méthodes sont fondées principalement sur diverses heuristiques. Elles peuvent être des méthodes empiriques utilisées pour guider nos actions.

2. Heuristiques

Les heuristiques sont des critères, des principes ou des méthodes permettant de déterminer parmi plusieurs lignes de conduite, celle qui promet d'être la plus efficace pour atteindre un but. Elle représente des compromis entre deux exigences : le besoin de rendre de tels critères simples et, en même temps, le désir de les voir établir une distinction entre les bons et les mauvais choix.

Les problèmes les plus complexes nécessitent l'évaluation d'un nombre immense de possibilités pour déterminer la solution exacte. La plupart du temps une vue humaine ne suffirait pas pour trouver une solution exacte. Les heuristiques jouent un rôle efficace dans de tels problèmes en indiquant un moyen de réduire le nombre d'évaluations et d'obtenir des solutions dans des délais raisonnables. Cette tache est extrêmement importante et délicate dans ce genre de situations où l'utilisateur n'a aucune idée sur l'intervalle de variation des solutions. La crédibilité des heuristiques, donc, n'est pas évidente que si elle est prouvée sur le terrain du problème à résoudre.

3. Exemple d'un problème nécessitant une optimisation

Problème du réseau routier : Soit un réseau routier comme celui montré dans la figure 1 ; les sommets du graphe représentent les villes et les arcs entre deux sommets représentent les routes reliant les deux villes correspondant aux sommets. On cherche à trouver le plus court chemin entre la ville A et la ville B.

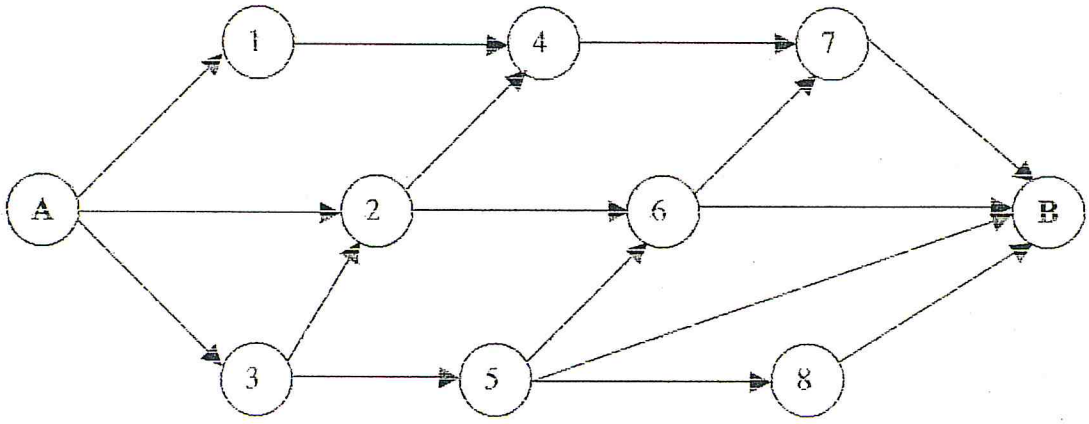


Figure 1 : exemple de réseau routier

A première vue, nous cherchons à trouver un chemin reliant la ville A à la ville B. Dans notre cas, il peut y avoir plusieurs. Mais le chemin demandé est celui avec un coût minimal, c'est pour cela qu'on classe ce problème parmi les problèmes d'optimisation.

Les problèmes d'optimisation occupent actuellement une place de importante dans la communauté scientifique. Non pas qu'ils aient été un jour considérés comme secondaires mais l'évolution des techniques informatiques a permis de dynamiser les recherches dans ce domaine.

4. Classification des problèmes d'optimisation

Plusieurs critères de classification des problèmes d'optimisation sont présents, selon la nature du problème lui-même. Parmi ces critères on peut citer [BER.01]:

- Problèmes combinatoires ou à variables continues

- Problèmes à un ou plusieurs objectifs
- Problèmes statiques ou dynamiques
- Problème dans l'incertain.

5. Classification des méthodes d'optimisation et des heuristiques les plus courantes

La classification suivante des algorithmes ne signifie en aucun cas qu'un problème doit être résolu avec une seule méthode ou une seule heuristique. En effet, des méthodes de résolution de problèmes complexes sont définies à partir d'une ou plusieurs approches de résolution [MEU.02].

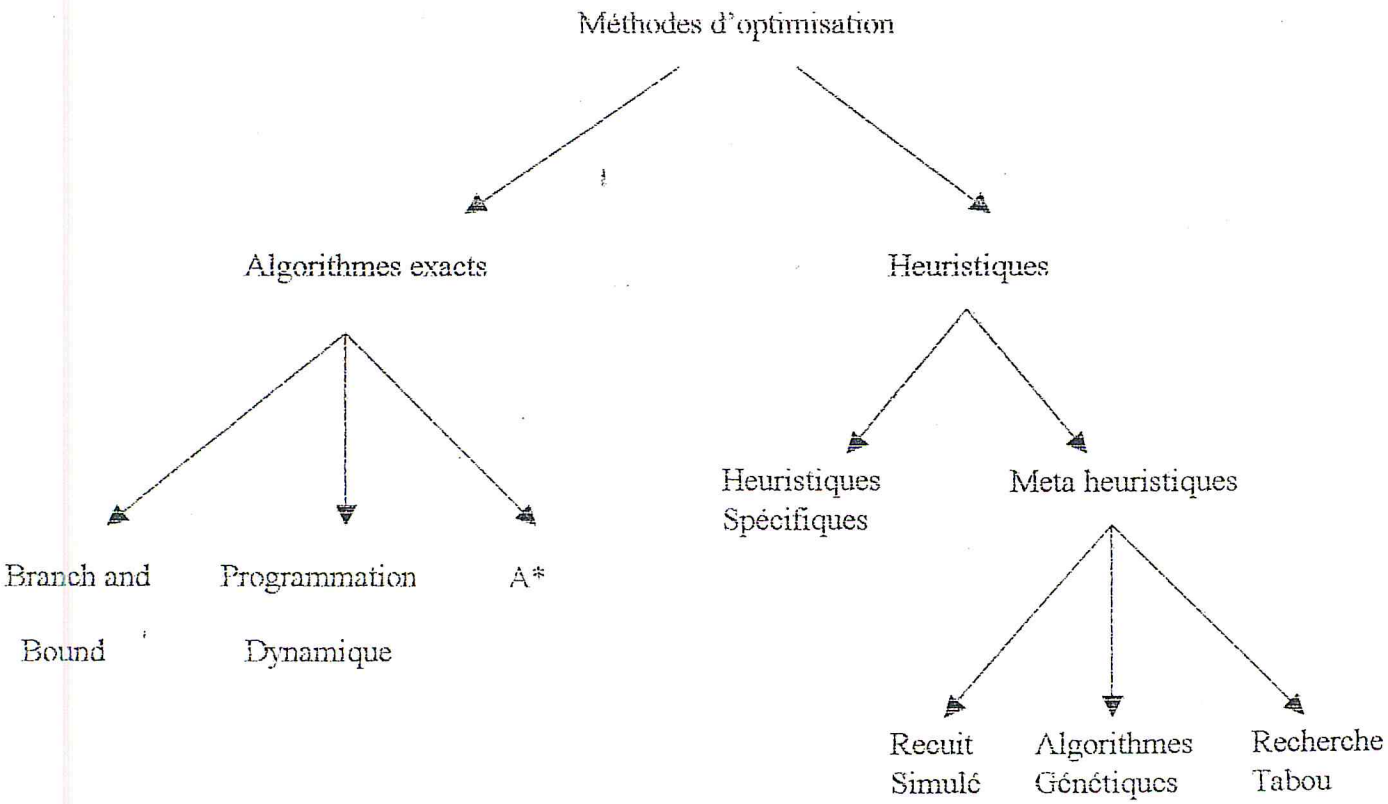


Figure 2 : classification des heuristiques [MEU.02]

Avant d'aborder les différentes méthodes de résolution d'un problème, nous devons donner quelques définitions nécessaires pour la modélisation du problème à résoudre. Ces définitions concernent les outils de traduction du problème littéraire à un model de résolution mathématique (théorie des graphes).

5.1. Définitions

Un problème d'optimisation est défini par un espace d'états, une ou plusieurs fonction(s) objectif(s) et un ensemble de contraintes.

L'espace d'état est défini par l'ensemble des domaines de définition des variables du problème. Dans la plupart des problèmes cet espace est fini car la méthode de résolution utilisée a besoin de travailler dans un espace restreint.

Les variables du problème peuvent être de nature diverse (réelle, entière, booléenne, etc.) et exprimer des données qualitatives ou quantitatives.

Une fonction objectif représente le but à atteindre pour le décideur (minimisation de coût, de durée, d'erreur, maximisation de revenue, etc.). Elle définit un espace de solutions potentielles au problème.

Ensemble de contraintes définit des conditions sur l'espace d'état que les variables doivent satisfaire. Ces contraintes sont souvent des contraintes d'inégalité ou d'égalité et permettent en général de limiter l'espace de recherche.

Une méthode d'optimisation recherche le point ou un ensemble de points de l'espace des états possibles qui satisfait au mieux un ou plusieurs critère(s). Le résultat est appelé *valeur optimale* ou *optimum*.

Dans ce qui suit, nous décrivons en bref les différentes stratégies citées dans l'arbre de classification (figure 2) en donnant une attention particulière à l'algorithme A* qui constitue la base de notre travail, sans aborder l'aspect mono ou multi-objectif de cet algorithme. Cet aspect fera l'objet du chapitre suivant.

5.2 *Méthodes d'optimisation exactes*

5.2.1 Méthode Branch and Bound (Séparation et Evaluation)

Le principe de cette technique est d'utiliser une liste triée et de sélectionner le nœud au coût minimal : au début, le nœud de départ est inséré dans la file, puis on génère ses successeurs et on les insère dans la liste triée : C'est donc une technique qui tient compte du passif [ZEG.INI].

Enfoncé :

1. Placer le nœud début de longueur 0 dans la liste.
2. Répéter jusqu'à ce que liste vide ou nœud recherché trouvé :

a) Si la première branche contient le nœud recherché, fin avec succès.

c) Sinon

- supprimer la branche de la liste et former des branches nouvelles en étendant la branche supprimée d'une étape.

- calculer les coûts cumulés des branches et les ajouter dans la liste de telle sorte que la liste soit triée en ordre croissant.

3. Autrement " pas d'élément"

5.2.2. Programmation dynamique

Le principe de la programmation dynamique est de diviser le problème à résoudre en plusieurs sous problèmes identiques, et de ne résoudre qu'un seul sous problème à la fois en sauvegardant les résultats (en utilisant une table de résultats déjà calculés, remplie au fur et à mesure qu'on résout les sous problèmes)

Remarque : C'est une méthode ascendante : On commence d'habitude par les sous problèmes les plus petits et on remonte vers les sous problèmes de plus en plus difficile [ZEG.INI].

5.2.3 Algorithme A* :

L'algorithme A* est un algorithme de recherche d'une solution optimale dans un graphe et qui utilise la programmation dynamique (division du graphe en sous branches et utilisation du même traitement pour les sous branches).

5.2.3.1 Données de l'algorithme:

- deux listes Ouvert (O) et Fermé (F) dont l'une est une file d'attente avec priorité (Ouvert).
- Nœud de départ S
- g : coût réel au nœud n
- h: estimation de la distance restante
- f = g + h

5.2.3.2 Enoncé de l'algorithme : [ZEG.INI]

1. O <-- nœud initial s.

2. Retirer un nœud n de Ouvert qui a la plus petite valeur de f(n). Si c'est le nœud but, stop avec succès. Autrement

- F <-- n
- Générer les successeurs n'
- Pour chaque successeur n' :

Si n' non dans O et n' non dans F :

. Calculer f(n') = g(n') + h(n')

. O <-- n'

. attacher un pointeur arrière vers n

Fsi

Si n' est dans O ou n' est dans F :

. Changer le chaînage arrière selon le coût de la plus petite branche (g)

Fsi

Si n' est dans F et son chaînage arrière est modifié:

. l'enlever de Fermé

. Enlever tous les nœuds de la descendance de n' qui sont dans O et dans F.

. O <-- n'.

Fsi

Fin pour

3. aller à 2

- Remarques à propos de g :

— La fonction g nous laisse choisir quel nœud élargir sur la base de tout le chemin depuis la racine. Ce nœud peut ne pas être le meilleur.

— Si nous voulons arriver à une solution d'une façon ou d'une autre, il suffit de donner à g la valeur 0. (Cas des algorithmes gloutons)

— Si nous voulons trouver un chemin (solution) avec le plus petit nombre de pas possible, il suffit de donner à g la valeur 1.

— Si nous voulons trouver le chemin le moins coûteux, il suffit de donner à g le coût du trajet (ou de l'opération).

— Ainsi A* peut être utilisé pour trouver un chemin au moindre coût ou un chemin quelconque aussi rapide possible.

- Remarques à propos de h :

— Si h' est parfait, A* converge immédiatement vers la solution sans recherche. Meilleur est h' plus nous approchons de plus près de la solution.

— Si h' vaut 0, la recherche sera contrôlée uniquement par g. Si g vaut 0, la recherche est aléatoire. Si g vaut toujours 1, la recherche se fera en largeur.

— *** Si h' ne surestime jamais h (distance réelle), A* est assuré de trouver un chemin optimal vers un but s'il existe. (propriété d'admissibilité)*

5.3 Métaheuristiques

Les métaheuristiques sont des techniques utilisées dans l'optimisation multi-objectif. Elles sont fondées sur des principes généraux et s'adaptent facilement à différents types de problèmes. Elles sont généralement utilisées comme amélioration pour d'autres méthodes. Dans ce cas on parle de méthodes hybrides (combinaison de méthodes élémentaires) : on site

brièvement le principe et les applications des trois métaheuristiques les plus courantes dans le domaine de l'optimisation et qui sont : recuit simulé, algorithmes génétiques et la méthode tabou. Dans notre cas, la méthode tabou est intégrée avec le A* multi-objectif (MOA*) et elle fera l'objet de cette étude dans le chapitre 4.

5.3.1 Recuit simulé : Le recuit simulé minimise de manière stochastique une fonction de coût. C'est une méthode de recherche locale dans la mesure où l'algorithme procède par de petites modifications de la solution courante, en autorisant éventuellement une dégradation temporaire de la qualité de la solution. Elle est appliquée essentiellement dans le problème du voyageur de commerce multi-objectif, le design du réseau de transport, et dans le problème du sac à dos multi-objectif.

5.3.2 Algorithmes génétiques : Les algorithmes génétiques nécessitent un codage des solutions potentielles semblable à l'identité génétique. L'algorithme procède par mutation et croisement de ces codes en sélectionnant les solutions de qualité élevée. Ils ont été utilisés pour résoudre plusieurs PMO transformés en problème mono-objectif, ordonnancement, planification de robots, génération de structures chimiques, placement et transport.

5.3.3 La recherche tabou : La recherche tabou est une méthode déterministe qui repose sur une gestion dynamique de la mémoire : La mémoire contient des caractéristiques dites tabou i.e. que l'on s'interdit pour les futures solutions. L'objectif premier est d'éviter une exploration cyclique pour augmenter l'efficacité de la recherche. Un jeu de poids peut être fixé selon la priorité des objectifs.

6. Conclusion

Les méthodes d'optimisation telles présentées dans cette partie font l'objet de recherche et de développement par les constructeurs pour affiner leurs résultats ; et ceci en les combinant avec des heuristiques et/ou des métaheuristiques pour subvenir aux besoins du domaine d'optimisation. En réalité, il n'existe pas de méthode efficace ou inefficace parmi ces méthodes. Chaque méthode donne ses performances dans un domaine précis : le choix d'une méthode revient a la nature du problème et des objectifs.

Pour le moment, nous avons présenté les techniques les plus courantes dans le domaine d'optimisation sans avoir abordé l'aspect mono ou multi-objectif. Cet aspect est

indispensable dans la résolution des problèmes et est le critère le plus prioritaire dans le choix de la méthode de résolution. Cet aspect fera l'objet du chapitre suivant.

Chapitre 2

Optimisation Mono-objectif et Optimisation Multi-Objectif

Cette partie s'intéresse à un aspect parmi les quatre déjà cités dans la classification des problèmes d'optimisation. Cet aspect est l'aspect « objectif » qui occupe actuellement une place très importante dans la classification. Un problème est soit mon-objectif, soit multi-objectif, et il n'y a pas de troisième alternative, ce qui rend la classification déterministe. La détermination de la nature du problème dans ce type de classification est la clé de sa résolution. Notons que notre travail s'intéresse à la résolution des problèmes multi-objectif.
Pour bien définir cet aspect, nous commençons par quelques définitions concernant la littérature du problème.

1. Définitions

Objectif

Un objectif indique le sens de l'amélioration qu'un décideur souhaite apporter à un système lors d'un changement d'état. Il reflète l'aspiration du décideur. Les trois manières de poursuivre un objectif sont de le maximiser, de le minimiser ou de le maintenir dans un certain état. Des exemples industriels classiques de ces situations sont : maximiser le profit, minimiser le coût ou maintenir un équilibre économique.

Critère

La signification du mot critère selon le dictionnaire (Le Robert) est « ce qui sert de base à un jugement, ce qui permet de distinguer une chose, une notion ». En théorie de la décision, il correspond à un attribut ou à un objectif. Dans ce sens, un problème multicritère désigne un problème multi-objectif.

2. Optimisation mono-objectif et optimisation multi - objectif

De nombreux secteurs de l'industrie (mécanique, chimie, télécommunications, environnement, transport, etc.) sont concernés par des problèmes complexes de grande dimension et multicritères (minimisation de risques, maximisation de la fiabilité, minimisation de coût, etc.) mettant en jeu des coûts financiers très importants et pour lesquels les décisions doivent être prises de façon optimale.

Le principal but de l'optimisation mono-objectif est de trouver la « meilleure solution », qui correspond à la valeur minimale ou maximale d'une fonction objectif. Ce type d'optimisation est utile pour la résolution de problèmes à plusieurs fonctions objectif non

conflictuelles (pas d'interaction entre elles) : et ceci en traitant chaque fonction à part. Mais hélas, dans la plupart des problèmes à plusieurs objectifs, les critères sont conflictuels et contradictoires: l'optimisation d'un critère se fait au détriment d'un autre. Ce qui entraîne l'impossibilité d'optimiser chaque critère à part.

L'optimisation multi-objectif vient pour remédier à ce problème: elle cherche des compromis entre les conflits et propose un ensemble de solutions dont aucune solution n'est meilleure que l'autre en terme d'optimalité. Cet ensemble est connu sous le nom de « ensemble non-dominé de solutions » ou « ensemble Pareto optimal (PO) ».

L'optimisation multi-objectif cherche donc à optimiser plusieurs composantes d'un vecteur fonction coût. Contrairement à l'optimisation mono-objectif, la solution d'un problème multi-objectif n'est pas unique, mais un ensemble (Pareto Optimal). Toute solution de cet ensemble est optimale dans le sens qu'aucune amélioration ne peut être faite sur une composante du vecteur sans dégradation d'au moins une composante du vecteur. Le premier objectif dans la résolution d'un problème multi-objectif est de trouver l'ensemble Pareto optimal. La détermination de l'ensemble PO n'est qu'une première phase dans la résolution pratique de PMO, qui nécessite dans un deuxième temps le choix d'une solution à partir de cet ensemble selon des préférences choisies par le décideur

CHOIX D'UNE SOLUTION :

Le choix d'une solution par rapport à une autre nécessite la connaissance du problème et des nombreux facteurs liés au problème. Ainsi une solution choisie par le décideur peut ne pas être acceptable pour un autre décideur. Le choix d'une solution peut aussi être remis en cause dans un environnement dynamique. Il est donc utile d'avoir plusieurs alternatives dans le choix d'une solution de l'ensemble Pareto optimal.

3. Exemple d'un problème mono-objectif

Soit le problème du calcul de la distance minimale dans un réseau routier cité dans le chapitre précédant (calcul de la distance minimale entre la ville A et la ville B). Ce problème est modélisé par un graphe d'état dont :

- Chaque nœud représente une ville.
- Le nœud de départ est la ville A
- Le nœud d'arrivée est la ville B

- Les arcs entre les nœuds représentent les chemins entre les villes, et les valeurs des arcs représentent les distances

Critère à optimiser : Distance entre la ville A et la ville B :

Cette distance est le cumul des distances entre les villes intermédiaires. Comme, il peut y avoir plusieurs combinaisons possibles pour arriver à la ville B, l'optimisation se fait sur le coût du chemin entre le nœud de départ et le nœud d'arrivée dans le graphe, en proposant une meilleure combinaison.

Remarques :

☞ Ce problème est un problème d'optimisation mono-objectif (mono-critère) car, le critère à optimiser est unique et (la distance). L'algorithme de résolution dans ce cas est un algorithme d'optimisation mono-objectif.

☞ La résolution de ce problème devient impossible par un algorithme d'optimisation mono-objectif si on introduit un autre critère conflictuel au critère existant. Par exemple, si on modifie les données du problème en introduisant les frais de transport (par moyen de transport payant : bus, train ou taxi), ou bien la durée de déplacement, l'optimisation portera sur la distance et également sur les frais de transport / durée de déplacement. Si on optimise le premier critère à part, on trouvera un chemin optimal en terme de distance, mais ce chemin peut ne pas être optimal en terme de frais de transport (ou temps de déplacement). La même chose pour le contraire. Ce qui résulte du fait que ce problème est multi-objectif (multicritère).

4. Exemple d'un problème multi-objectif

Problème de réseau routier a deux critères

Étant donné un réseau routier dans une localité constituée de plusieurs villes reliées par des chemins. Étant données les critères de distance et de temps de déplacement, quel est le chemin qui optimise en même temps la distance et le temps de déplacement entre la ville A et la ville B ?

Modélisation du problème : Ce problème est modélisé par un graphe dont :

- Chaque nœud représente une ville.
- Le nœud de départ est la ville A
- Le nœud d'arrivée est la ville B
- Les arcs entre les nœuds représentent les chemins reliant les villes, mais contrairement au problème précédant, les valeurs des arcs sont représentées par des couples de valeurs (D, T) où : D représente la distance entre deux villes et T représente le temps nécessaire pour traverser cette distance.

Critères à optimiser : contrairement à l'exemple précédant, il y a deux critères à optimiser simultanément, car ces deux critères peuvent être conflictuels.

☞ Le problème qui se pose ici est comment faire la comparaison entre deux valeurs à deux critères. Ou d'une autre manière, comment définir une notion de préférence entre deux couples de valeurs contradictoires.

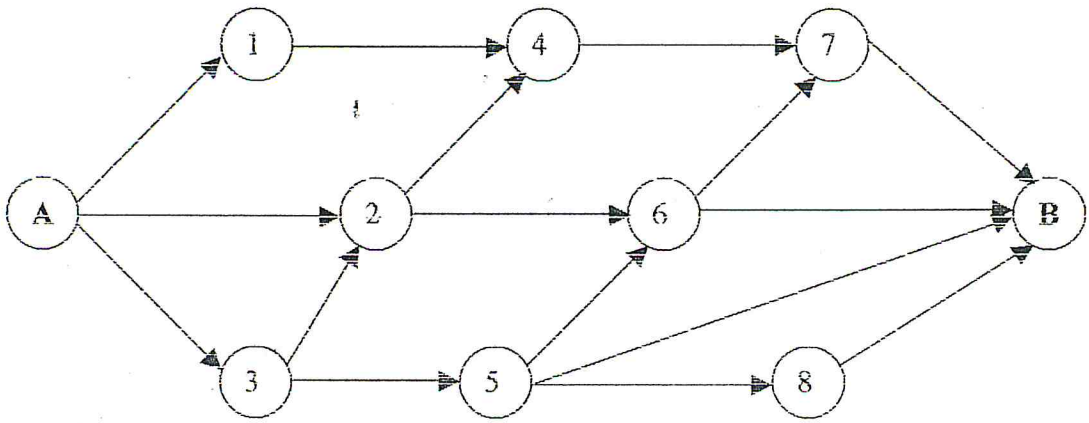


Figure 3 : exemple de réseau routier

Exemple : sachant qu'aucun critère n'est préférable par rapport à l'autre, étant donnés les deux chemins trouvés entre A et B suivants :

C1 : la distance = 5 Km ; le temps de déplacement = 17 minutes.

C2 : la distance = 5,50 Km ; le temps de déplacement = 15 minutes.

- Le chemin C1 est il meilleur que le chemin C2 ?

Le chemin C1 est meilleur que C2 en matière de distance, mais le chemin C2 est meilleur que C1 en matière de temps de déplacement. Et l'on sait qu'aucun critère n'est préféré qu'un autre. Dans ce cas, on dit que la valeur de C1 est non dominée par la valeur de C2 et la valeur de C2 est non dominée par la valeur de C1. Ce qui donne qu'aucun chemin n'est meilleur que l'autre.

- Par contre, le chemin C1 est meilleur que le chemin C3 défini comme suit :

C3 : la distance = 5,40 Km ; le temps de déplacement = 18 minutes.

Car sur les deux critères, le chemin C1 est optimal. Dans ce cas on dit que la valeur de C1 domine la valeur de C3, ou bien la valeur de C3 est dominée par la valeur de C1. Mais, la valeur de C1 est non dominée par la valeur de C3.

☞ On voit donc, à travers cet exemple, que la résolution des problèmes multi-objectif est différente de l'optimisation mono-objectif dans une notion de base qui est la notion de préférence entre les valeurs durant, et à la fin de la recherche, ce qui entraîne la multitude des solutions dans l'optimisation multi-objectif.

5. Modélisation d'un problème multi-objectif

Un problème multi-objectif ou multicritère peut être défini comme un problème dont on cherche l'action qui satisfait un ensemble de contraintes et qui optimise un vecteur de fonctions objectif. Le paragraphe suivant donne la définition mathématique d'un problème multi-objectif. Nous conserverons les mêmes notations dans la suite de ce document.

5.1 Notations et définitions

☞ *Une action* (ou *un vecteur de décision*) sera notée :

$x = (x_1, x_2, ...x_k)$

Avec x_i les variables du problème et k le nombre de variables.

☞ Un PMO peut être défini de la manière suivante :

$$\text{PMO} \begin{cases} \min/\max F(x) = (f_1(x_1), f_2(x_2), ... f_n(x_n)) \\ x \in C \end{cases}$$

Ou $n \geq 2$ est le nombre de fonctions objectifs, $x = (x_1, x_2, ...x_k)$ est le vecteur de décision, C représente l'ensemble des solutions réalisables associées à des contraintes d'égalité, d'inégalité et des bornes explicites (espace de décision) et $F(x) = (f_1(x_1), f_2(x_2), ...f_n(x_n))$ est le vecteurs des critères à optimiser [BER.01].

Dans le cadre de l'optimisation multi-objectif, le plus souvent le décideur raisonne plutôt en terme d'évaluation d'une solution sur chaque critère et se place naturellement dans l'espace des critères. L'ensemble $Y = F(C)$ représente les points réalisables dans l'espace des critères (espace des objectifs), et $y = (y_1, ..., y_n)$ avec $y_i = f_i(x)$ représente un point de l'espace des critères. On impose une relation d'ordre partiel sur cet ensemble de points, appelée *relation de dominance*.

5.1.1 Notion de dominance

La notion de dominance est tirée telle qu'elle est de la littérature [MEM.03], et qui exprime la faveur que possède une entité sur une autre de manière générale. Sa définition est déterminée selon le genre du problème (minimisation, maximisation, ou hybridation des deux).

Définition de dominance dans un problème de minimisation : [BER.01]

Une solution $y = (y_1, ..., y_n)$ domine une solution $z = (z_1, ..., z_n)$ ssi $\forall i \in [\![1..n]\!]$ $y_i \leq z_i$ et $\exists i \in [\![1..n]\!] / y_i < z_i$.

Définition de dominance dans un problème de maximisation : [BER.01]

Une solution $y = (y_1, ..., y_n)$ domine une solution $z = (z_1, ..., z_n)$ ssi $\forall i \in [\![1..n]\!]$ $y_i \geq z_i$ et $\exists i \in [\![1..n]\!] / y_i > z_i$

Dominance dans un problème de minimisation/maximisation :

Elle est définie par la formule de minimisation pour les critères à minimiser, et la formule de maximisation pour les critères à maximiser.

EXEMPLE :

Soit le problème qui consiste à minimiser le critère y_1 et à maximiser le critère y_2.

Une solution $A = (y_{11}, y_{21})$ domine la solution $B = (y_{12}, y_{22})$ si et seulement si :

$y_{11} \leq y_{12}$ et $y_{21} \geq y_{22}$ et $(y_{11} \neq y_{12}), (y_{21} \neq y_{22})$

Nous avons rarement un vecteur x^* qui est optimum pour tous les objectifs :

$\forall x \in C, f_i(x^*) \leq f_i(x), i = 1,2,...,n.$

Etant donné que cette situation arrive rarement dans les problèmes réels où les critères sont en conflit, d'autres notions ont été établies pour considérer une solution optimale. Le concept généralement utilisé est la notion d'*optimalité Pareto*.

5.1.2 Ensemble Pareto Optimal

Au XIX^{ieme} siècle, Vilfredo Pareto, un mathématicien italien, formule le concept suivant : dans un problème multi-objectif, il existe un équilibre tel que l'on ne peut pas améliorer un critère sans détériorer au moins un des autres critères [BER.01].

Définition 1 : Une solution $x^* \in C$ est Pareto optimale ssi il n'existe pas une solution $x \in C$ tel que $F(x)$ domine $F(x^*)$.

La définition de solution Pareto optimale découle directement de la notion de dominance. Elle signifie qu'il est impossible de trouver une solution qui améliore les performances sur un critère sans que cela entraîne une dégradation des performances sur au moins un autre critère. Les solutions Pareto optimales sont aussi connues sous le nom de solutions *admissibles*, *efficaces*, *non-dominées*, et *non-inférieures* [BER.01]. L'ensemble exact de toutes les solutions Pareto optimales sera noté PO. ND(S) dénote l'ensemble des solutions non-dominées de l'ensemble S [GOL.89].

Définition 2 : Le *vecteur idéal* $y^* = (y_1^*, y_2^*, ..., y_n^*)$ est le vecteur qui optimise chacune des fonctions objectif f_i i.e : $y_i^* = \min(f_i(x)), x \in C$

Le vecteur idéal est généralement une solution utopique, dans le sens où il n'appartient pas à l'espace objectif réalisable. Dans certains cas, le décideur définit un vecteur de référence exprimant le but qu'il veut atteindre pour chaque objectif. Ceci généralise la notion de vecteur idéal.

Définition 3 : *Un vecteur de référence* $z^* = (z_1^*, z_2^*, ..., z_n^*)$ est un vecteur qui définit le but à atteindre pour chaque objectif f_i.

Enfin, nous présentons la notion d'*optimum Pareto local*. Pour cela, commençons par définir la notion de voisinage qui est utilisée dans les méta heuristiques de recherche locale comme la recherche tabou.

Définition 4 : *Un voisinage* N est une fonction $N : C \rightarrow P(C)$, qui associe pour chaque $x \in C$ un sous-ensemble $N(x)$ de C des voisins de x.
Ce voisinage est défini par un opérateur de recherche locale.

Définition 5 : Une solution est *localement Pareto optimale* ssi $\forall w \in N(x), w$ ne domine pas x.

5.2. Problématique

La difficulté principale d'un problème multi-objectif est qu'il n'existe pas une définition de la solution optimale. Le décideur peut simplement exprimer le fait qu'une solution est préférable à une autre mais il n'existe pas une solution meilleure que toutes les autres.
Dès lors, résoudre un problème multi-objectif ne consiste pas à rechercher la solution optimale mais l'ensemble des solutions satisfaisantes pour lesquelles on ne pourra pas effectuer une opération de classement. Les méthodes de résolution de problèmes multi-objectif sont donc des méthodes d'aide à la décision car le choix final sera laissé au décideur.
Pour répondre à ce problème, la communauté scientifique a adopté deux types de comportement. Le premier est de ramener un problème multi-objectif à un problème simple objectif (mono-objectif) au risque d'enlever toute signification au problème. Mais cette approche contient des risques énormes de déviation de la solution dans le cas de valeurs de critères éloignés. Le second comportement est de tenter d'apporter des réponses aux problèmes en prenant en compte l'ensemble des critères. Cette partie de la communauté scientifique a emmené ces dix dernières années un grand nombre d'innovations dans les méthodes de résolution.

6. Approches de résolution de problèmes multi-objectif

Les approches utilisées pour la résolution de PMO peuvent être classées en trois catégories (figure.3) :

– approches basées sur la transformation du problème en un problème mono-objectif : Cette classe d'approches comprend par exemple les méthodes basées sur l'agrégation qui combinent les différentes fonctions coût f_i du problème en une seule fonction objectif. Ces approches nécessitent pour le décideur d'avoir une bonne connaissance de son problème ;

– approches Non-Pareto : les approches Non-Pareto ne transforment pas le PMO en un problème mono-objectif. Ils utilisent des opérateurs de recherche qui traitent séparément les différents objectifs ;

– approches Pareto : les approches Pareto utilisent directement la notion d'optimalité Pareto dans leur processus de recherche. Le processus de sélection des solutions générées est basé sur la notion de non-dominance.

Dans les sections suivantes, nous présentons respectivement les trois classes d'approches de résolution de problèmes d'optimisation combinatoire multi-objectif.

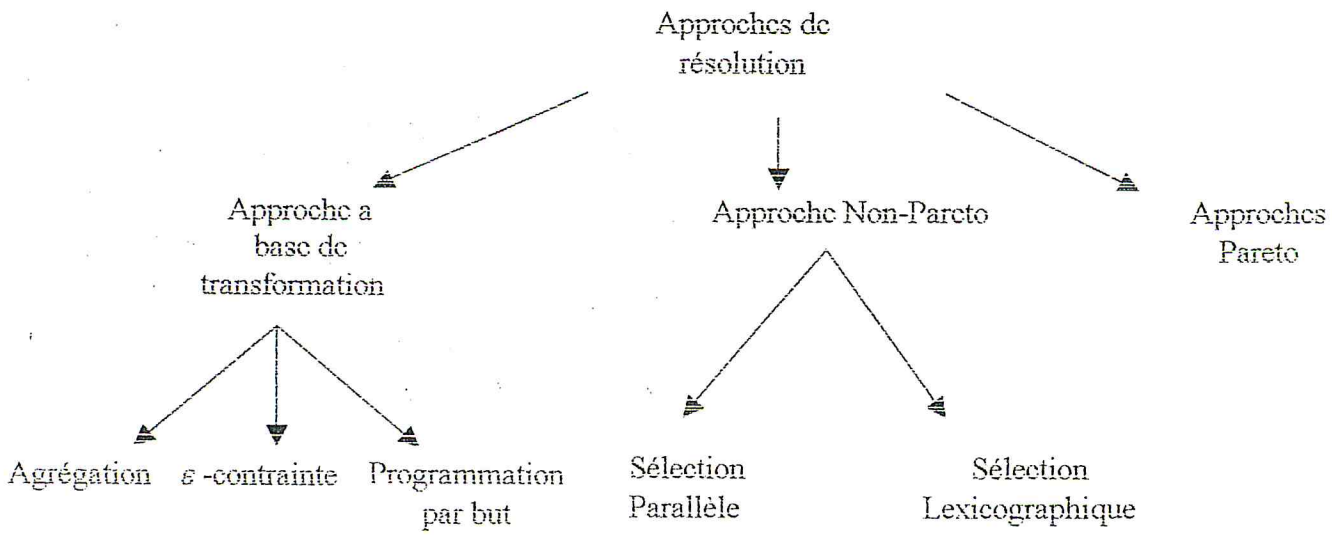


Figure 4 : différentes approches de résolution d'un PMO

6.1 Approches à base de transformations du problème vers le mono-objectif

Dans la résolution de PMO, plusieurs méthodes traditionnelles transforment le PMO en un problème mono-objectif. Parmi ces méthodes on trouve les méthodes d'agrégation, les méthodes ε-contrainte, et les méthodes de programmation par but.

6.1.1 Méthode d'agrégation

C'est l'une des premières méthodes utilisées pour la génération de solutions Pareto optimales. Elle consiste à transformer le problème (PMO) en un problème (PMO λ) qui revient à combiner les différentes fonctions coût f_i du problème en une seule fonction objectif F généralement de façon linéaire [H&M.79] :

$$F(x) = \sum_{i=1}^{n} \lambda_i f_i(x)$$

Où les poids $\lambda_i \in [0..1]$ et $\sum_{i=1}^{n} \lambda_i = 1$. Différents poids fournissent différentes solutions.

Les résultats obtenus dans la résolution du problème (PMO λ) dépendent fortement des paramètres choisis pour le vecteur de poids λ. Les poids λ_i doivent aussi être choisis en fonction des préférences associées aux objectifs, ce qui est une tâche délicate. Ainsi, une approche généralement utilisée est de résoudre le problème (PMO λ) avec différentes valeurs de λ.

6.1.2 Méthode ε-contrainte

Dans cette approche, le problème consiste à optimiser une seule fonction objectif f_k sujette à des contraintes sur les autres fonctions objectif.

$$(PMO_k(\varepsilon)) \begin{cases} \min f_k(x) \\ x \in C \\ f_j(x) \le \varepsilon_j, j = 1,..,n, j \neq k \end{cases}$$

Où $\varepsilon = (\varepsilon_1,...,\varepsilon_{k-1},\varepsilon_{k+1},...,\varepsilon_n)$.

Ainsi, un problème mono-objectif (objectif f_k) sujet à des contraintes sur les autres objectifs est résolu. Différentes valeurs de ε_i peuvent être données pour pouvoir générer différentes solutions Pareto optimales. La connaissance a priori des intervalles appropriés pour les valeurs ε_i est requise pour tous les objectifs. Pour pouvoir définir les valeurs adéquates pour ε_i, le vecteur idéal doit être calculé pour déterminer les bornes inférieures. On aura donc :

$\varepsilon_i > f_i(x^*), i = 1,2,k-1,k+1,n$

Un ordre lexicographique peut être établi entre les objectifs. Dans ce cas, l'ensemble PO est obtenu en minimisant les objectifs, commençant par le plus prioritaire et suivant l'ordre décroissant des priorités. Supposons que les indices des fonctions objectifs désignent aussi la priorité ; la fonction f_1 est donc la plus prioritaire. Le premier problème résolu sera formulé de la manière suivante :

$\min f_1(x), x \in C$

Les contraintes ne sont pas prises en compte. Soit x_1^* la solution optimale trouvée. Le deuxième problème traité serait :

$\min f_2(x), x \in C$, Avec $f_1(x) = f_1(x_1^*)$

Une contrainte d'égalité est associée aux fonctions déjà optimisées. Ce processus est itéré jusqu'au traitement de la fonction f_n. La solution finale du problème sera x_n^*.

Cette méthode impose à l'utilisateur un ordre de priorité sur les critères dès les premiers moments de la résolution et ne donne pas la chance aux autres critères d'être favorisés, et pour tester une autre configuration de priorités on doit refaire tout le processus de résolution. Ce qui rend cette méthode inefficace dans le cas des problèmes nouveaux (non connus par l'utilisateur)

6.1.3 Programmation par but (Goal Programming) [MEU.02]

Dans cette méthode, le décideur doit définir des buts ou références qu'il désire atteindre pour chaque objectif. Ces valeurs sont introduites dans la formulation du problème, le transformant en un problème mono-objectif. Par exemple, la fonction coût peut intégrer une norme pondérée qui minimise les déviations par rapports aux buts.

Le problème de cette méthode réside dans la difficulté de la définition des buts à l'avance par l'utilisateur. Cette méthode exige une très bonne connaissance du comportement du problème

à résoudre et la variation de son état durant la résolution. Chose qui n'est pas évidente dans les problèmes industriels de grande taille où le décideur ne sait pas grande chose sur le comportement du problème à résoudre.

Analyse de la première classe de méthodes : la transformation du problème multi-objectif en un problème mono-objectif nécessite des connaissances à priori du problème traité. L'optimisation d'un problème mono-objectif peut garantir l'optimalité Pareto de la solution trouvée mais trouve une seule solution. Dans les situations réelles, les décideurs ont généralement besoin de plusieurs alternatives. En effet, si certains objectifs sont bruités ou des données sont incertaines, ces méthodes ne sont pas efficaces. Ils sont aussi sensibles au paysage de la frontière Pareto (convexité, discontinuité, etc.). L'autre inconvénient de ces méthodes est leur sensibilité par rapport aux poids, aux contraintes ou aux vecteurs de référence. Les solutions obtenues dépendent fortement de ces paramètres. Pour différentes situations, différents paramètres sont utilisés, et le problème doit être résolu plusieurs fois, d'où le coût associé à cette classe d'algorithmes, qui nécessitent parfois l'optimisation indépendante de chacun des objectifs. Dans l'exécution multiple des algorithmes, on espère trouver différentes solutions Pareto optimales.

Dans les deux classes suivantes de méthodes, ces difficultés peuvent être éliminées. Une seule résolution du problème permet de trouver un ensemble de solutions Pareto optimales. Le décideur peut ainsi choisir une solution suivant la situation courante. La connaissance de plusieurs solutions Pareto optimales est utile aussi pour une utilisation future, particulièrement quand la situation change et une nouvelle solution est requise. Nous présentons dans les sections suivantes comment ces méthodes surmontent les difficultés présentées précédemment.

6.2 Approches Non-Pareto

Dans les approches Non-Pareto basées sur les populations de solutions, la recherche est réalisée en traitant séparément les différents objectifs non commensurables (qui ne sont pas de même type).

L'inconvénient majeur de ces approches est qu'elles tendent à générer des solutions qui sont largement optimisées pour certains objectifs, et l'inverse pour les autres objectifs. Les solutions « compromis » sont négligées.

6.3 *Approches Pareto*

Les approches Pareto utilisent directement la notion de dominance dans la sélection des solutions générées, contrairement aux autres approches qui utilisent une fonction d'utilité ou traitent séparément les différents objectifs. Cette idée a été introduite initialement dans les algorithmes génétiques par Goldberg [GOL.89]. Le principal avantage de ces approches est qu'elles sont capables de générer des solutions Pareto optimales dans les portions concaves de la frontière Pareto.

6.4 Analyse des trois approches

Les deux premières approches sont appliquées dans des classes de problèmes très restreints. Quoi que l'utilisation de la deuxième (approche non-pareto) est plus large que celle de l'approche de transformation, elles restent peu praticables dans les problèmes à critères indépendants, qui est la caractéristique principale de la plupart des problèmes actuels. L'approche Pareto reste l'approche la plus utilisée actuellement dans la majorité des domaines existant. Ceci est dû au fait qu'elle maintient l'indépendance des critères du problème. Mais l'inconvénient inévitable de cette approche reste la largeur importante de la gamme des solutions. Cette largeur peut être atténuée par l'introduction d'autres contraintes sur les critères (contraintes d'égalité, d'inégalité, etc.). Mais ceci pourra affecter le problème lui-même.

7. Conclusion

Nous avons présenté dans ce chapitre la notion de résolution des problèmes mono-objectif et son insuffisance pour résoudre une certaine classe de problèmes. Cette classe est prise en charge par la classe des méthodes d'optimisation multi-objectif. Nous avons également donné la modélisation du problème multi-objectif, la notion d'ensemble Pareto, qui est une notion primordiale dans ce genre de problèmes, et les différentes approches de résolution des problèmes multi-objectif. Le prochain chapitre portera sur un algorithme efficace de résolution de problèmes multi-objectif appelé MOA* qui est dérivé de l'algorithme exact A* et dont l'idée est donnée par S. STEWART et CHELSEA C. WHITE de l'université de Virginia aux états unis. Cet algorithme sera sujet de mise en œuvre, d'amélioration et d'expérimentation.

Chapitre 3

Algorithme A* Multi-Objectif (MOA*)

1. Introduction

Dans cette partie nous allons présenter l'algorithme A* multi-objectif (**MOA***) comme algorithme de résolution de problèmes multi-objectif : son principe, son fonctionnement, les données dont il a besoin pour son bon fonctionnement et la terminologie utilisée. Le MOA a été proposé littérairement à l'état brut par BRADLEY S. STEWART et CHELSEA C. WHITE de l'université de Virginia aux Etats Unis. Cet algorithme a besoin d'être généralisé, affiné, développé et programmé. Mais tout d'abord, on fait un petit rappel sur le A* classique (mono-objectif) : son principe, ses insuffisances dans certaines applications, et les modifications apportées sur cet algorithme pour aboutir au A* multi-objectif. Il est indispensable de préciser que ce genre de travaux fait appel à une certaine logique pour sa conception. Dans un premier temps, nous allons présenter le principe de fonctionnement de cet algorithme et les données et terminologies qui lui sont appropriées.

2. Algorithme A*

2.1 Présentation et fonctionnement

L'algorithme A* effectue une recherche heuristique dans un graphe. Le graphe code usuellement l'espace de recherche associé aux états et aux opérateurs, mais il peut aussi être vu comme un graphe usuel. Il n'est pas nécessaire que le graphe soit complètement développé, mais seulement qu'il soit aisément constructible.

A la formalisation usuelle du A*, s'ajoute une notion de coût sur les opérateurs. Ceci permet de représenter quelques aspects de la réalité. Par exemple, supposons que le problème soit de trouver le plus court chemin entre deux villes, qu'un état corresponde à la ville où l'on se trouve et que les arcs correspondent au déplacement d'une ville à une ville adjacente. On sera alors intéressé par la distance séparant les villes. On pourra représenter cela par des coûts pour chaque opérateur, c'est-à-dire pour chaque arc entre deux états.

Remarquons que cette formalisation est plus générale que celle ne tenant pas compte des coûts. En effet, lorsque les coûts des opérateurs ne sont pas définis, ou pas distingués, il est possible de choisir un même coût pour tous.

L'algorithme A* permet de résoudre tout les problèmes de cette famille. Il peut les résoudre de façon exacte et sans exploiter aucune heuristique, avec la garantie de toujours trouver la (les) meilleure(s) solution(s). Mais sa formulation permet aussi d'intégrer des informations supplémentaires. Selon les propriétés que l'on pourra monter pour l'information

ajoutée, A* aura ou non un comportement garanti (i.e. l'assurance de trouver la meilleure solution).

A* considère pour chaque état n, une évaluation $f(n)$ du coût total d'un chemin allant de l'état initial à l'état final et passant par cet état. Ce coût se décompose en deux parties : $f(n) = g(n) + h(n)$, où g est une estimation du plus court chemin allant de l'état initial jusqu'à n, et h est une estimation du plus court chemin allant de n a l'état final. (f est donc bien une estimation du plus court chemin de l'état initial à l'état final passant par n).

Au fur et à mesure de l'exploration du graphe, l'évaluation de g s'affine, puisque l'on découvre de plus en plus de chemins partant de l'état initial. De plus, l'estimation de $g(n)$ est toujours une valeur majorée. En effet, une fois que l'on a trouvé un premier chemin allant jusqu'à N, on ne peut que découvrir des raccourcis, mais on ne peut détériorer le chemin déjà trouvé. Le principe de A* est donc de faire confiance à h, pour poursuivre son exploration en premier par les chemins qui semblent les plus prometteurs. L'ensemble des alternatives à un instant de l'exploration sera représenté par un ensemble de nœuds que l'on appellera OUVERT. Les chemins déjà vus seront retenus dans un autre ensemble que l'on nommera FERME.

2.2. Algorithme

```
OUVERT ← {état initial};
FERME ← { };
Tant que OUVERT ≠ { }
    Etat-courant ← tête (OUVERT);
    FERME = FERME ∪ état courant;
      Si etat-courant = etat-but alors rendre (chemin (etat-initial, etat-courant))
      Sinon
        Pour tout les successeurs s de etat-courant faire
          évaluer g(s) et h(s)
          Si s ∈ OUVERT ∪ FERME alors
            Si g (etat-courant) + c (etat-courant, s) ≤ g(s) alors
              mettre a jour père(s); insérer(s, OUVERT);
            Sinon insérer(s, OUVERT);
            rendre (échec).
          Fin Si
        Fin Pour
    Fin Si
Fin Tant Que.
```

Figure 5 : Enoncé de l'algorithme A* [TDI.01]

Structure de données de l'algorithme :

Etat-courant : représente le nœud du graphe en cours de traitement

Inserer(n, file) : désigne l'insertion du nœud *n* dans la file *file*.

Tête(file) : désigne la tête de la file *file*.

Etat-initial : désigne le nœud de départ.

Etat-but : désigne le nœud d'arrivée.

Dans cet algorithme, OUVERT et FERME sont des ensembles, c (n; m) est le coût de l'opérateur permettant de passer de n a m.

2.3 Propriétés

$h(n)$ est une estimation du plus court chemin allant de n à l'état final. Les valeurs de $h(n_i)$ sont généralement fixée selon la nature du problème à résoudre. Plus h est élevée, et moins on testera de chemins. En effet, il n'y aura que peu de chemins prometteurs, puisque pour tous, on donne une évaluation exagérée de la distance restant à parcourir. Des qu'un premier chemin aura été trouvé, les autres seront très certainement délaissés. On obtiendra alors bien un chemin de l'état initial a l'état final, mais il est peu probable qu'il s'agisse du plus court.
Partant de cette observation, on peut en prendre le contre pied pour déduire une première propriété de l'algorithme A* : si h est minorante, alors A* trouve toujours la meilleure solution. Que h soit minorante signifie que pour chaque nœud, elle promet un coût inférieur ou égal a la réalité ; c'est a dire qu'elle le rend plus prometteur qu'il ne l'est en fait. Ainsi, on sera toujours poussé à explorer le maximum de chemins pour s'assurer que celui que l'on a découvert est vraiment le meilleur, puisque h nous pousse à conserver l'espoir d'en trouver un plus court. Un cas particulier est celui correspondant a h = 0 pour tout nœud. On retombe alors dans le cas d'une exploration exhaustive de tous les chemins.

3. Passage de l'algorithme A* au MOA*

- la notion de meilleur coût dans le A* mono objectif est remplacée par la notion de **non-dominance** (déjà définie) dans le A* multi objectif.

- Une conséquence logique de cette notion de non-dominance : quand on trouve des ensembles de coûts où personne ne domine l'autre, comment garder trace de ces ensembles ? et lequel choisir pour continuer la recherche ?

  - On dispose d'une file ND qui a la même structure que OPEN et qui contiendra les éléments les non dominés entre eux et qui seront développés par la suite (prochains nœuds à traiter) : ceci remplace en A* mono objectif le choix de l'élément au coût minimal (f minimale).

  - Le choix du nœud à traiter peut se faire selon une heuristique choisie parmi plusieurs (on peut calculer par exemple la distance entre les coûts de

l'ensemble et prendre le minimum, comme on peut le choisir arbitrairement). Cet aspect sera étudié dans un paragraphe où plusieurs heuristiques seront citées, et le choix de la bonne heuristique revient aux résultats apportées par celle-ci (on choisira celle donne le résultat voulu le plus vite possible. C'est-à-dire, celle qui minimise le nombre d'itérations)

- Comme mentionné dans le chapitre précédant, les solutions dans ce genre de problèmes sont généralement pas uniques, à cause de la notion de non dominance, on pourra trouver une multitude de solutions où aucune n'est meilleure que l'autre. Mais avec l'introduction des heuristiques le choix des nœuds à traiter peut être réduit.

Dans ce qui suit nous citons quelques définitions utiles pour la description et la preuve du MOA*,

4. Concepts de base de l'algorithme MOA*

4.1 Définition mathématique de la notion de dominance

La définition suivante est la même que celle citée avant, mais avec un aspect mathématique. Elle sera utile dans la formalisation mathématique de l'algorithme A* multi-objectif.

Etant donné un ensemble Y de M vecteurs, on définit la relation $R \subset Y \times Y$ comme suit :

$$(y,y') \in R \Leftrightarrow y_i \leq y_i' \quad \text{pour tout } i \in 1 \ldots M.$$

Cette relation induit un ordre stricte R' sur Y défini comme suit :

$$(y, y') \in R' \Leftrightarrow (y, y') \in R \quad \text{et} \quad (y', y) \notin R$$

ou d'une autre manière :

$$(y, y') \in R' \Leftrightarrow y_i \leq y_i' \quad \text{pour tout } i \in 1 \ldots M \text{ et } y \neq y'.$$

On dit que y domine y' (dans Y) si $(y,y') \in R'$. Un élément $y \in Y$ est dit non dominé (dans Y) s'il n'existe pas un élément $y' \in Y$ tel que y' domine y (dans Y. on utilise la notation *nondom*{Y}= {$y \in Y$: il n'existe pas $y' \in (y',y) \in R'$}.

Cette relation de dominance forme la notion de préférence entre les chemins. On dit qu'un chemin est non dominé dans un ensemble Q de chemins si la valeur du vecteur de P est non-dominée dans l'ensemble des valeurs des vecteurs de Q.

4.2 Ensembles mathématiques concernant le graphe

Graphe des états **G**

G = (N(**G**),A(**G**)) est le graphe des états représentant le problème a résoudre. La correspondance détaillée entre le problème et sa représentation graphique constitue un souci majeur pour la résolution de ce problème. N(**G**) est l'ensemble des nœuds du graphe **G**.

$A(\mathbf{G}) \subseteq N \times N$ représente l'ensemble des arcs orientés reliant des paires de nœuds du graphe.

Pour toute paire de graphes **G** et **G'**, on dit que **G'** est contenu dans **G''**, écrit $\mathbf{G'} \subseteq \mathbf{G''}$, si $A(\mathbf{G'}) \subseteq A(\mathbf{G''})$ et $N(\mathbf{G'}) \subseteq N(\mathbf{G''})$.

Nœud de départ S

S est unique et $s \in N(\mathbf{G})$

Ensemble des états but Γ

Γ est un ensemble fini non vide de nœuds.

Chemin dans un graphe **G**

$P = (n_0, n_1, n_2, \ldots)$ consiste en une séquence d'au moins deux nœuds, $n_i \in N(\mathbf{G'})$, et des arcs associés $a_i = (n_i , n_{i+1}) \in \Lambda(G') \forall i = 1,2,3,\ldots$ n_0 est dit la racine du chemin P. un chemin possède au moins un arc(bien entendu 2 nœuds différents).

- On dit qu'un chemin P' = (N(P'),A(P')) est un sous-chemin de P = (N(P),A(P)) si $N(P') \subseteq N(P)$ et $A(P') \subseteq A(P)$.
- un chemin est cyclique s'il contient plus d'une occurrence d'un même nœud.
- la concaténation de deux chemins P1 et P2, dont l'arrivée de P1 est le départ de P2, donne un chemin qui commence par le nœud de départ de P1 et se termine par le nœud d'arrivée de P2.

Ensemble des chemins non-dominés

$P^*(X1, X2) = \text{nondom}\ \{P(X1, X2)\}$

C'est l'ensemble des chemins dont le nœud de départ $\in$ X1 et le nœud d'arrivée $\in$X2, et les valeurs des vecteurs de coûts sont non-dominées.

Ensemble des nœuds buts non-dominés Γ^*

$\Gamma^* \subseteq \Gamma$. Γ^* est l'ensemble des nœuds buts qui peuvent être atteints par l'ensemble des chemins non-dominés qui commencent du nœud de départ.

Cette définition signifie qu'il peut y avoir des chemins qui appartiennent a la solution et qui ne se terminent pas par certains nœuds (ces nœuds sont ceux de Γ^*).

$\Gamma^* = \{\gamma \in \Gamma : P(s,\gamma) \cap P^*(s,\Gamma) \neq \phi\}$

$P^*(s,\Gamma)$ est l'ensemble des chemins non-dominés commençant par le nœud de départ S et qui se termine par les nœuds buts (appelés aussi chemins solutions).

Vecteur des coûts d'un chemin

$c(P) = \{c_1(p), c_2(p), \ldots, c_m(p)\}$

$c_1, c_2, \ldots, c_m$ sont les coûts à maximiser ou à minimiser.

m est le nombre de ces critères.

Ensemble des vecteurs de coûts de tous les chemins solutions (non-dominés)

$C^* = \{c^*(p)\} = \{c(P) : P \in P^*(s, \Gamma)\}$

4.3 Algorithme MOA* (Multi-Objectif A*)

Avant d'illustrer l'énoncé et le principe de l'algorithme MOA*, nous allons énumérer les différents ensembles utilisés par cet algorithme ; ces ensembles sont soit des ensembles de nœuds, soit des ensembles de chemins à manipuler durant la résolution.

4.3.1 Ensembles utilisés

Ils sont au nombre de sept (7) : OPEN, CLOSED, ND, LABEL, SOLUTION, SOLUTION-COSTS et SOLUTION-GOALS. Dans la programmation, ces ensembles sont

représentés par des listes contenant des entités à manipuler (nœuds et chemins) et dont le contenu varie durant le l'exécution de l'algorithme.

☞ Ensemble OPEN

C'est l'ensemble des nœuds considérés des candidats potentiels pour le développement par cet algorithme (MOA*) à une certaine itération k. si $n \in OPEN_k$ et $n \notin OPEN_r \ \forall \ r< k$, on dit que le nœud n est développé a l'itération k-1.

☞ Ensemble des successeurs ou fils d'un nœud n

Succ(n) = {n' : (n, n') $\in$ A(G)}. n est dit père ou successeur des nœuds n' $\in$ Succ(n).
Le père d'un nœud n est désigné par la fonction inverse $Succ^{-1}(n)$.

☞ Ensemble CLOSED

Il est inclus dans l'ensemble N(G), l'ensemble des nœuds du graphe.
$CLOSED_k$ (CLOSED a l'itération k) c'est l'ensemble utilisé pour garder trace des nœuds qui ont été générés, mais qui ne sont pas dans $OPEN_k$. un nœud ne peut être développé que si tous ses parents sont générés

Remarque : $OPEN_k \cap CLOSED_k = \phi \ \ \forall k$

☞ Ensemble LABEL

C'est l'ensemble des coûts des chemins provenant de s (nœud de départ) à n' en passant par le dernier nœud n, et qui sont non-dominés parmi tous les chemins à l'itération k-1. On dit qu'il existe un chemin de retour de n' à n, à l'itération k-1 si et seulement si : $LABEL_k(n',n) \neq \Phi$.

☞ Ensemble SOLUTION

C'est l'ensemble des chemins de s (nœud de départ) jusqu'aux nœuds buts dont les coûts sont non-dominés.

☞ Ensemble SOLUTION-GOALS

C'est l'ensemble des nœuds solutions non-dominés trouvés associés aux chemins non-dominés dans LABEL. Se sont les nœuds d'arrivée dont les coûts sont non-dominés.

Définition : Ensemble des coûts non-dominés qui mènent à un nœud n à l'itération k : G(n)

$G_k(n) = \text{nondom}\{\cup \text{LABEL}_k(n,n') : n' \in \text{Succ}^{-1}(n)\}$.

$G_k(n)$ est l'ensemble des vecteurs des coûts de tous les chemins non-dominés parmi les chemins de s (nœud de départ) à n qui ont été découverts en premier au début de l'itération k. en d'autres termes, les coûts des meilleurs chemins de s à n.

☞ Ensemble SOLUTION-COSTS$_k$

C'est l'ensemble des coûts non-dominés des chemins solutions se trouvant dans SOLUTION-GOALS. En d'autres termes, c'est l'ensemble des coûts solutions.

Formellement :

$\text{SOLUTION-COSTS}_k = \text{nondom}\{c(P) : P \in \text{SOLUTION}_k\}$

$= \text{nondom}\{\cup G_k(\gamma) : \gamma \in \text{SOLUTION-GOALS}_k\}$

Définition 1

On appelle $G^*(n)$, l'ensemble des coûts non-dominés des chemins entre le nœud de départ s et le nœud n.

$G^*(n) = \{c(P) : P \in \mathbf{P}^*(s,n)\} = \text{nondom}\{c(P) : P \in \mathbf{P}(s,n)\}$

Définition 2

On appelle $H^*(n)$, l'ensemble des estimations des chemins non domines de n aux nœuds buts.

$H^*(n) = \{c(P) : P \in \mathbf{P}^*(n,\Gamma)\} = \text{nondom}\{c(p) : P \in \mathbf{P}(n,\Gamma)\}$.

Définition de la fonction heuristique h

$h : N(G) \rightarrow \Re^{M+}$, ou les valeurs h(n) sont sensés estimer un certain les membres de l'ensemble des coûts définis par $H^*(n)$ ci-dessus. On utilise $\Re^{M+}$ pour indiquer l'ensemble : $\{x : x \in \Re^M, x \geq 0\}$.

H est l'ensemble des fonctions heuristiques pour un problème précis.

H(n) est le sous-ensemble non-dominé des valeurs des fonctions heuristiques associé au nœud n. donc, $H(n) = \text{nondom}\{h(n) : n \in N(G)\}$.

Définition de la fonction de sélection F

$F_k(n)$: ensemble des valeurs de fonction du nœud n à l'itération k. F est la fonction qui détermine le nœud à sélectionner. L'ensemble $F_k(n)$ est défini comme suit :

$$F_k(n) = \begin{cases} \phi & \text{si } n \text{ n'est pas dans le sous - graphe parcouru à l'itération } k \\ \text{nondom}\{g + h : g \in G_k(n) \text{ et } h \in H(n)\} & \text{sinon} \end{cases}$$

$F_k(n)$: est l'ensemble des valeurs de sélection des nœuds associés avec les nœuds de $OPEN_k$:

$F_k = \{f \in F_k(n) : n \in OPEN_k\}$.

Définition de l'ensemble C_k

C'est l'ensemble des coûts à utiliser pour faire des sélections a partir de $OPEN_k$, qui se compose des valeurs de sélection (F_k) des nœuds de $OPEN_k$, et des coûts des chemins solutions trouvés (placés dans SOLUTION-COSTS).

$C_k = F_k \cup SOLUTION\text{-}COSTS_k$.

☞ Ensemble ND

Est un ensemble inclus dans OPEN ; il est formé des nœuds de OPEN qui ont des valeurs de fonction qui ne sont pas dominées ni par :

(1) le coût d'aucun chemin solution trouvée a l'itération k (i.e: dans $SOLUTION\text{-}COSTS_k$), ni par :

(2) les valeurs de fonction d'autres solutions représentées par les nœuds de $OPEN_k$.

$ND = \{n \in OPEN_k : F_k(n) \cap \text{nondom}\{C_k\} \neq \phi\}$.

Quelques propriétés concernant les ensembles OPEN et ND

1- Si un nœud n est sélectionné pour expansion durant l'itération k, alors $n \in ND_k \cap CLOSED_{k+1}$.

Effectivement, un nœud sélectionné pour expansion durant une itération k est forcement dans ND. Et ce nœud sera automatiquement mis dans la liste CLOSED à la fin de son traitement. Ce qui donne qu'à l'itération k+1, il sera dans la liste des nœuds fermés (CLOSED).

2- Si n n'est pas un nœud but et n est sélectionné pour expansion durant l'itération k, alors n est dilaté durant l'itération k.

3- La construction des ensembles OPEN et CLOSED assure la propriété suivante :

$$OPEN_k \cup CLOSED_k \subseteq OPEN_{k+1} \cup CLOSED_{k+1} \forall k . \qquad (1)$$

4- si $n' \in succ(n)$ et $n \in CLOSED_k$, alors $n' \in OPEN_k \cup CLOSED_k$.

5- si $n \in N(G)$, pour tout chemin $P \in \mathbf{P}(s,n)$ et $k \geq 0$, soit $N(P) \subseteq CLOSED_k$ ou $N(P) \cap OPEN_k \neq \Phi$.

Cette relation implique que pour un chemin donné de s (départ) au nœud n, les nœuds constituant ce chemin soit, ils sont tous dans CLOSED (ils sont déjà parcourus), sinon, il en existe au moins un qui est dans OPEN (non parcouru ou il est sélectionné pour régénération).

4.3.2 Organigramme de l'algorithme MOA*

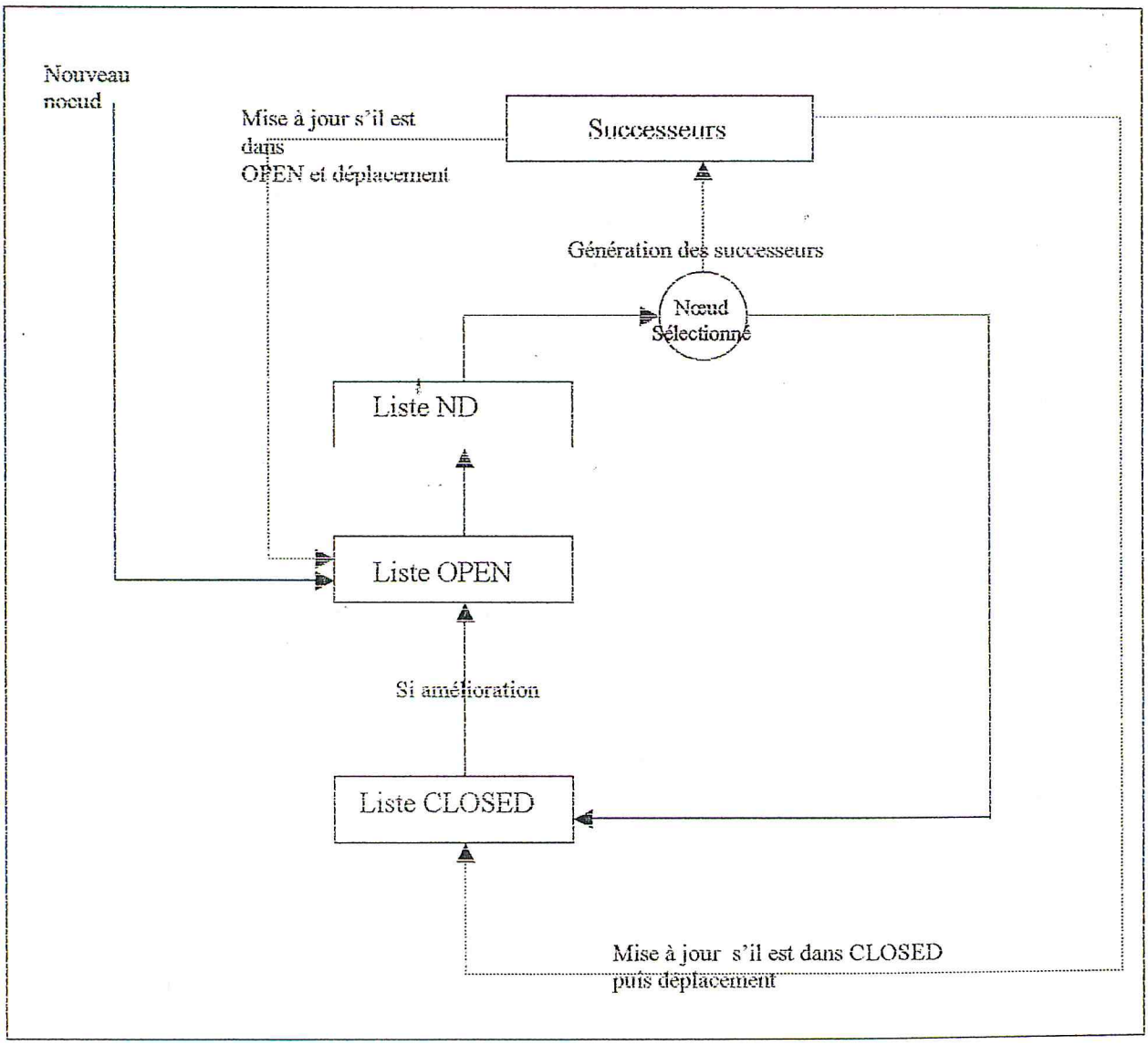


Figure 6 : organigramme de l'algorithme MOA*

4.3.3 Enoncé et principe de l'algorithme MOA*

```
Début
(1)  OPEN ← S (nœud de départ)
(2)  TANT QUE (OPEN n'est pas vide) FAIRE
(3)      Nb-itération ++ ;
(4)      ND ← ensemble des nœuds non-dominés de OPEN ;
(5)      SI (ND n'est pas vide) ALORS
             Nœud ← un élément de ND selon une heuristique ;
             Mettre nœud dans CLOSED ;
(6)          POUR tout les successeurs de nœud FAIRE
(7)              SI (successeur n'est pas un nœud d'arrivée) FAIRE
                     SI (successeur est nouvellement généré)
(8)                      OPEN ← nouveau élément avec mise à jour de f dans LABEL ;
(9)                  ELSE
(10)                     SI (successeur est dans CLOSED) ALORS
                             S'il y a amélioration, m-à-j dans LABEL et le mettre dans OPEN ;
                         FINSI
(11)                     SI (successeur est dans OPEN)
                             S'il y a amélioration, mettre à jour dans OPEN et LABEL;
                         FINSI
                     FINSI
(12)             SINON
(13)                 SI (successeur est dans SOLUTION-GOALS) ALORS
                         S'il y a amélioration, mettre à jour dans SOLUTION-GOALS et LABEL;
(14)                 SINON
                         Ajouter le nœud dans SOLUTION-GOALS ;
                     FINSI
                 FINSI
             FINPOUR
(15)     SINON fin de l'algorithme
     FINTANTQUE
FIN.
```

Figure 7 : Enoncé d'algorithme MOA*.

Comme dans l'algorithme A* classique, l'algorithme MOA* commence par mettre le nœud de départ dans la liste OPEN pour être développé en premier (étape 1). A chaque exécution de l'étape 2, le compteur d'itérations est incrémenté puisqu'elle est la boucle principale de l'algorithme : Elle consiste à sélectionner les nœuds à développer (nœuds dont les valeurs sont non dominées) et les mettre dans ND (liste des nœuds non-dominés) : étape 4. Un nœud est sélectionné à partir de ND selon une heuristique appropriée (voir chapitre suivant). Ce nœud fera l'objet de traitement dans la suite de l'algorithme après avoir mis dans la liste CLOSED (puisqu'il sera traité). On génère ses successeurs, et pour chacun d'eux (étape 6), s'il ne s'agit pas d'un nœud d'arrivée, on fera le traitement suivant (étape 7) :

- s'il est nouvellement généré, il sera mis dans OPEN (puisqu'il est nouveau), et on met à jour les pointeurs de retour dans LABEL (étape 8). LABEL est conçue pour garder trace des chemins.
- Sinon, et s'il est dans CLOSED et il apporte une amélioration (étape 10), il sera remis dans OPEN dans l'espoir d'être régénéré (avec mise à jour de son amélioration).
- On fait un autre test sur son existence dans OPEN (étape 11), si c'est le cas, sa valeur sera mise à jour dans OPEN et LABEL.

Si le successeur en question est un nœud d'arrivée (étape 12), on teste s'il s'agit d'un nœud final d'un chemin 'solution' (ceci se fait par un simple test d'existence dans SOLUTION-GOALS qui contient les nœuds d'arrivée 'but').

- Si c'est le cas, on met à jour dans LABEL et SOLUTION-GOALS son amélioration s'il en a apporté.
- Sinon (étape 14), on le met simplement dans SOLUTION-GOALS (c'est un nouveau chemin optimal).

Le programme s'arrête quand la liste ND est vide (étape 15)

4.3.4 Application sur exemple

L'exemple suivant de problème du plus court chemin illustre le fonctionnement de l'algorithme MOA* sur un graphe d'état simple. Le but de cet exemple est de présenter une vue générale du comportement de l'algorithme et de montrer étape par étape le processus de résolution. Nous tenons à préciser que la sélection d'un nœud pour développement à partir de la liste ND se fait au hasard sur cet exemple.

Le graphe d'état présenté dans la figure 6. résume un certain problème à résoudre avec un nombre d'objectifs égal à 2 (M=2). Les valeurs (coûts) des arcs sont représentées par des couples de valeurs positives. Le nœud de départ est le nœud 0, et les nœuds encadrés par des pointillés représentent les nœuds d'arrivée (ensemble Γ déjà défini).

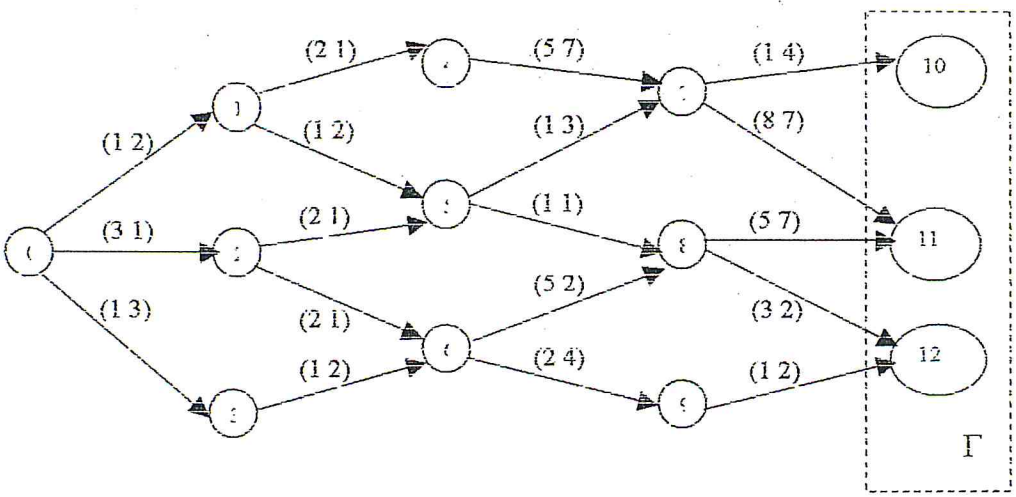


Figure 8 : Exemple de graphe d'état

A partir des informations de la figure ci-dessus, nous devons trouver le(s) plus court(s) chemin(s) du nœud 0 aux trois nœuds 10, 11, 12 et garder leurs traces (listes des nœuds de 0 au(x) dernier(s)). Mais tout d'abord, on doit définir une heuristique qui estime le chemin restant à chaque nœud. Nous allons simplement donner des valeurs à cette fonction pour chaque nœud pour simplifier le déroulement de l'algorithme. Ces valeurs sont représentées par le symbole H* dans le tableau 1.

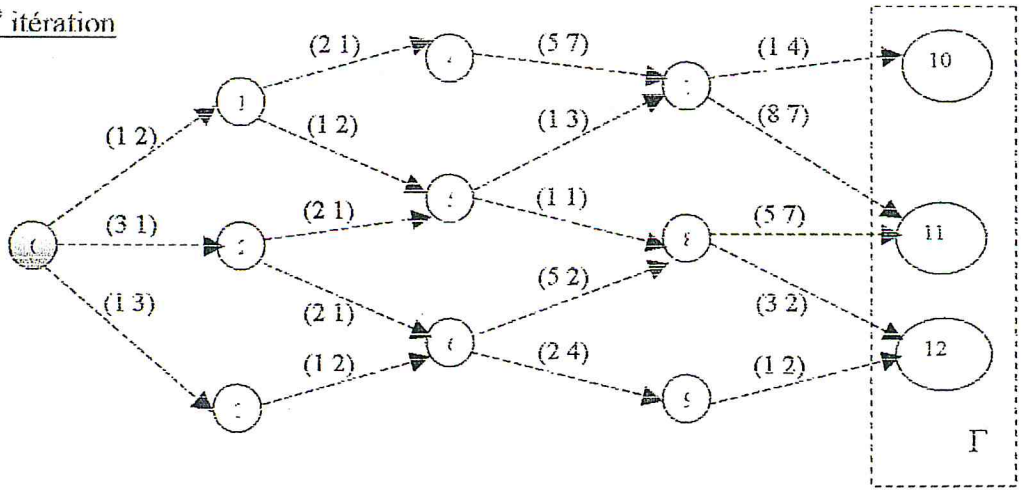
noeud	H*(n)
0	(4 11) (6 7) (9 5)
1	(3 9) (6 5)
2	(4 8) (6 4) (5 7)
3	(9 6) (4 8)
4	(6 11)
5	(2 7) (4 3)
6	(8 4) (3 6)
7	(1 4)
8	(3 2)
9	(1 2)
10	(0 0)
11	(0 0)
12	(0 0)

Tableau 1 : valeurs de l'heuristique

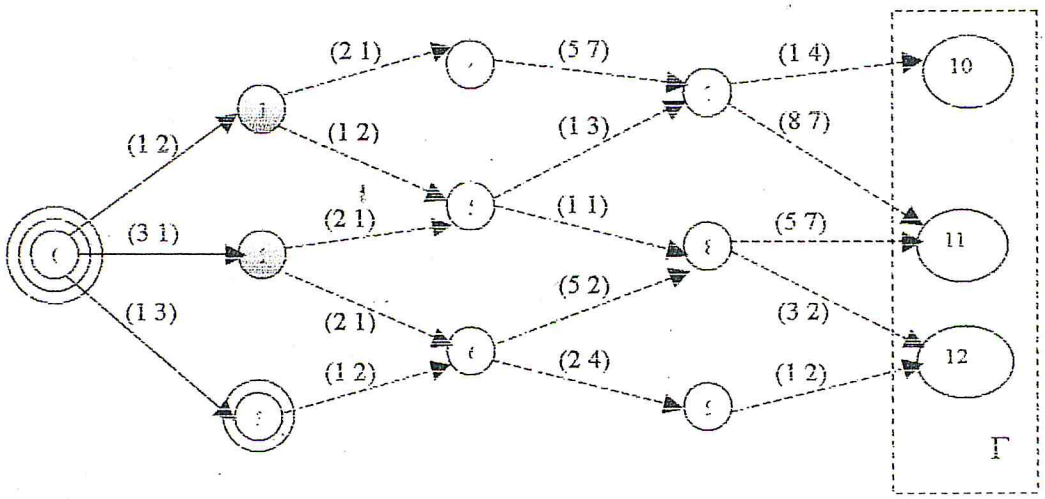
Le déroulement de l'algorithme est suivi au bout de 11 itérations. Chaque itération est représentée par un graphe (figure 4.). La trace de l'algorithme est représentée dans le tableau 2. Chaque graphe représente l'état de l'algorithme à une itération. Les nœuds du graphe initial sont représentés par de simples cercles. Les nœuds de OPEN sont représentés par des doubles cercles. Les nœuds non-dominés de OPEN, qui sont ceux de ND, sont représentés par des cercles à fond ombré. Les nœuds sélectionnés pour développement sont représentés par 3

cercles. Et enfin les arcs des chemins « solution » sont dotés d'arcs supplémentaires en pointillés.

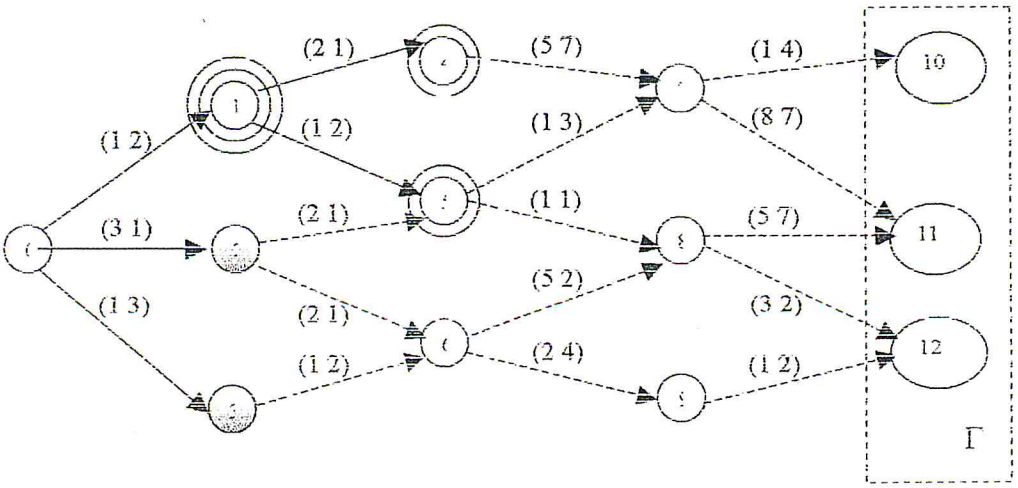
1ere itération



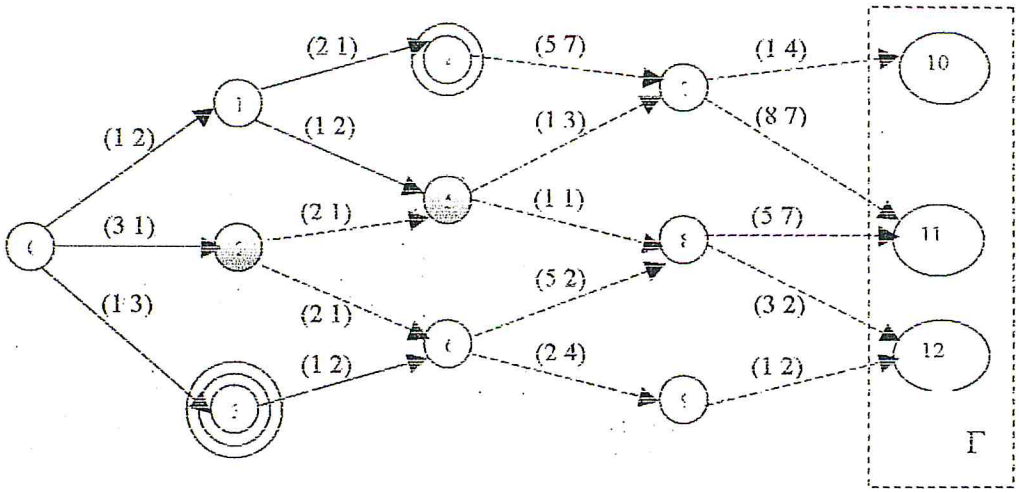
2eme itération



3eme itération



4ième itération



Le processus continue jusqu'à la 11ième itération, et on aura le graphe suivant :

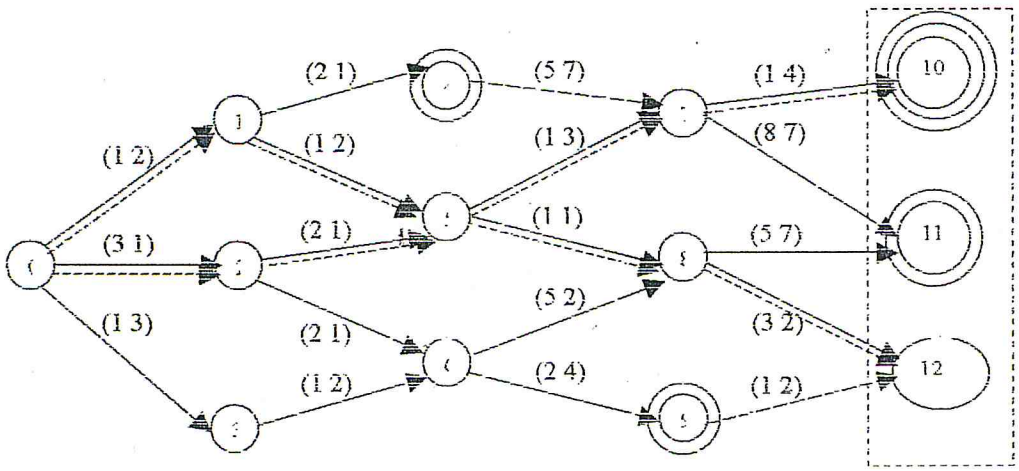


Figure 9 : déroulement de l'algorithme MOA*

L'algorithme s'est terminé sur le dernier graphe du schéma ci-dessus. On voit clairement que la solution trouvée est présentée par les trois chemins aux arcs en pointillés : {0, 1, 5, 8, 12}, {0, 2, 5, 8, 12}, {0, 1, 5, 7, 10}.

Les coûts de ces chemins sont obtenus de la trace de l'algorithme présentée dans le tableau 2.

- le chemin 1 est constitué des nœuds 0, 1, 5, 8, 12. son coût est (6 7).
- le chemin 2 est constitué des nœuds 0, 2, 5, 8, 12. son coût est (9 5).
- le chemin 3 est constitué des nœuds 0, 1, 5, 7, 10. son coût est (4 11).

On voit que les valeurs des coûts des trois chemins sont non dominées entre elles.

$OPEN_k$				Ajustement des points de retour			
k	n	Nouveau $G_k(n)$	Nouveau $F_k(n)$	n'	$LABEL_k(n',n)$	CLOSED	$SOLUTION-COSTS_k$ et $GOALS_k$
0	0**	(0 0)	(1 2) (3 1)			Φ	Φ
1	1** 2* 3	(1 2) (3 1) (1 3)	(2 4) (3 3) (5 2) (2 5)	0 0 0	(1 2) (3 1) (1 3)	0	Φ
2	2*,3** 4 5	 (3 3) (2 4)	 (8 10) (3 5)	 1 1	 (3 3) (2 4)	0,1	Φ
3	2**,4,5* 6	 (2 5)	 (7 7) (4 9)	 3	 (2 5)	0,1,3	Φ
4	4 5** 6	 (2 4) (5 2) (2 5) (5 2)	 (3 5) (6 3) (4 9) (7 6) (10 4)	 2 2	 (5 2) (5 2)	0,1,2,3	Φ
5	4,6** 7 8*	 (3 7) (6 5) (3 5) (6 3)	 (4 11) (7 9) (6 7) (9 5)	 5 5	 (3 7) (6 5) (3 5) (6 3)	0,1,3,2,5	Φ
6	4,7*,8** 9	 (7 6) (4 9)	 (8 8) (5 11)	 6	 (7 6) (4 9)	0,1,3,2,5,6	Φ
7	4,7*,9 11 12**	 (8 12) (11 10) (6 7) (9 5)	 (8 12) (11 10) (6 7) (9 5)	 8 8	 (8 12) (11 10) (6 7) (9 5)	0,1,3,2,5,6	Φ
8	4,7**,9,11					0,1,3,2,5,6,8,12	{(6 7), (9 5)} ; {12}
9	4,9,11 10**	 (4 11) (7 9)	 (4 11) (7 9)	 7	 (4 11) (7 9)	0,1,3,2,5,6,8,12,7	{(6 7) (9 5)} ; {12}
10	4,9 11					0,1,3,2,5,6,8,12,7,10	

Tableau 2 : trace de l'algorithme MOA* sur l'exemple précédent.

5. Conclusion :

Pour le moment, nous avons introduit l'aspect général de l'algorithme A* multi-objectif : son principe, son fonctionnement, ainsi que ses particularités par rapport à l'algorithme A* classique (mono-objectif). L'algorithme MOA*, tel qu'il est, peut présenter quelques points susceptibles d'être affinés, à savoir l'optimisation en temps, en espace,...etc. cela fera appel à des notions et techniques plus compliquées, à savoir la méthode TABOU et les outils d'aide à la décision, le choix des meilleurs heuristiques,...etc. Cette recherche fera l'objet du prochain chapitre, à savoir l'introduction d'une liste tabou pour interdire de refaire le parcours d'une solution, et le choix de l'heuristique de sélection des nœuds à partir de la liste ND, et dont les performances varient selon la nature du graphe d'état initial (type du problème).

Chapitre 4

Contributions

1. Première contribution : Heuristiques pour l'amélioration de l'algorithme MOA*

L'algorithme MOA* tel qu'il est présenté par BRADLEY S. STEWART et CHELSEA C. WHITE, donne à l'utilisateur la liberté de choisir une heuristique de sélection d'un nœud à partir de la liste ND, qui jugera intéressante. A première vue, l'heuristique la plus simple est de choisir un nœud au hasard à partir de ND (cas du MOA*), et faire le traitement nécessaire sur ce nœud. Effectivement, à première vue, cette idée parait bonne, ou à la rigueur « pas très mauvaise ». Mais, dès qu'on rentre dans les détails de l'algorithme MOA*, on s'aperçoit que lorsqu'on met un nœud *n* dans OPEN avant un autre nœud *n'*, et qu'ils rentrent tout les deux dans ND, ça se pourrait que le nœud *n'* soit développé avant *n* d'après le choix de la machine (au hasard) : chose qui peut ralentir le traitement du nœud *n* qui peut être le nœud qui mène au chemin optimal. Comme ça pourra être le cas pour l'inverse. Ce choix donc peut éloigner le nombre d'itération de l'ordre des espérances de l'utilisateur, puisque ce choix n'est fondé sur aucune heuristique. Chose qui est prouvée par des exemples lors de sa comparaison avec d'autres heuristiques après les avoir implémentées. Notons que la mise en œuvre de ces heuristique dans l'algorithme MOA* se fait au niveau de la sélection du nœud de la file ND (étape 4 de l'énoncé du MOA*).

1.1 Première heuristique : ordre d'arrivée de la liste OPEN

1.1.1 Idée générale

La deuxième idée développée est d'ordonner les nœuds de ND selon leur ordre d'arrivée de la liste OPEN. Si un nœud est non-dominé et il est arrivé le premier à ND, il sera le premier à être développé. Cette idée remédie au problème cité dans le paragraphe précédant puisqu'elle garde l'ordre d'arrivée des nœuds non-dominés à OPEN.

1.1.2 Mise en œuvre

L'idée citée ci-dessus est mise en œuvre en transformant la liste ND en une file d'attente (FIFO). Le nœud qui rentre en premier dans ND sera le premier à être sélectionné si sa valeur est non-dominée. Sinon (si sa valeur est dominée), l'heuristique ne s'applique pas sur lui.

1.1.3 Avantage

L'avantage de cette approche est qu'elle assure un parcours en largeur des nœuds non-dominés et interdit tout chevauchement. La fiabilité de cette idée ne peut être prouvée est généralisée, mais expérimentalement, elle donne souvent des résultats meilleurs que le choix au hasard. Cela n'exclue pas le fait que le choix aléatoire peut apporter un meilleur résultat que celui de l'heuristique citée, mais ceci reste du pur hasard.

1.1.4 Test sur exemple :

Soit le graphe suivant :

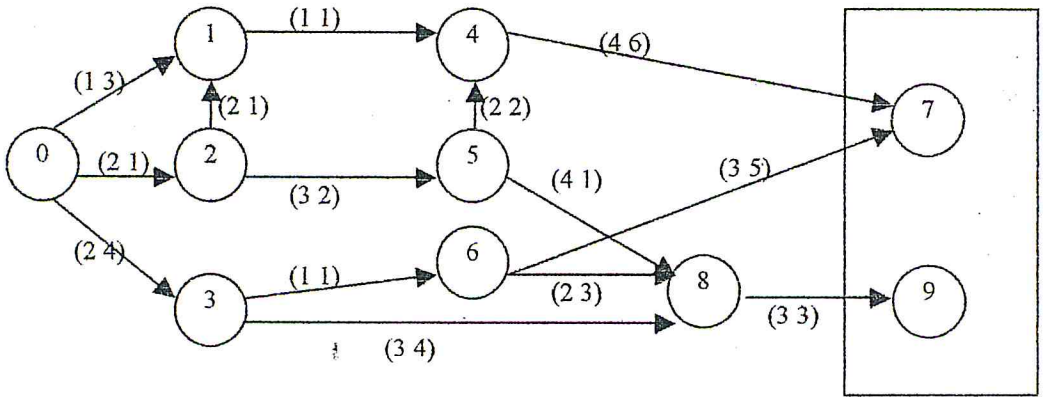


Figure 10 : test sur la première heuristique

Résultats obtenus :

Nombre de chemins optimaux : 4

Chemin 1 : {0, 1, 4, 7} ; coût = (6 10) ;

Chemin 2 : {0, 3, 6, 7} ; coût = (6 10) ;

Chemin 3 : {0, 2, 1, 4, 7} ; coût = (9 9) ;

Chemin 4 : {0, 2, 5, 8, 9} ; coût = (12 7) ;

Nombre d'itérations = 9

1.2 Deuxième heuristique : choix du nœud avec moins de prédécesseurs (pères)

1.2.1 Idée générale

Dans l'algorithme MOA*, la raison pour laquelle le nombre d'itérations augmente est l'augmentation des déplacements des nœuds de CLOSED vers OPEN car à chaque déplacement vers OPEN, le nœud sera régénéré : ce qui provoque une itération de plus à chaque déplacement. L'idée est de commencer le développement par les nœuds avec moins de prédécesseurs : par exemple un nœud avec un seul père ne sera régénéré que si son père est régénéré (il est moins probable qu'il soit régénéré) : donc on commence par le développement de ce nœuds dans l'espoir de rencontrer les nœud avec plus de pères et les mettre à jour seulement au lieu de le développer à plusieurs reprises. De même, qu'un nœud avec deux (2) pères est préférable qu'un nœud avec trois (3) pères.

1.2.2 Mise en œuvre :

La mise en œuvre de cette idée se fait en introduisant une information supplémentaire sur les nœuds du graphe. Et le choix du nœud approprié se fait en sélectionnant le minimum de ces informations.

1.2.3 Avantage :

Comme cité dans l'idée de cette heuristique, l'avantage est de minimiser les déplacements des nœuds de la liste CLOSED vers la liste OPEN.

Cela n'empêche pas qu'elle présente un inconvénient qui est la non optimalité : elle est observée dans certains types de graphes rares où les nœuds ont presque tous le même nombre de pères. Ceci est montré dans l'exemple 4 de la comparaison des algorithmes.

1.2.4 Test sur exemple :

On teste le même exemple de la première itération. Les résultats obtenus sont :

- les mêmes chemins et les mêmes coûts
- Nombre d'itérations = 8

Résultats obtenus avec l'algorithme MOA* avec sélection aléatoire :

Nombre d'itérations = 9

1.3 Conclusion :

On voit clairement, à travers cet exemple, que le nombre d'itérations dans l'algorithme MOA* avec heuristique de moins de pères est inférieur à celui MOA* avec sélection aléatoire et également par rapport au MOA* avec ordre d'arrivée des nœuds. L'égalité des nombres d'itérations dans ces deux dernières versions n'implique pas la généralisation de leur égalité de performance, cette généralisation peut se faire sur des types de problèmes donnés : on verra dans la section de comparaison de ces heuristiques sur divers problèmes que, dans un type de problèmes possédant certaines propriétés, l'heuristique d'ordre d'arrivée vers ND apporte des améliorations en nombre d'itérations par rapport à la sélection aléatoire, et même par rapport à l'heuristique de moins de pères dans un autre type de problèmes.

2. Seconde contribution : Méta heuristique pour l'amélioration de l'algorithme MOA*

2.1 Introduction

Comme présentées dans le premier chapitre, les heuristiques sont des techniques utilisées pour approcher la solution désirée le maximum possible en un temps raisonnable, ou encore pour affiner la solution trouvée en terme d'un critère voulu.

Une heuristique peut être conçue pour résoudre un type de problème donné, ou bien être conçue comme une méthode générale, qui peut être adaptée à divers problèmes d'optimisation: dans le second cas, elle est désignée sous le terme « *méta heuristiques* ». [MEM1]

Les diversions des problèmes d'optimisation élargissent l'utilisation des heuristiques et des métaheuristiques et créent des variantes de ces techniques à chaque nouvelle forme de problème. Dans notre cas, l'algorithme MOA* nous donne des solutions appartenant à l'ensemble 'Pareto Optimal', ce qui veut dire qu'aucune amélioration ne pourra être apportée à ces solutions. Par contre, on pourra jouer sur le nombre d'itérations de l'algorithme (inévitablement le temps d'exécution) grâce à ces méthodes. A partir de l'idée « ne pas revisiter les solutions déjà générées dans un graphe », la méthode Tabou, présentée ici comme métaheuristique, procède à l'interdiction de repasser par un nœud ou des nœuds susceptibles de ralentir le processus de recherche d'une solution. A l'interdiction, ce nœud est remplacé par un de ses voisins pour continuer la recherche. Mais cela n'est pas toujours évident, car dans certains problèmes de petite taille ou de petite complexité, l'adoption de cette méthode (Tabou) peut ralentir le processus de résolution. Cela est dû à la présence de la liste tabou qui demande des itérations de plus pour l'insertion ou le déplacement des éléments (nœuds). Des exemples seront présentés plus tard pour « approximer » la classe des problèmes où l'algorithme MOA* avec recherche tabou donne ses performances. Avant de rentrer dans les détails de cette méthode, nous commençons par la définition de la classe des méthodes de voisinage à qui la méthode Tabou appartient.

2.2 Méthodes de voisinage

Les méthodes de voisinage sont basées sur la notion de voisinage, dans ce qui suit nous allons introduire quelques définitions concernant le principe de ces méthodes :

➢ **Définition** :

Soit X un ensemble de configurations admissibles (états possibles de l'espace de recherche) d'un problème, on appelle voisinage l'application $N : X \rightarrow 2^X$

Le voisinage d'une configuration peut être défini de plusieurs manières, cela dépend du problème et du raisonnement du concepteur. On pourra définir comme voisinage :

- l'ensemble des configurations dont la distance mathématique entre la configuration en question et ces configurations est minimale (la distance est en terme de coût).

  Exemple de distance mathématique : *distance de Tchebycheff, distance de Hamming* [B&H].
- L'ensemble des nœuds successeurs du dernier nœud d'une configuration.

Le voisinage pourra être choisi d'une autre manière en fonction des données du problème.

➢ **Principe des méthodes de voisinage :**

- Une méthode de voisinage commence par une configuration initiale, elle réalise en suite un processus itératif qui consiste à remplacer la configuration courante par une voisine, en tenant compte de la fonction coût. Ce processus s'arrête et retourne la meilleure configuration trouvée lorsque la condition d'arrêt est réalisée ; celle-ci concerne généralement une limite pour un nombre d'itérations ou un objectif à réaliser.
- Les méthodes de voisinage différent essentiellement entre elles par le voisinage utilisé, ainsi que la stratégie de parcours de celle-ci.
- L'avantage des méthodes de voisinage réside dans la possibilité de contrôler le temps de calcul : la qualité de la solution trouvée tend à s'améliorer progressivement au cours du temps, l'utilisateur est donc libre d'arrêter l'exécution au moment qu'il jugera propice.
- Un voisinage peut être représenté à l'aide d'un graphe orienté : voir la figure (6).

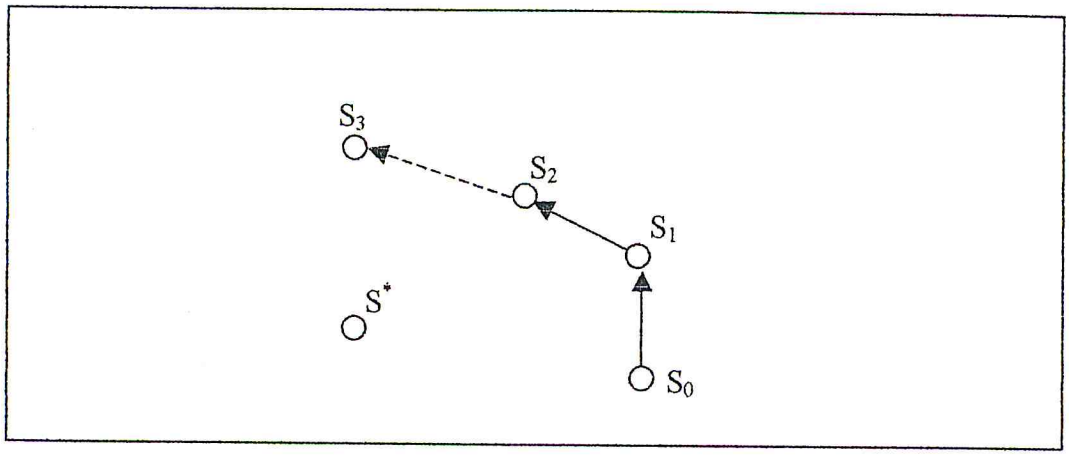


Figure 11 : notion de voisinage

2.3 Méthode de recherche Tabou

Cette méthode a été présentée pour la première fois par F. Glover (« Future Paths for Integer Programming and Links to Artificial Intelligence », 1986).

L'idée de base consiste à introduire la notion d'histoire dans la politique d'exploration des solutions. Le nom « tabou » donné en 1986 par Glover exprime l'interdiction de reprendre des solutions récemment visitées.

Un mouvement permet de passer d'une solution (configuration) valide à une autre, on dit que la nouvelle solution est un voisin de la précédente. L'ensemble des solutions que l'on peut atteindre à partir d'une solution est le voisinage.

A chaque itération, tous les mouvements possibles sont examinés et le « meilleur », ou plus exactement le moins mauvais est sélectionné, et la résolution continue à partir de cette configuration.

Comme cette manière peut cycler c'est-à-dire répéter indéfiniment la même suite de mouvements, les t derniers mouvements effectués sont considérés comme interdits (« tabous »), t représentant la taille de la liste tabou. Le mouvement effectivement choisi à chaque itération, est donc le meilleur mouvement non tabou.

Algorithme :

Initialisation :

1- trouver une solution initiale x_0. $x^* \leftarrow x_0, c^* \leftarrow f(x_0)$. x^* est la meilleure solution rencontrée, c^* son coût et f la fonction d'évaluation.

2- O←k, ListeTabou = ϕ

Répéter tant qu'un critère de fin n'est pas vérifié :

1- choisir parmi le voisinage de $x_k, V(x_k)$, le mouvement qui minimise f et qui n'appartient pas à ListeTabou, *meilleur*(x_k).

2- $x_{k+1} \leftarrow$ *meilleur*(x_k)

3- si (c(x_{k+1})<c^*) alors $x^* \leftarrow x_{k+1}, c^* \leftarrow c(x_{k+1})$.

4- Mise à jour de ListeTabou.

[VAN.02]

2.4 Application au MOA* :

Dans l'algorithme MOA*, les nœuds à développer sont mis dans la file ND (ensemble des nœuds dont les valeurs de fonctions sont non-dominées). Quand on est en possession d'un nœud à développer, ce nœud peut contenir plusieurs valeurs de fonctions. Ce qui implique qu'un nœud ne suffit pas pour former une configuration. La configuration doit avoir une valeur de fonction unique. La configuration dans notre cas est définie comme suit :

Définition 1 :

Dans un graphe de recherche, une configuration est formée d'un nœud muni d'une seule de ses valeurs de fonction de coût

Cette définition induit que si un nœud possède n valeurs de fonction, ce nœud possède n configurations (à valeurs différentes).

Voisinage : le voisinage d'un nœud est formé de l'ensemble de ses nœuds successeurs, car le passage n'est permis que s'il se fait du père au fils. Cette condition impose un choix du nœud suivant parmi les successeurs, et qui, dans notre cas, est pris en charge par l'algorithme MOA* lui-même (cela se fait au niveau de l'heuristique de choix du nœud à extraire). Le

nœud voisin peut représenter plusieurs configurations, ce qui donne un ensemble de configurations voisines

Définition 2 : *un nœud n' est un voisin du nœud n si et seulement si n' est un successeur de n.* ($n' \in succ(n)$).

2.5. Principe de l'algorithme MOA* avec recherche TABOU

La méthode TABOU est intégrée dans l'algorithme MOA* non pas sous forme d'un module connexe à insérer, mais sous forme de tests ou d'affectations aux endroits idéaux. Cela est dû à la multitude des ensembles (files) utilisées par cet algorithme, et les traitements appropriés. La méthode TABOU impose à l'algorithme MOA* de ne visiter un nœud qu'une seule fois. Ceci se fait au niveau de l'extraction de ce nœud de la file ND. A chaque fois que l'algorithme arrive à cette phase, le traitement TABOU intervient pour mettre le nœud dans la liste tabou (voir la figure 7) pour interdire le passage par ce nœud aux prochaines itérations. A la sortie de la liste TABOU, si le nœud apporte une amélioration en matière de coût du chemin, il sera remis dans la file OPEN (file des nœuds ouverts) et il aura la possibilité d'être régénéré. Et s'il n'apporte pas d'amélioration, il sera mis dans la liste CLOSED (liste des nœuds fermés).

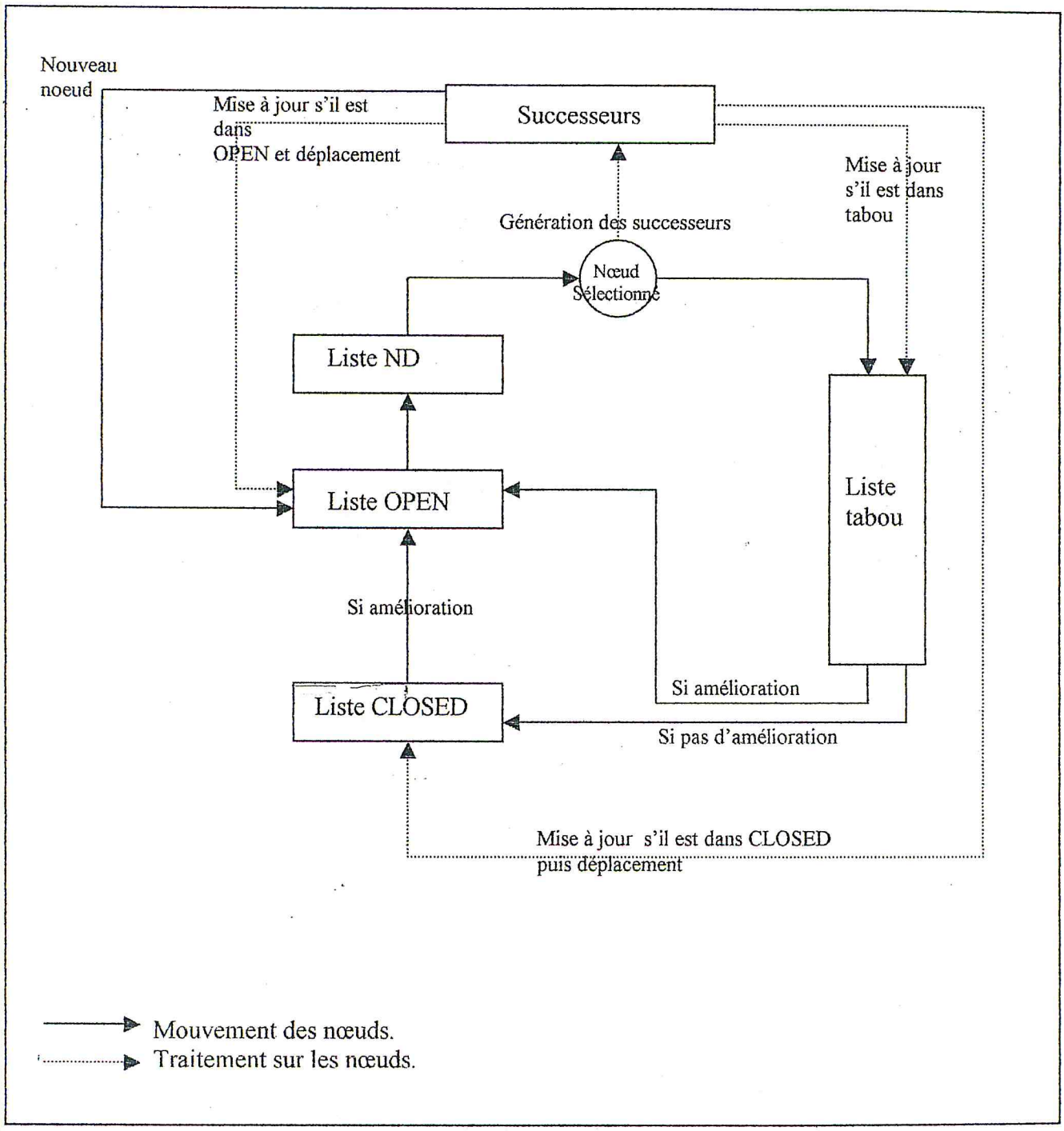


Figure 12 : organigramme de l'algorithme MOA* avec recherche TABOU

2.6 Enoncé de l'algorithme

```
DEBUT
    S ← nœud de départ ;
    OPEN ← S ;
    TANT QUE (OPEN n'est pas vide) FAIRE
        Nb-itération ++ ;
        ND ← ensemble des nœuds non-dominés de OPEN ;
        SI (ND n'est pas vide) ALORS
            Nœud ← un élément aléatoire de ND ;
            Déplacer sommet de OPEN vers TABOU ;
            POUR tout les successeurs de nœud FAIRE
                SI (successeur n'est pas un nœud d'arrivée) ALORS
                    SI (successeur est nouvellement généré) ALORS
                        OPEN ← nouvel élément avec f = g+h ;

                    SINON
                        SI (successeur est dans TABOU) ALORS
                            S'il y a amélioration au niveau du coût (nouvelle valeur domine au moins
                            un élément), mettre à jour les valeurs de ce noeud ;
                        FINSI

                        SI (successeur est dans CLOSED) ALORS
                            S'il y a amélioration, mettre à jour les valeurs du nœud et le déplacer vers
                            OPEN ;
                        FINSI

                        SI (successeur est dans OPEN) ALORS
                            Mettre à jour s'il y a amélioration ;
                        FINSI
                    FINSI
                SINON
    1 2 3 4 5
```

```
1 2 3 4 5   SI successeur est dans SOLUTION-GOALS  ALORS
                Mise à jour s'il y a amélioration ;
            SINON
                Ajouter le nœud dans la liste dans SOLUTION-GOALS ;
            FINSI
        FINSI
    FINPOUR
SINON
        Défiler un élément de la liste TABOU si la taille = 6
        Mettre le nœud dans OPEN s'il y a une amélioration ;
        SINON le mettre dans CLOSED ;
FINSI
FINTANTQUE

FIN
```

Figure 13 : Enoncé de l'algorithme MOA* avec recherche TABOU

2.6.1 Taille de la liste TABOU :

La liste TABOU est une liste de taille fixe dont la taille est fixée par l'utilisateur ; mais la plupart des implémentations de la liste TABOU se fait avec une liste de taille entre 6 et 8. La plupart des travaux faits actuellement dans ce domaine utilisent des listes de taille 7 ou 8. Dans notre travail, nous avons adopté le principe d'expérimentation pour déterminer la taille de cette liste (les meilleurs résultats sont ceux avec une taille de 6).

2.6.2 Gestion de la liste TABOU :

La liste TABOU est sous forme de table qui contient les nœuds munis des informations sur ces nœuds et qui sont :

- le numéro du nœud dans le graphe
- le vecteur de coût du nœud
- l'ensemble des nœuds prédécesseurs
- l'ensemble des nœuds successeurs.

2.6.3 Remarques à propos de la liste TABOU

- le principe de la liste TABOU est FIFO (premier arrivé premier sorti).
- un élément existant dans cette liste ne peut pas être regénéré pendant la recherche jusqu'à ce qu'il sorte de la liste.
- à chaque fois qu'un nœud existant dans la liste TABOU est rencontré pendant la recherche, il doit être mis à jour si sa nouvelle valeur est meilleure que l'ancienne valeur.
- A chaque itération de l'algorithme, la liste TABOU peut être vide ou pleine ou demi pleine.

3 Comparaison des performances des 4 algorithmes sur des exemples :

dans cette partie nous allons montrer les performances obtenues pour chacun des quatre algorithmes à travers quatre exemples. Le choix de ces exemples est fait de telle sorte que l'efficacité de chaque algorithme soit montré sur un type spécifique de graphes. C'est à dire qu'un algorithme n'est optimal que sur un type spécifique de graphes.

Exemple 1 : soit l'exemple de graphe suivant :

Nœud de départ : 0

Nœud d'arrivée (ensemble Γ) : 6

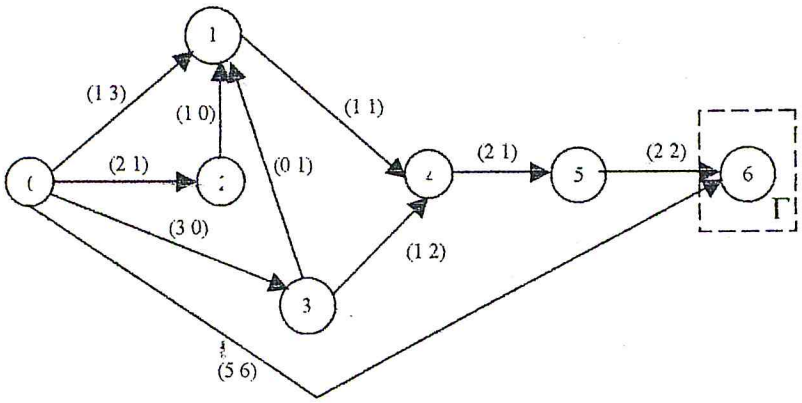


Figure 14 : graphe de l'exemple 1.

Résultats obtenus :

Le nombre de chemins optimaux : 4

Chemin 1 : {0, 2, 1, 4, 5, 6} ; coût = (8 5) ;

Chemin 2 : {0, 3, 4, 5, 6} ; coût = (8 5) ;

Chemin 3 : {0, 3, 1, 4, 5, 6 } ; coût = (8 5) ;

Chemin 3 : {0, 6} ; coût = (5 6) ;

Nombre d'itérations : - MOA* : 7 itérations.

- MOA* avec recherche TABOU : 7 itérations.
- MOA* avec heuristique d'ordre d'arrivée de OPEN : 7 itérations
- MOA* avec heuristique de moins de pères : 6 itérations

Tableau récapitulatif des résultats obtenus (nombre d'itérations) :

	MOA*	MOA* avec recherche TABOU	MOA* avec ordre d'arrivée de OPEN	MOA* avec moins de pères
Nombre d'itérations	7	7	7	6

Tableau 3 : résultats de l'exemple 1

Interprétation des résultats :

On remarque dans le tableau que l'heuristique de moins de pères est la plus optimale en temps d'exécution (nombre d'itérations de l'algorithme). Quand aux deux autres heuristiques (sélection aléatoire et selon l'arrivée de OPEN), elles enregistrent le même nombre d'itérations avec une de plus par rapport à l'heuristique de moins de pères. L'algorithme MOA* avec métaheuristique (recherche TABOU) apporte le même nombre d'itérations que celles-ci. L'optimalité de l'heuristique avec moins de pères dans cet exemple peut être expliquée par le fait que dans la deuxième itération de l'algorithme, les nœuds sélectionnés de ND pour développement en premier sont 2 et 3 et ils mènent à des solutions non-dominées, ceci est du au fait que chacun d'eux a 1 seul père. Contrairement au nœud 1 qui a 3 pères.

Exemple 2 :

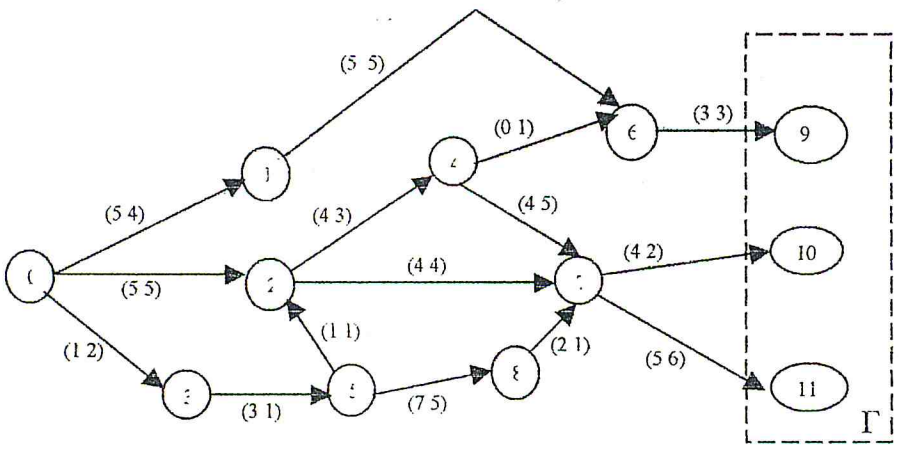


Figure 15 : graphe de l'exemple 2

Résultats :

Nombre de chemins optimaux : 2

Chemin 1 : {0, 3, 5, 2, 4, 6, 9} ; coût = (12 11)

Chemin 2 : {0, 3, 5, 2, 7, 10} ; coût = (13 10)

Nombre d'itérations : - MOA* : 10 itérations.
- MOA* avec recherche TABOU : 15 itérations.
- MOA* avec heuristique d'ordre d'arrivée de OPEN : 10 itérations
- MOA* avec heuristique du moindre de pères : 10 itérations.

Tableau récapitulatif des résultats obtenus (nombre d'itérations) :

algorithmes	MOA*	MOA* avec recherche TABOU	MOA* avec ordre d'arrivée de OPEN	MOA* avec moins de pères
Nombre d'itérations	10	15	10	10

Tableau 4 : résultats de l'exemple 2

Interprétation des résultats :

Dans cet exemple les trois heuristiques enregistrent le même nombre d'itérations. Ceci peut s'expliquer par le fait que les deux chemins optimaux ont un sous chemin commun durant les quatre premières itérations de leur exécution et que durant les autres itérations, les nœuds restants sont des nœuds appartenant aux deux solutions. Ce qui donne que le choix de l'heuristique ne change rien au déroulement de l'algorithme. Pour la métaheuristique (recherche TABOU), le graphe n'est lui pas favorable car il n'y a pas de revisite des nœuds. Ce qui fait qu'elle fait des itérations de plus lors du déplacement des nœuds vers la liste tabou (5 itérations de plus).

Exemple 3 :

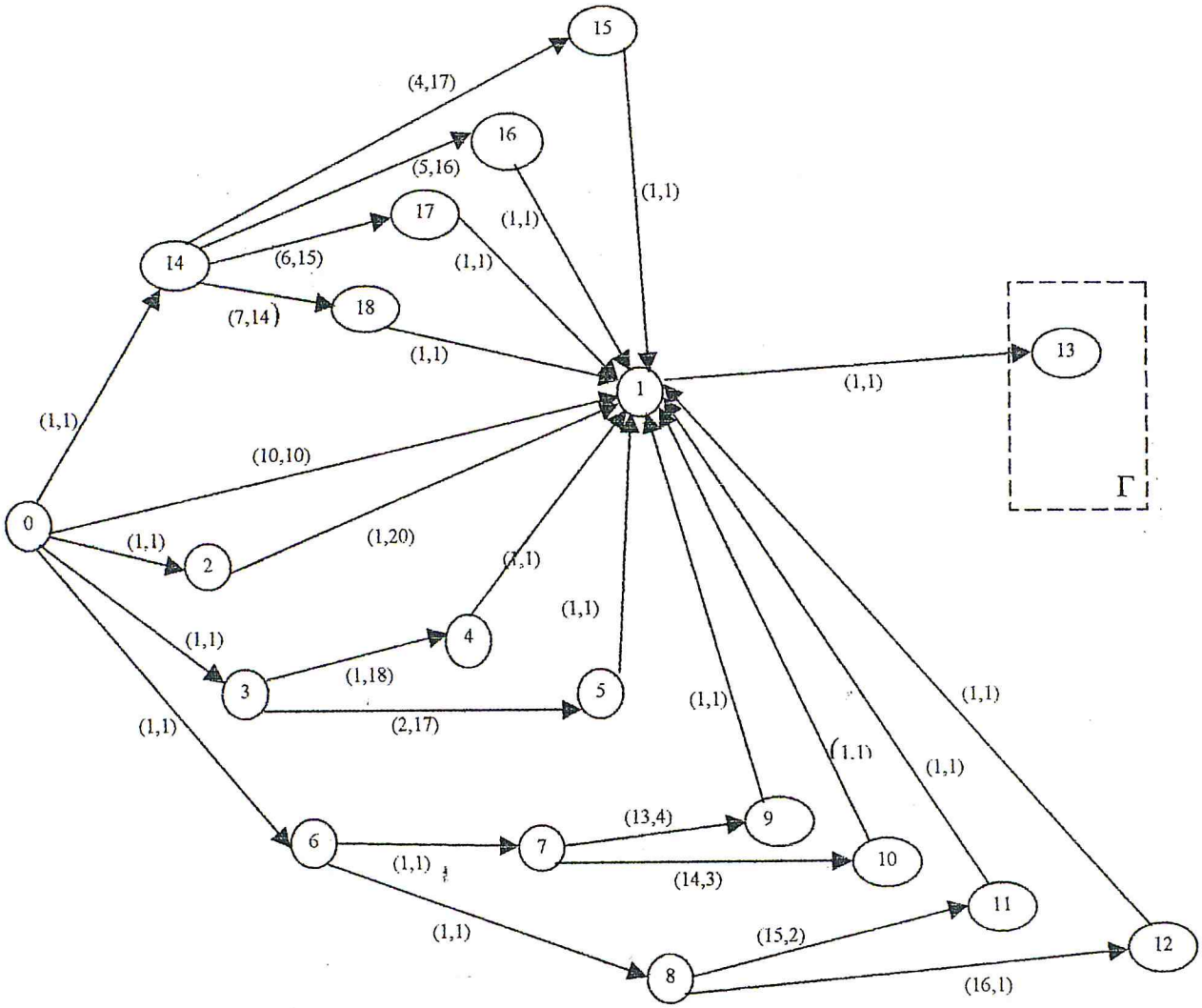


Figure 16 : graphe de l'exemple 3.

Résultats :

Tous les chemins sont des solutions (non-dominés).

Nombre d'itérations : - MOA* : 27 itérations.

- MOA* avec recherche TABOU : 25 itérations.
- MOA* avec heuristique d'ordre d'arrivée de OPEN : 19 itérations
- MOA* avec heuristique du moins de pères : 18 itérations.

algorithme	MOA*	MOA* avec recherche TABOU	MOA* avec ordre d'arrivée de OPEN	MOA* avec moins de pères
Nombre d'itérations	27	25	19	18

Tableau 5 : résultats de l'exemple 3

Interprétation des résultats :

Les résultats de cet exemple sont semblables à ceux de l'exemple 1 avec une différence qui réside dans l'écart enregistré dans la sélection aléatoire et la méthode TABOU.

Exemple 4 :

le graphe de cet exemple est comme l'exemple 3 en lui ajoutant des chemin dominés par la valeur (10 10) dès le départ et qui vont vers les nœuds {2,4,5,9,10,11,12,15,16,17,18} de telle sorte que ces sommets aient chacun 13 père. On aura les résultats suivants :

Nombre d'itérations : - MOA* : 22 itérations.

- MOA* avec recherche TABOU : 26 itérations.
- MOA* avec heuristique d'ordre d'arrivée de OPEN : 20 itérations.
- MOA* avec heuristique de moins de pères : 29 itérations.

	MOA*	MOA* avec recherche TABOU	MOA* avec ordre d'arrivée de OPEN	MOA* avec moins de pères
Nombre d'itérations	22	26	20	29

Tableau 6 : résultats de l'exemple 5

Interprétation des résultats :

En introduisant 13 arcs entrants vers chacun des nœuds {2,4,5,9,10,11,12,15,16,17,18} avec des valeurs dominées par la valeur de l'arc allant de 0 à 1, on force l'algorithme MOA* avec heuristique de moins de pères à sélectionner le nœud 1 avant les autres car il a 12 pères. Or, cet arc fait augmenter des itérations à l'algorithme en faisant des déplacements inutiles de OPEN vers CLOSED et de CLOSED vers OPEN. Ce qui fait que dans ce genre de graphes, l'heuristique de moins de pères n'est pas intéressante.

4 Tableau englobant les résultats des 4 exemples :

	Exemple 1	Exemple 2	Exemple 3	Exemple 4
MOA* avec heuristique 1	7	10	27	22
MOA* avec heuristique 2	7	10	19	20
MOA* avec heuristique 3	6	10	18	29
MOA* avec recherche Tabou	7	15	25	26

Tableau 7 : comparaison des 4 algorithmes

Heuristique 1 : représente l'heuristique de sélection aléatoire.

Heuristique 2 : représente l'heuristique de sélection selon l'arrivée de OPEN.

Heuristique 3 : représente l'heuristique de sélection du noeud aux moins de pères.

5. Conclusion :

L'heuristique de moins de pères semble la plus intéressante sauf dans le type de problèmes dont les graphes possèdent la caractéristique déjà citée (même nombre de pères pour la plupart des nœuds). L'ordre d'arrivée des nœuds de la liste OPEN est également intéressant à implémenter dans le cas de graphes riches en matière d'arcs et de branches. La méthode Tabou n'enregistre pas le meilleur des résultats, mais elle est classée comme intéressante car, elle n'enregistre pas un grand écart par rapport aux autres (exemple 1) et ça malgré qu'elle fait des itérations supplémentaires lors des déplacement des nœuds dans la liste Tabou : si la liste Tabou est entièrement exploitée, la méthode perd six (6) itérations « gratuitement ».

Chapitre 5 :

Application du MOA* sur un problème de réseau de télécommunication

1. Introduction :

La conception des réseaux de télécommunications mobiles est un problème peu étudié dans la réalité comparé au problème classique d'affectation de fréquences [MEU.02]. La plupart des travaux concernant le problème de conception ont étudié des réseaux de petites tailles (réseau micro-cellulaire ou intérieur) et utilisent généralement des modèles mono-objectif, dans lesquels un seul objectif est optimisé (couverture, agrégation d'objectifs, etc.) [V&H.00]. Un exemple récent d'une telle étude a été réalisé dans le cadre du projet européen esprit IV ARNO (Algorithms for Radio Network Optimization) dans lequel France Telecom R&D était le coordinateur. Le présent travail constitue un model de ce genre d'applications qui est proposé par France Telecom.

Dans ce chapitre, un modèle multicritère pour la conception de réseaux de télécommunications mobiles est présenté. Nous présentons les notions de base associées à la problématique (cellule, antenne, modèle de propagation, etc.). Les objectifs et les contraintes associés au modèle sont détaillés.

2. Position du problème :

Le développement sans précédant de la téléphonie mobile et la demande dans ce secteur incite les opérateurs à déployer des réseaux de télécommunication mobiles de plus en plus performants pour satisfaire la demande des consommateurs. Le but d'un réseau de téléphonie mobile est de permettre l'accès au réseau téléphonique fixe depuis un terminal portatif mobile (portable). Les communications entre mobiles se font par l'intermédiaire d'une liaison radio. Pour que la communication soit effective, les stations mobiles communiquent par l'intermédiaire d'une station de base fixe (BST, *Base Station Transmitter*) (Figure 17). Pour assurer un service satisfaisant, la qualité du signal radio doit être suffisante, ce qui nécessite une puissance importante des émetteurs. Afin de limiter cette puissance, l'opérateur place un ensemble de BST sur le territoire de manière à ce que tout terminal mobile se trouve à moins de quelques kilomètres de l'une d'entre elles. Ainsi, il pourra être nécessaire au terminal mobile de changer la station de base sur laquelle est relié le terminal, tout en maintenant la communication. On parle alors de mécanisme de *transfert intercellulaire* ou *Handover*. Chose qui n'était pas assurée dans les premiers systèmes appelés « *sans cordon* », qui ne présentent pas la possibilité de se déplacer librement sur le territoire, tout en maintenant la communication.

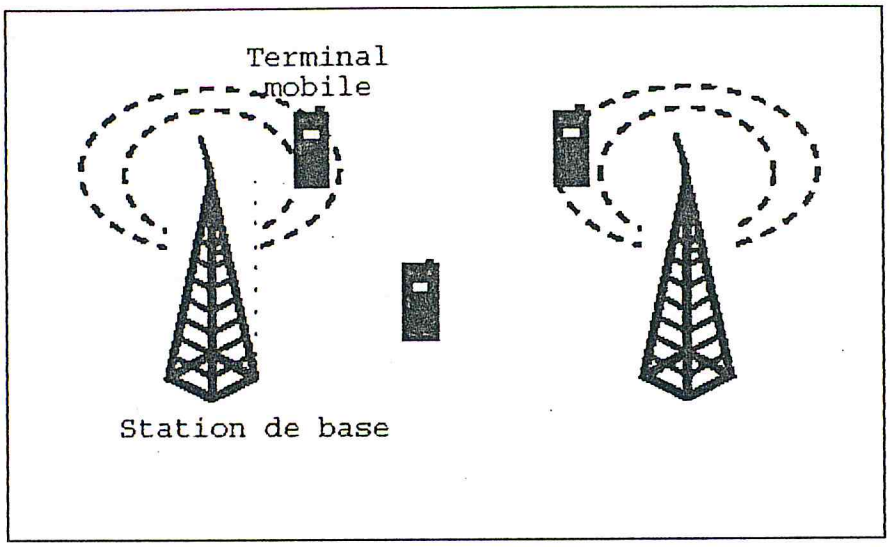


Figure 17 : station de base fixe (BST)

Les systèmes cellulaires actuels, tels que GSM ou GPRS, permettent une totale liberté de déplacement sur la zone couverte. Lors du déploiement du réseau, l'opérateur doit faire face à une contrainte de fonctionnement d'une station de base. Une BST ne peut écouler qu'une quantité de communication (trafic) limitée. Le déploiement du réseau en milieu rural met alors en oeuvre des cellules de grande taille, jusqu'à 30Km de diamètre, on parle alors de réseau macro-cellulaire (Figure 18). Tandis que le déploiement dans un environnement urbain et sub-urbain génère un trafic dense et nécessite donc une multitude de petites cellules pour écouler le trafic.

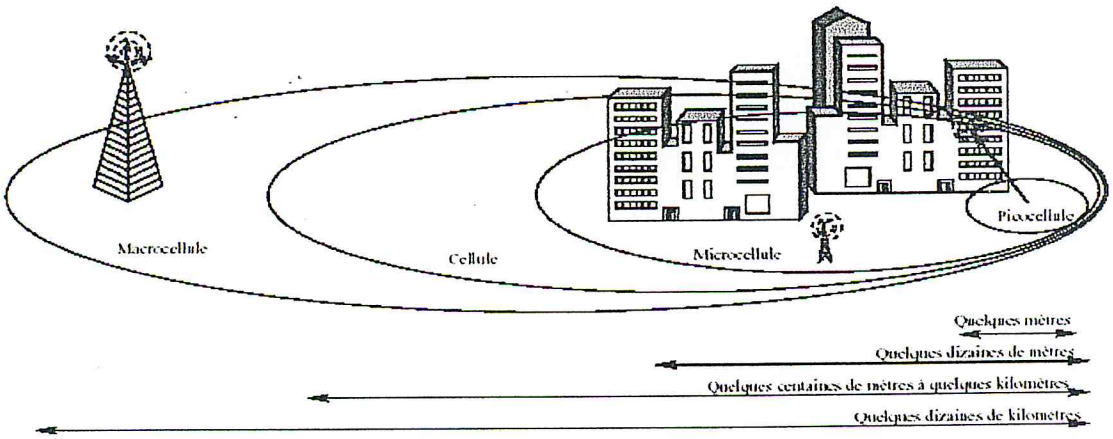


Figure 18 : Réseau macro-cellulaire.

3. Différentes étapes d'installation d'un réseau cellulaire

L'installation d'un réseau cellulaire est composé de trois étapes importantes : le calcul des dimensions du réseau (dimensionnement), la conception du réseau, et le déploiement.

1ere étape : Dimensionnement du réseau :
Dans cette étape, il s'agit de planifier les moyens qui devront être mis en oeuvre pour satisfaire les objectifs commerciaux, et les besoins en terme d'équipements et de qualité de service. C'est donc une étape de prévision des ressources nécessaires pour répondre à l'évolution de la demande et des services.

2eme étape : Conception du réseau :
Il s'agit ici d'effectuer la recherche des sites, du choix des antennes, de leur paramétrage, de valider les couvertures, et de calculer et valider les plans de fréquence. Les solutions issues du design sont alors évaluées par les équipes de dimensionnement et de déploiement.

3eme étape : Déploiement du réseau :
Cette étape consiste à déployer le réseau de manière à le rendre opérationnel et doit par exemple résoudre les divers problèmes d'autorisation pour l'installation des sites. Il reste ensuite l'optimisation fine sur le terrain.
Le déploiement du réseau exige de configurer chaque BST de manière à optimiser les objectifs considérés. Le problème qui consiste à optimiser le nombre de BST nécessaires pour couvrir une zone géographique avec une qualité de service optimale, tout en assurant un bon écoulement des communications (trafic), est un problème multicritère difficile auquel doit faire face les opérateurs de télécommunication lors de la conception d'un réseau performant. Dans le but d'automatiser le processus d'optimisation d'un réseau, il est nécessaire de développer un modèle d'optimisation de réseau cellulaire sous une forme multicritère.

Ce chapitre développe la modélisation de ce problème multi-objectif (multicritère) proposé par France Telecom. La résolution de ce problème par l'algorithme MOA* consiste à générer l'ensemble des solutions possibles (ensemble Pareto Optimal), dont les opérateurs de France Telecom vont choisir une (des) solution(s) qui les intéresse(nt).
Le problème consiste à choisir un sous-ensemble de sites parmi un ensemble de sites potentiels, à placer des antennes(BST) sur ces sites, configurées de telle manière qu'une

contrainte soit assurée tandis que les deux objectifs à optimiser sont : minimisation du nombre de sites et maximisation du trafic écoulé.

4. Données du problème :

Les données fournies par France Telecom sont des valeurs de coûts relatifs au type du matériel utilisé (type d'antennes, type de terminaux mobiles), et à la zone géographique (densité de population, qualité du champs d'émission/réception, etc.). Ces données sont fournies de telle sorte que pour chaque état du problème (pour un certain type d'antenne, une densité de population et un type de terminal), on fait correspondre un coût qui lui est relatif. Ces coûts sont calculés en fonction de combinaisons des cas possibles par des opérateurs de cet organisme. Il en résulte que le présent travail est une suite d'une autre étude faite au niveau de France Telecom pour valoriser le problème à optimiser.
Pour bien cerner le problème, nous allons présenter quelques notions de base concernant la problématique.

4.1 Zone géographique :

Une zone géographique Z représente l'emplacement sur lequel le réseau est déployé. Elle est caractérisée par :

- Une liste d'emplacements potentiels de sites, notée $L = \{L_i / L_i \in Z, i \in \aleph\}$. Chaque site est défini par ses coordonnées dans la zone géographique. Un réseau est donc constitué d'une sous liste de sites sur lesquels sont placées les antennes (les BST).
- Un ensemble de points de la zone géographique pour lesquels la puissance du champ est calculable : $PCC = \{PCC_i / PCC_i \in Z, i \in \aleph\}$
- Un ensemble de points de la zone géographique $ASM = \{ASM_i / ASM_i \in Z, i \in \aleph\}$ pour lesquels un service minimum doit être assuré, afin de permettre l'établissement de la communication. A chaque ASM_i est associé un seuil de service S_i.
- Un ensemble de points de test de la zone géographique qui comportent une quantité statique de trafic potentiel estimée par l'opérateur $QST = \{QST_i / QST_i \in Z, i \in \aleph\}$.

Les ensembles PCC, ASM, QST vérifient la relation d'inclusion $QST \subset ASM \subset PCC$. La figure (19) illustre un exemple de zone géographique avec les ensembles PCC, ASM et QST associés.

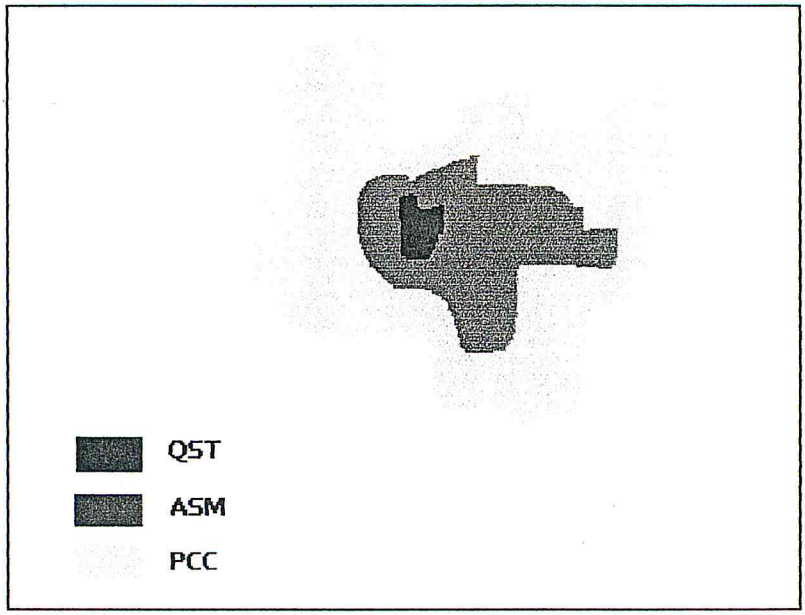


Figure 19 : Exemple de zone géographique

4.2 Données concernant le matériel

4.2.1 Les antennes (BST)

Les BST sont les éléments du réseau qui génèrent un champ porteur des communications entre mobiles. Les antennes utilisées sont caractérisées par leur gain, et un diagramme d'affaiblissement. Nous avons considéré 3 types d'antennes :

– l'antenne à diagramme omni-directionnel (AO) qui émet uniformément dans toutes les directions ;

– l'antenne à diagramme sectoriel large (ASL) qui émet un champ dans un secteur large ;

– l'antenne à diagramme sectoriel étroit (ASE) qui émet un champ dans un secteur étroit.

Un site installé a pour contrainte d'être équipé soit d'une seule antenne omni-directionnelle, soit d'une à trois antennes sectorielles, chacune pouvant être étroites ou larges.

Chaque antenne est équipée d'un ou plusieurs dispositifs de transmission TRX *(transmitters-receivers*) en fonction de la quantité de trafic à supporter. Chaque BST peut recevoir sept TRX au maximum, ce qui correspond à un trafic écoulé de 43 Erlang par BST (Tableau 8). Lorsque le trafic écoulé par la BST dépasse ce seuil, le surplus de trafic est perdu. En pratique, pour une quantité donnée de trafic à écouler, le nombre de TRX à installer sur la BST est déduit par interpolation suivant le tableau de correspondance Erlang/TRX en fonction du nombre d'Erlang. Par exemple, pour un trafic de cinq Erlang, deux TRX seront utilisés.

TRX	1	2	3	4	5	6	7
Erlang	2.9	8.2	15	22	28	35.5	43

Tableau 8 : Correspondance TRX-Erlang

Les principaux paramètres d'une antenne sont donc : la puissance d'émission, le nombre de transmetteurs (TRX) qui lui sont affectés et le diagramme d'affaiblissement pour les antennes directionnelles.

4.2.2 Les stations mobiles :

Un réseau fournit un service pour une catégorie de stations mobiles. Un mobile *(Mobile Station*, MS) possède un équipement d'émission/réception de puissance variable, selon la nature du service fourni (Tableau 9). Pour que la MS soit capable d'établir une communication, il est nécessaire que la puissance du signal radio qu'elle reçoit soit supérieure à son seuil de service S, exprimé en décibel (dB).

Mobile	Service selon le type de MS	Sensibilité en dB
2 watt	En voiture	-78
2 watt	Extérieur	-84
8 watt	En voiture	-82
8 watt	Extérieur	-90

Tableau 9 : type de mobile (MS) selon le service requis.

5. Modélisation multicritère du problème :

Après avoir défini le problème du réseau de télécommunication dans les paragraphes précédents, nous nous intéressons, dans cette partie, à sa modélisation multicritère pour générer ses solutions possibles avec l'algorithme A* multi-objectif. Cette tache consiste à définir le modèle mathématique associé à ce problème.

Le problème est constitué de deux critères à optimiser, à savoir :

- minimisation du nombre de sites utilisés
- Maximisation du trafic écoulé.

Ceci en la présence d'une contrainte à respecter : celle de la couverture de la zone géographique par le champ d'émission/réception.

5.1. *Minimisation du nombre de sites utilisés :*

La fonction de minimisation du nombre de sites, F_1, est donnée par le model mathématique suivant :

$$F_1 = \min \sum_{i=1}^{n} f_i y_i$$

$$\text{Avec } y_i = \begin{cases} 1, & \text{si l'emplacement } L_i \text{ est utilisé} \\ 0, & \text{sinon} \end{cases}$$

Tel que f_i est le coût d'installation du site i, et n : le nombre d'emplacements potentiels dans la zone géographique.

5.2. Maximisation du trafic écoulé :

La fonction de maximisation du trafic écoulé F_2, est donnée par le model mathématique suivant :

$$F_2 = \max \sum_{i=1}^{n} \sum_{j=1}^{m} QST_{ij} z_i$$

$$\text{Avec } z_i = \begin{cases} 1, & \text{si l'emplacement } L_i \text{ est utilisé} \\ 0, & \text{sinon} \end{cases}$$

Tel que QST_{ij} est le trafic écoulé du point de test j appartenant à la zone i et m : le nombre de points de test dans la zone géographique i .

Cet objectif revient à maximiser la somme des trafics écoulés dans chaque point de test dans tous les emplacements utilisés (emplacement où l'on a installé les BST).

5.3 Contrainte du problème :

Dans ce problème, la contrainte à qui doit répondre les solutions réalisables (ensemble Pareto Optimal) est la contrainte de couverture. Ceci se traduit par le fait que chaque ASM_i de la zone géographique soit supérieur au seuil S_i. Cette contrainte est traduite par la formule mathématique suivante : $\forall i = 1,..n, ASM_i > S_i$

6. Modélisation du problème en graphe d'état :

6.1. Données réelles du problème

Le problème est donné sous forme de deux jeux de données :

- Jeu de données avec 50 sites potentiels
- Jeu de données avec 100 sites potentiels

Chaque jeu de données est représenté par trois vecteurs et une valeur de la contrainte

6.1.1. Les vecteurs de données

Les trois vecteurs du problème sont : vecteur des sites potentiels, vecteur du trafic écoulé pour chaque site et le vecteur de couverture de chaque site.

Pour le problème avec 50 sites potentiels, la taille des vecteurs est 50 ; et pour les données avec 100 sites potentiels, la taille des vecteurs est 100.

Pour généraliser le nombre de sites potentiels, on suppose qu'il y a ***n*** sites.

(a) Vecteur des coûts d'installation des sites ***f***

C'est un vecteur de taille *n*, qui représente pour chaque site potentiel, le coût d'installation qu'il lui est associé.

$f = (f_1, f_2, f_3, ..., f_n)$ Tel que f_i est le coût associé à l'installation du site i.

Ce vecteur est celui de la fonction de minimisation du nombre de sites cité dans le paragraphe (4.1).

(b) Vecteur des trafics écoulés *T*

C'est un vecteur de taille *n*, qui représente les trafics écoulés pour chaque site potentiel

$T = (t_1, t_2, t_3, ..., t_n)$ Tel que t_i est le trafic écoulé du site i.

(c) Vecteur des contraintes *R*

C'est un vecteur de taille *n*, représentant une information sur la satisfaction de la contrainte de couverture par les sites potentiels.

$R = (r_1, r_2, r_3, ..., r_n)$ Tel que r_i est la contrainte associée au site i.

6.1.2 La valeur de contrainte C

Cette valeur correspond à un maximum qu'il ne faut pas atteindre par la somme des informations sur la satisfaction de la contrainte de couverture des sites utilisés. D'une autre manière, si on a 3 sites à installer : S_1, S_2, S_3, alors la contrainte suivante doit être satisfaite :

$r_1 + r_2 + r_3 < C$

Cette propriété parait illogique le fait qu'on cherche à ce que la couverture soit élevée. Mais en réalité cette valeur de contrainte (C) ne correspond pas à la couverture elle-même, mais elle est donnée de telle manière à ce qu'elle exprime le contraire de la couverture (la non-couverture).

6.2 Modélisation du problème sous forme de graphe d'état :

- La notion de configuration peut être définie ici comme un ensemble d'information contenant : le nombre de sites, le coût d'installation de ces sites, le trafic écoulé par cet ensemble de sites et le cumul des valeurs de contraintes correspondant aux sites.
- Un état de l'algorithme peut correspondre à un ensemble de configurations.
- Chaque nœud du graphe correspond à un état de l'algorithme.
- Un arc entre deux nœuds représente le passage d'un état à un autre en rajoutant un site.

Exemple : étant donné un problème avec 3 sites, le graphe correspondant est représenté par la figure 20 :

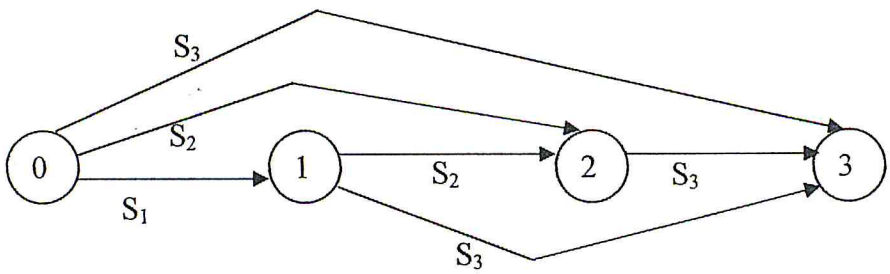


Figure 20 : exemple avec trois sites

- Le nœud 0 représente l'état initial : aucun site n'est installé.
- L'arc entre le nœud 0 et le nœud 1 indique qu'on a sélectionné le site S_1 pour l'installation.
- L'arc entre le nœud 0 et le nœud 2 indique qu'on a sélectionné le site S_2 pour l'installation.

- L'arc entre le nœud 0 et le nœud 3 indique qu'on a sélectionné le site S_3 pour l'installation.

On voit, donc à travers cet exemple, que les arcs correspondent au changement d'état de l'algorithme en rajoutant un nouveau site. Leurs valeurs représentent le coût d'installation.

Taille de l'espace de recherche

Tous les cas de figure doivent être pris on considération dans le graphe, ce qui donne que l'espace de recherche est un arbre binaire de profondeur n (taille des vecteurs). Ce qui donne un nombre de cas d'ordre de 2^n.

5. Résultats obtenus

Nous allons dans cette partie donner les résultats obtenus pour deux jeux de données de l'application avec 50 sites potentiels. Pour chacun d'eux, on présentera un tableau contenant l'ensemble Pareto optimal de solutions (avec deux critères), ainsi qu'une représentation graphique des solutions.

Remarque : *comme cité avant, les données sont fournies cryptées pour des raisons de confidentialité. Par conséquent, le coût de l'installation ne correspond pas à une fonction minorante, mais à une fonction majorante. C'est-à-dire que dans les résultats qui viennent le coût de l'installation correspond au rendement de l'installation. Donc minimiser le coût revient à maximiser le rendement de l'installation.*

5.1. premier jeu de données

a- TABLEAU DES RESULTATS :

Coût d'installation	Trafic écoulé	Nombre de sites installés
354	457	6
384	446	6
395	441	6
414	415	6
427	408	6
448	398	6
449	356	6
481	325	6
468	332	6

Tableau 10 : résultats du premier jeu de données

Les valeurs des couples formés des deux critères (coût d'installation et le trafic écoulé) sont des valeurs non-dominées. Ces couples forment l'ensemble Pareto optimal. On remarque d'après le tableau 10. que le nombre de sites utilisés est toujours égal à 6, mais

le coût de ces sites est variable. Ceci explique le fait que les antennes ne sont pas de même type, ainsi que l'emplacement de l'installation.

b- REPRESENTATION GRAPHIQUE DES RESULTATS

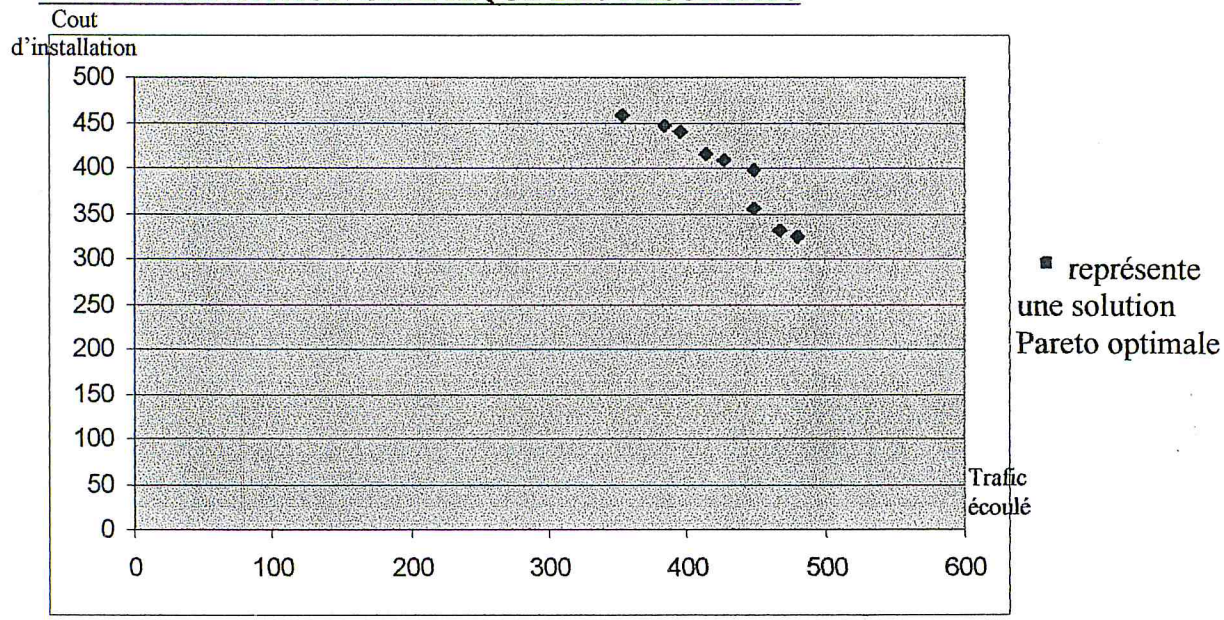


Figure 21 : Représentation graphique des résultats du premier jeu de données

Dans la figure (21), l'axe des X représente le coût d'installation, et l'axe des Y représente le trafic écoulé. Les points en bleu représentent les solutions présentées dans le tableau (10). Elles représentent l'ensemble Pareto Optimal dans l'ensemble des réalisations du problème. En d'autres termes, elles sont les combinaisons possibles entre le coût d'installation total du réseau et le trafic écoulé. Cet ensemble est irréductible, car toutes les solutions sont non-dominées (ensemble Pareto Optimal).

La jointure des points par un trait forme une courbe appelée Frontière Pareto

7.2. Deuxième jeu de données

TABLEAU DES RESULTATS

Coût d'installation	Trafic écoulé	Nombre de sites installés
1820	1595	21
1801	1598	21
1772	1665	21
1663	1690	21

Tableau 11 : Résultat du deuxième jeu de données

REPRESENTATION GRAPHIQUE DES RESULTATS

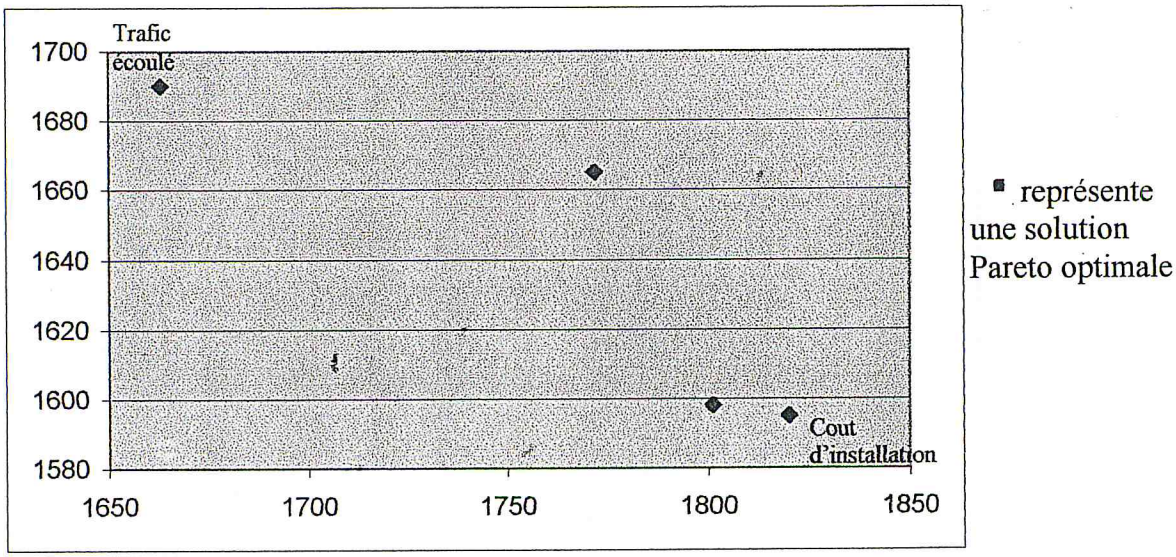


Figure 22 : Représentation graphique des résultats du deuxième jeu de données

Dans ce jeu de données, il y a 4 solutions possibles dans l'espace des résolutions, représentées par les points bleus dans la figure (22). Ces solutions forment l'ensemble Pareto optimal.

Remarque : Le problème de réseau de télécommunication donné par France Telecom est un prototype des applications réelles résolues au sein de cette compagnie, mais avec une petite taille. Mais la complexité de ces applications ne réside pas toujours dans la taille du problème, mais aussi dans le type des données et l'ordre des objectifs. Dans notre cas, le calculateur

prends un temps d'exécution énorme pour exécuter l'algorithme MOA* avec le jeu de données possédant des vecteurs de taille 100 fourni par France Telecom. Ceci fait appel au parallélisme pour pouvoir le résoudre. D'ailleurs, cette technique est indispensable pour les utilisateurs de ce domaine.

8. Nombre d'itérations des trois versions du MOA*

Tableau des résultats

	MOA* avec selection aléatoire	MOA* avec ordre d'arrivée de OPEN	MOA* avec moins de pères	MOA* avec recherche TABOU
1er jeu de données	56	54	54	54
2eme jeu de données	485	211	211	152

Tableau 12 : Résultats des versions du MOA* sur les données de l'application

On remarque, d'après les résultats du tableau (12), que les heuristiques d'ordre d'arrivée et avec moins de pères et la métaheuristique de recherche Tabou apportent le même nombre d'itérations pour le premier jeu de données (54). Quand à la sélection aléatoire, elle enregistre une dégradation de deux itérations.

Pour le deuxième jeu de données, la métaheuristique de recherche Tabou est la plus intéressante. Elle enregistre un grand écart par rapport aux deux autres heuristiques (59 itérations). Ceci prouve ce qui a été déjà dit sur cette technique, qu'elle diminue le nombre d'itérations quand le déplacement des nœuds est élevé. Ce qui est le cas pour ce jeu de données où les traitements sont nombreux et le nombre d'itérations minimal est 152.

CHAPITRE 6

Conclusion et perspectives

L'optimisation multi-objectif est une branche très importante dans la résolution des problèmes réels. La présence de différentes approches de résolution des problèmes multi-objectif offre aux chercheurs la possibilité de choisir un axe de résolution très vaste. Malgré la restriction des deux approches (approche à base de transformation d'un problème multi-objectif vers un problème multi-objectif et l'approche non-Pareto), elles restent toujours utilisées dans certains problèmes avec une faible indépendance entre les critères à optimiser. Les approches Pareto, qui forment le pole le plus important dans la résolution des problèmes multi-objectif, reposent sur le principe d'optimalité Pareto des solutions : Toute solution de cet ensemble est optimale dans le sens qu'aucune amélioration ne peut être faite sur une composante du vecteur sans dégradation d'au moins une composante du vecteur. Cette propriété permet de maintenir l'aspect multi-objectif du problème à résoudre et assure qu'une solution n'appartenant pas à cet ensemble n'est plus intéressante.

Nous avons présenté dans ce travail un algorithme très efficace pour la résolution de problèmes multi-objectif. L'algorithme MOA*, tel qu'il a été proposé par BRADLEY S. STEWART et CHELSEA C. WHITE, peut résoudre des problèmes très complexes de façon exacte, avec la garantie de toujours trouver les meilleures solutions. Cette caractéristique a été hérité de l'algorithme A* qui reste l'algorithme le plus utilisé dans la résolution des problèmes mono-objectif sous forme de graphes d'état. Faisant partie des approches Pareto, le MOA* travaille avec une organisation très apparente pour l'utilisateur. Les étapes d'évolution de cet algorithme sont transparentes du moment que chaque ensemble contient des données spécifiques de l'état du graphe à une certaine itération de l'algorithme. Ce qui contribue énormément dans la facilité d'implémentation de l'algorithme et le manipuler de telle sorte qu'il soit adéquat avec le problème à résoudre.

Nous avons jugé les améliorations apportées à l'algorithme MOA* de pertinentes, et ceci en faisant référence aux performances enregistrées sur le nombre d'itérations de l'algorithme, qui agit d'une manière directe sur le temps d'exécution de l'algorithme. Les performances des heuristiques et la métaheuristique introduites pour cet algorithme (sélection aléatoire, sélection selon l'ordre d'arrivée, sélection avec moins de pères et la métaheuristique de recherche Tabou) varient selon le type du problème. Nous avons prouvé cette thèse sur des exemples différents où chaque heuristique apporte des améliorations sur un exemple

possédant des caractéristiques différentes de celles des autres exemples. Ces caractéristiques peuvent être résumé es comme suit :

- l'heuristique avec moins de pères est intéressante sauf dans les graphes où la plupart des nœuds possèdent le même nombre de pères.
- L'heuristique d'ordre d'arrivée de la liste OPEN est la concurrente de la première. Elle est la plus intéressante dans les graphes riches en matière de nœuds et d'arcs possédant la caractéristique non favorable à la première (même nombre de pères pour la plupart des noeuds).
- Les améliorations qu'apporte la sélection aléatoire sont du pur hasard. Elle n'est pas intéressante dans la plupart des cas.
- Les performances de la métaheuristique de recherche Tabou ne sont pas loin des deux premières heuristiques, malgré qu'elle perde des itérations lors des déplacements des nœuds vers la liste TABOU.

Le choix d'une version parmi les quatre nécessite une connaissance à priori du problème par l'utilisateur, ainsi qu'un aperçu sur le graphe d'état.

La difficulté de l'algorithme MOA* apparaît dans certains problèmes où les données varient selon l'environnement du problème. Ces problèmes sont connus sous le nom de *problèmes à environnement dynamique*. L'état des données ne peut pas être prédéfini dans ce genre de problèmes. Cette difficulté incite les chercheurs à travailler avec les méthodes approchées (algorithmes évolutionnaires avec métaheuristiques) où l'on définit une solution aléatoire, et on essaye de rapprocher la solution exacte à l'aide de la notion de voisinage. Ces méthodes sont très utilisées dans des problèmes dont on ne connaît pas le comportement. Ceci ne remet pas en cause l'efficacité de l'algorithme MOA* puisqu'il est présenté comme méthode exacte applicable sur une classe de problèmes à caractère déterministe.

L'application du MOA* sur un réseau de télécommunication, proposé par France Télécom, a donné des résultats répondant aux besoins du domaine de télécommunication. L'ensemble non-dominé de solutions (ensemble Pareto Optimal) laisse aux décideurs le choix de la solution selon leurs préférences. Ce type d'applications, qui consistent à minimiser le nombre de sites et à maximiser le trafic en assurant la couverture de la totalité du réseau, est un problème très fréquent ces dernières années. Il constitue la préoccupation majeure des compagnies de télécommunication, vue la demande croissante dans ce domaine.

Il reste à noter que nos contributions dans ce travail ont été faites de telle manière qu'elles soient adéquates avec la classe des problèmes multi-objectif, et qu'elles répondent aux besoins de cette classe. Ceci implique l'introduction de quelques perspectives concernant

l'implémentation de l'algorithme MOA* dans ces différentes versions et les améliorations qui peuvent être apportées dans les prochains travaux concernant cette approche. Ces perspectives peuvent être résumées comme suit :

- l'introduction des outils permettant de mieux estimer le parcours restant dans le graphe (la fonction *h* de l'algorithme MOA*). Ces heuristiques doivent satisfaire la propriété d'admissibilité (les estimations ne doivent pas être loin des valeurs réelles)
- Introduire des outils d'aide à la décision pour le choix des nœuds à traiter durant les itérations de l'algorithme
- Introduire un mécanisme de rapprochement de la solution durant l'évolution du MOA* pour prendre en compte l'aspect dynamique du problème, en faisant appel à la notion de *génération* utilisée par les algorithmes génétiques.

Ces perspectives seront intéressantes dans les prochains travaux qui porteront sur le développement de l'algorithme A* multi-objectif en particulier et sur les méthodes exactes en général.

BIBLIOGRAPHIE

[B&C.91]: BRADLEY S. STEWART and CHELSEA C. WHITE 1991, "Multiobjective A*" University of Virginia, Charlottesville, Virginia.

[BER.01] : ALAIN BERRO. 2001 Thèse de doctorat, « Optimisation multi-objectif et stratégies d'évolution en environnement dynamique »
Université de Toulouse.

[B&H.99]: V. BARICHARD and J-K. HAO, "An Empirical Study of Tabu Search for the Multi-objective Knapsack Problem".
LERIA - Faculty of Sciences - University of Angers, 2 Boulevard Lavoisier, 49045 Angers Cedex 01 FRANCE
Email: {Vincent.Barichard, Jin-Kao.Hao}@info.univ-angers.fr

[GOL.89]: DAVIS.E.GOLDBERG and J.J. RICHARDSON. 1989 "genetic algorithm with sharing for multimodal function optimisation, genetic algorithms and their applications".

[H&M.79]: Hwang, C. and Masud, A. 1979. Article sur:"Multiple objective decision making - methods and applications". Springer Verlag, Berlin.

[MEM.03]: W. HADJADJ et A.MEKKAOUI. 2003 Thèse d'ingéniorat, « Réalisation d'un outil de planification d'emplois du temps basé sur les métaheuristiques ».
Institut National d'Informatique (INI), Oued-smar, Alger.

[MEU.02] : HERVE MEUNIER. 2002 Thèse de doctorat, « Algorithmes évolutionnaires parallèles pour l'optimisation multi-objectif de réseaux de télécommunications mobiles »
Université de Lille.

[PEA.90] : JUDEA PEARL.
« Heuristique, stratégies de recherche intelligente pour la résolution de problèmes par ordinateur ».

[TDI.01] : « Cours d'intelligence artificielle », 2001. Université Des Sciences et des Technologies de Lille.

[ZEG.INI] : Dr ZEGGOUR. Cours de « méthodes de conception de programmes ». www.ini.dz

[VAN.02] : MICHAEL VAN CANEGHEM décembre 2002, « le voyageur de commerce:algorithme Branch and Bound, algorithme glouton, méthode de recherche locale ».

[V&H.00]: Vasquez, M. and Hao, J.-K. "A heuristic approach for antenna positioning in cellular net-works".