UNIVERSITÉ DU QUÉBEC EN ABITIBI-TÉMISCAMINGUE

MÉMOIRE
PRÉSENTÉ À
L'UNIVERSITÉ DU QUÉBEC À CHICOUTIMI
COMME EXIGENCE PARTIELLE
DE LA MAÎTRISE EN INGÉNIERIE

PAR
ADIL KAMIL

APPLICATION D'UN ALGORITHME HYBRIDE À COLONIES DE FOURMIS AU PROBLÈME D'AFFECTATION QUADRATIQUE

FÉVRIER 2008

CE MÉMOIRE A ÉTÉ RÉALISÉ
À L'UNIVERSITÉ DU QUÉBEC EN ABITIBI-TÉMISCAMINGUE
DANS LE CADRE DU PROGRAMME
DE MAÎTRISE EN INGÉNIERIE
DE L'UNIVERSITÉ DU QUÉBEC À CHICOUTIMI
OFFERT PAR EXTENTION À L'UNIVERSITÉ DU QUÉBEC
EN ABITIBI-TÉMISCAMINGUE

RÉSUMÉ

Le problème d'aménagement d'usines consiste à trouver l'arrangement le plus efficace d'un nombre donné de départements. Ce problème est un problème complexe et difficile. Les méthodes de résolution exactes sont limitées à des problèmes de taille relativement petite. C'est la raison pour laquelle de nombreuses méthodes approchées ont été adaptées à la résolution de ce problème. Parmi ces adaptations on retrouve les algorithmes à base de méta-heuristiques. Dans ce mémoire, le problème d'aménagement d'usines est modélisé comme un problème d'affectation quadratique qui est étudié en utilisant la méta-heuristique des colonies de fourmis. Cette dernière s'inspire du comportement collectif du dépôt et du suivi de pistes observées dans les colonies de fourmis réels. L'algorithme proposé est couplé à un algorithme de recherche locale permettant d'améliorer les solutions générées par les fourmis. L'exécution de cet algorithme est évaluée selon deux facteurs : la qualité de solution et le temps de calcul. Des résultats numériques illustrant l'approche proposée sont présentés.

REMERCIEMENTS

Cette recherche s'est réalisée grâce au soutien et à l'encouragement de plusieurs personnes qui ont contribué chacune à leur façon à l'atteinte de l'objectif que constitue le dépôt de ce mémoire. Je tiens à leur adresser un très grand merci.

Je tiens à remercier très sincèrement mon directeur de recherche le Professeur Mustapha Nourelfath, pour sa grande disponibilité tout au long de ce travail. Son encadrement, ses conseils et son expérience m'ont permis d'acquérir les connaissances indispensables au développement et à l'aboutissement de ce travail.

Je remercie également Monsieur Nabil Nahas pour son support. Il m'a souvent encouragé et supporté lors de la réalisation de ce travail, et a m'a généreusement fait profiter de son expérience et de ces connaissances. Je lui en suis très reconnaissant.

Je tiens également ma profonde reconnaissance pour L'Université du Québec en Abitibi-Témiscamingue pour le support financier à travers une exemption partielle des frais de scolarité.

Également, je présente mes remerciements et ma gratitude à mes parents pour leur soutien moral et financier à la poursuite de mes études. Vous m'avez toujours encouragé à poursuivre mes rêves et à me dépasser. Ce mémoire vous est dédié.

Finalement, à toutes les autres personnes qui ont gravité autour de moi pendant ces deux dernières années, qui m'ont supporté et encouragé. Je vous dis un gros merci. J'espère que ce mémoire saura combler vos attentes.

Bonne lecture!

TABLE DES MATIÈRES

Chapitre 2 *Revue de la littérature*

Chapitre 3 *Méthodologie*

Chapitre 4 *Expérimentation et Résultats*

Chapitre 5 *Conclusion*

LISTES DES FIGURES

Chapitre 1

Chapitre 2

Chapitre 3

LISTE DES TABLEAUX

Chapitre 2

Chapitre 4

INTRODUCTION GÉNÉRALE

Le marché des consommateurs d'aujourd'hui exige que les fabricants soient concurrentiels. Afin de maintenir la compétition, cela requiert des opérations efficaces de la part des usines et des capacités à répondre rapidement aux changements de produits. L'aménagement d'usines est un domaine de recherche exploité depuis longtemps. Les approches de conception ont évolué au fil des ans et la problématique est encore bien présente. Il existe un ensemble important de formulations et de modèles pour résoudre le problème de l'aménagement d'usines. Certaines de ces formulations sont simples, tandis que d'autres peuvent être complexes. L'une des formulations simples utilisée est la représentation par le problème de l'affectation quadratique (PAQ, ou QAP en anglais pour *Quadratic Assignment Problem*). Bien que simple à modéliser, cette formulation reste difficile à résoudre.

Le problème d'affectation quadratique est en effet un problème classique d'optimisation combinatoire dans lequel il convient de trouver le placement optimal de ***n*** objets ***''usines, matériels''*** dans ***n*** locations tout en minimisant le coût qui dépend à la fois des distances inter-locations et des flux inter-objets. Il s'agit de trouver l'affectation qui permettra de minimiser le coût total. Le QAP est un problème important en optimisation combinatoire qui possède de nombreuses applications (placement, ordonnancement,

synthèse d'images, etc.), dont les algorithmes exacts tel que l'algorithme par séparation et évaluation (*branch-and-bound*) ne peuvent traiter que des instances taille relativement petite.

L'objectif de ce travail est d'aborder le problème d'affectation quadratique en utilisant des méta-heuristiques comme approche de résolution, en particulier l'algorithme à colonies de fourmis couplé à l'algorithme du plafond dégradé.

Ce mémoire est organisé comme suit : le premier chapitre présente des généralités sur l'aménagement d'usines, une définition du problème d'affectation quadratique (PAQ) et quelques applications principales.

Dans le deuxième chapitre, une revue de la littérature présentera succinctement les différentes approches de résolution qui ont été proposées. En effet, nous nous intéresserons aux méta-heuristiques. Il existe deux types d'approches de résolution : les méthodes exactes et les approches heuristiques. Les sections *2.2* et *2.3* vont présenter brièvement une liste des méthodes disponibles. Pour des cas de petite taille, les méthodes exactes de résolution sont très performantes et donnent d'excellents résultats. Toutefois, lorsque la taille augmente, les méthodes exactes deviennent plus « lourdes » puisque leur temps augmente. Les méta-heuristiques forment une famille d'algorithmes d'optimisation. Ce sont des stratégies qui permettent de guider la recherche vers une solution optimale. Parmi ces méthodes, les algorithmes à colonies de fourmis forment une classe de méta-heuristiques récemment

proposée pour les problèmes d'optimisation difficile. Ces algorithmes s'inspirent des comportements collectifs de dépôt et de suivi de pistes observés dans les colonies de fourmis. Une colonie d'agents simples (les fourmis) communiquent indirectement via les modifications dynamiques de leur environnement (les pistes de phéromone) et construisent ainsi une solution à un problème donné en s'appuyant sur leur expérience collective.

Le troisième chapitre présentera l'approche que nous utilisons pour résoudre le problème d'aménagement d'usines modélisé comme un problème d'affectation quadratique. Dans le cadre de ce travail, les efforts de recherche ont été mis sur la résolution du problème d'affectation quadratique par les méta-heuristiques en appliquant une hybridation de deux méta-heuristiques. L'idée c'est de faire coopérer des algorithmes de recherche, ce qui signifie que nous parlons alors d'un algorithme hybride. Les résultats fournis par le premier algorithme sont les solutions initiales du second. En ce qui concerne l'hybridation que nous avons réalisée, un algorithme à colonies de fourmis et un algorithme à plafond dégradé collaborent pour résoudre le problème d'affectation quadratique.

Au quatrième chapitre, nous présentons un réglage des paramètres de l'algorithme hybride et des résultats expérimentaux issus de son application au PAQ. Nous terminons par une discussion de ces résultats.

Enfin, le dernier chapitre présente une conclusion et quelques perspectives de ce travail.

CHAPITRE 1

GÉNÉRALITÉS SUR L'AMÉNAGEMENT D'USINES

1.1 INTRODUCTION

Actuellement dans le monde des affaires, les entreprises font face à divers enjeux stratégiques et financiers, de telle sorte que la planification et l'aménagement d'usines deviennent nécessaires. De ce fait, les sociétés qui désirent réussir doivent régulièrement actualiser leurs stratégies d'aménagement afin d'adapter ses procédés au niveau de la production, pour satisfaire les besoins grandissants dans le marché. Aussi, cette capacité d'ajustement est indispensable pour demeurer à l'affût de la compétition et pour de nouvelles innovations technologiques.

Afin de répondre à la demande grandissante des consommateurs, les entreprises doivent maintenir ou augmenter leur rendement. Pour demeurer dans l'engrenage de la compétition, les sociétés doivent fréquemment prendre plusieurs décisions stratégiques

concernant la répartition départementale de leur entreprise. Bien que ces décisions aient un impact positif sur l'accroissement de l'entreprise, les résolutions adoptées sont centrées sur des problématiques anticipées. Néanmoins, au fur et à mesure que la compagnie va prendre de l'expansion, elle sera tout de même confrontée à certaines difficultés liées à l'aménagement, autrement dit, elle devra faire face à des problèmes d'aménagement.

La conception de l'aménagement d'une organisation relève d'une décision importante d'investissements et requièrent des efforts considérables, ainsi que des engagements à long terme. Son but essentiel est basé sur la détermination d'un arrangement départemental ou entre unités de travail (work unit) afin d'en tirer le plus de satisfaction (et d'en générer le maximum de profits) en minimisant le coût d'exploitation, en augmentant la qualité et en réduisant le délai de livraison au client. Généralement, le choix d'un arrangement doit supporter la stratégie de fabrication de cette entreprise. Cette stratégie de fabrication détermine la façon que l'entreprise a choisie pour satisfaire la demande de ses clients (produit à qualité supérieure, coût inférieur, délais de livraison très courts).

L'entreprise doit se positionner par rapport à ses déférents environnements, par la suite, elle aura à décider le type d'arrangement qu'elle voudra adopter. Il s'agit d'une décision importante qui doit se baser sur une anticipation de l'évolution du marché et du comportement de la compétition.

Critères qui affectent l'aménagement:

Dans l'aménagement, quelque soit l'environnement de fabrication, les entreprises ont intérêt à passer par différentes études et analyses qui leur permettront de mettre en jeu des critères majeurs et importants afin de déterminer l'arrangement départemental le plus efficace. Parmi ces critères, on peut citer les éléments suivants:

- Investissements en équipements de production, de manutention, d'entreposage et en bâtisse ;
- Politique de gestion de stock ;
- Coûts de fabrication dans la nouvelle implantation ;
- Coûts de gestion dans la nouvelle implantation ;
- Facilitation d'une expansion future, adaptabilité, flexibilité ;
- Efficience du circuit des produits ;
- Efficience de la manutention ;
- Efficience du stockage ;
- Utilisation des surfaces ;
- Utilisation des équipements ;
- Qualité du produit ;
- Compatibilité avec les plans à long terme de l'entreprise ;
- Conditions de travail et satisfaction du personnel ;
- Facilité de supervision et de contrôle ;
- Adaptation à la structure générale de l'entreprise.

C'est la prise en compte globale de ces critères majeurs qui permet de déterminer un arrangement efficace et bénéfique. La connaissance de ces critères conduit à la modélisation mathématique du problème de l'aménagement. Diverses analyses et études ont été effectuées afin de quantifier ces critères et de les représenter par quelques paramètres contrôlables. Les études ont démontré par exemple, que le flux et le volume de la matière transportée entre départements représentaient bien les paramètres de résolution dans un modèle mathématique.

Dans le cadre de ce travail, nous allons étudier le problème d'affectation quadratique qui est lié à l'aménagement d'usines. Ce dernier revêt un intérêt particulier au sein des entreprises, car ces dernières doivent faire face aux défis de l'économie. En effet, les sociétés doivent anticiper les impacts qu'auront l'ajout d'un produit à l'avenir. Par ailleurs, l'aménagement d'usines est un domaine de recherche qui est exploité depuis longtemps. Aussi, les approches de conception ont évolué au fil des ans et la problématique est encore bien présente.

La section suivante va concerner l'élaboration du problème d'aménagement d'usines tel que vu par divers chercheurs.

1.2 PROBLÈME DE L'AMÉNAGEMENT

Qu'une entreprise soit ancienne ou nouvellement établie, elle rencontre le même problème : l'aménagement de l'usine. Cette problématique d'ordre général doit être suivie d'une décision stratégique qui peut être bénéfique à long terme à l'entreprise qui désire augmenter son rendement opérationnel et monétaire, et répondre aux exigences de la demande des consommateurs. Parmi les différentes situations qui font appel au besoin d'un aménagement ou réaménagement, nous avons :

- La conception d'une nouvelle organisation ;
- L'inefficacité des opérations ;
- Le changement de conception des produits ou services ;
- Les nouveaux produits ou services ;
- Le changement de volume ;
- Le changement de modèle ;
- L'ajout des nouvelles opérations;
- La sécurité du travail;
- Les problèmes psychologiques.

Toutefois, l'aménagement d'usines doit tenir compte de divers facteurs, dont les coûts. L'aménagement d'usines génère des coûts énormes, d'où l'importance de faire un bon choix. Il arrive qu'une compagnie doit investir un capital important pour l'acquisition de nouveaux biens. Certaines compagnies, par exemple, auront à acquérir un nouvel

établissement ou un terrain, alors que d'autres n'auront qu'à se procurer des matériaux de construction ou des équipements spécialisés; parfois elles devront obtenir les deux. Cependant, l'obtention de ces biens ne doit pas constituer un enjeu négatif pour l'aménagement d'usines, car à long terme cette stratégie s'avère avantageuse. En effet, ***Francis, McGinnis et White (1992)*** mentionnent que l'aménagement d'usines a un impact positif sur la productivité générale de l'entreprise.

Le problème d'aménagement d'usines est la détermination de l'arrangement le plus efficace des départements dans une usine. L'arrangement efficace des ressources (par exemple : machines, départements, ou main d'œuvre) dans l'usine a pour effet une bonne coordination du déroulement des opérations entre les ressources. Un aménagement efficace est avantageux pour des opérations qui dépendent du flux du travail afin qu'elles soient bien performantes. Par exemple, dans une entreprise manufacturière, l'efficacité d'un aménagement se traduit par une bonne coordination du flux du matériel entre les machines, de sorte que les bonnes quantités de matériaux soient fournies aux machines au bon moment de manière à éviter l'accumulation d'inventaire causée par le processus du travail. De plus, cet aménagement efficace évite la sur-utilisation de manutention du matériel et réduit son coût. Ainsi, un bon aménagement aide une compagnie et contribue à l'efficacité globale des opérations. De plus, ***Francis, McGinnis et White (1992)*** mentionnent que d'autres objectifs peuvent être considérés :

- Minimiser le temps global de production ;
- Minimiser les investissements en équipements ;

- Utiliser le plus efficacement l'espace disponible ;
- Faciliter le processus manufacturier ;
- Favoriser l'utilisation efficace de la main d'œuvre ;
- Maintenir la flexibilité de l'arrangement et de l'opération ;
- Fournir aux employés un environnement pratique, sécuritaire et confortable ;
- Minimiser les variations dans les types d'équipements de manutention utilisés.

Tompkins et al (1996) soulignent que le coût de manutention est la mesure qui détermine l'efficacité d'un aménagement :

"Material handling cost is the most significant measure for determining the efficiency of a layout and is most often considered, since it represents 20 to 50% of the total operating cost and ***15 to 70%*** *of the total cost of manufacturing a product."*

1.3 CLASSIFICATION DE PROBLEMES D'AMENAGEMENT

Dans un sens, le problème d'aménagement d'usines peut être considéré comme une classe spéciale de problème de localisation de service. Le problème d'aménagement d'usines comprend la détermination de l'endroit, par exemple des bureaux, aussi bien qu'une détermination de la taille et de la configuration des bureaux.

En classifiant des problèmes de localisation de service, six (6) éléments principaux doivent être considérés: de nouvelles caractéristiques de service, endroits existants de service, nouvelles et existantes interactions de service, les caractéristiques de l'espace de solution, la mesure de distance, et la fonction objective. Ces éléments sont dépeints sur les ***figures 1 à 10*** adaptées de ***(Richard L. Francis John A. White « facility layout and location an analytical approach »)***

Les problèmes de localisation de département peuvent être classifiés selon le nombre d'équipements impliqués. En outre, les nouveaux équipements peuvent occuper un point précis de l'espace ou toute la surface. Le nombre de nouveaux équipements est une variable de décision. Finalement, le nouvel endroit du nouveau département peut être dépendant ou indépendant de la localisation des nouveaux équipements restants.

Si la surface est considérée pour les nouveaux et les anciens équipements, le problème de localisation est souvent classifié comme un problème d'aménagement d'usines. Dans ce cas, la taille et la configuration de l'espace sont des variables de décision. Dans le problème d'aménagement les équipements peuvent être des bureaux, des entrepôts, des machines, etc.

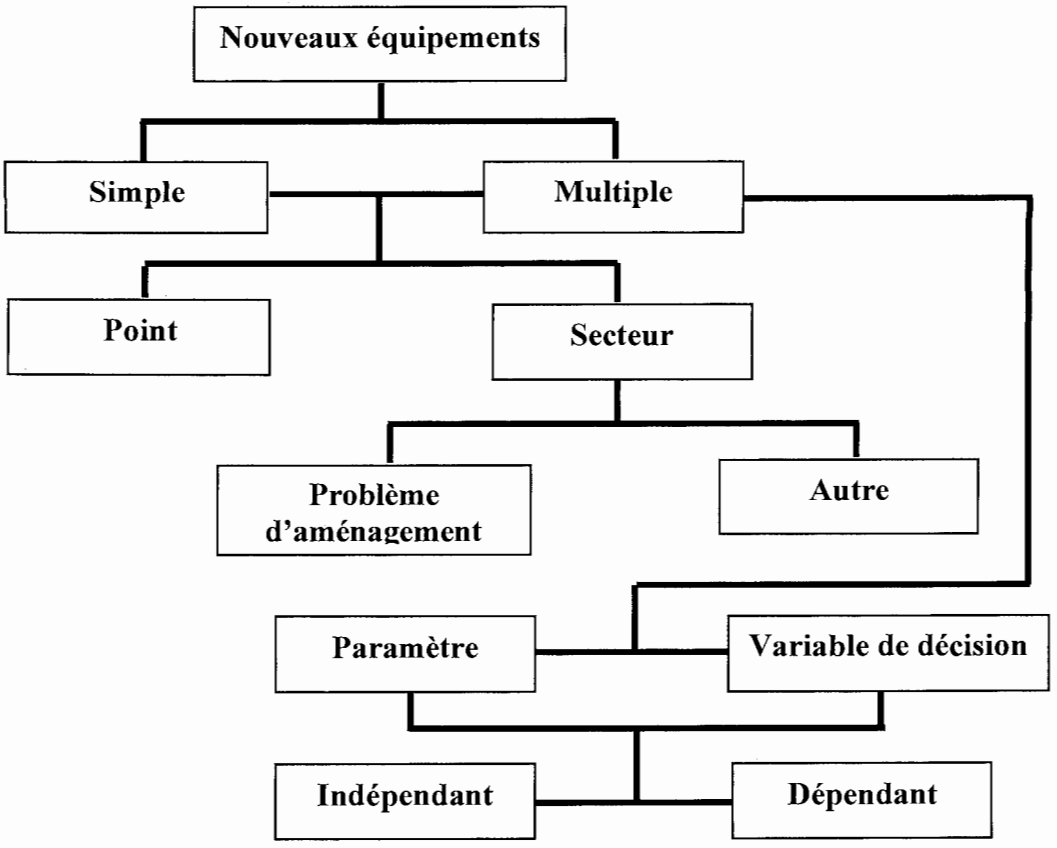


Fig.1.1- Classification du problème d'aménagement (nouvelles caractéristiques de service)

La localisation d'un département existant peut être statique ou dynamique, aussi bien que déterministe ou probabiliste. En plus, dépendant des tailles des départements existant, il peut être considéré comme ayant un point de localisation ou toute la surface. Une classe spéciale du problème de localisation de département impliquant l'occupation de l'espace est un problème d'aménagement. Quand les équipements existant sont inclus dans le problème d'aménagement, un problème de réaménagement peut exister. Dans ce cas, la

configuration et la position des équipements existant doivent être considérées. Un problème d'aménagement ou de re-localisation intervient quand la position d'un département existant est une variable de décision.

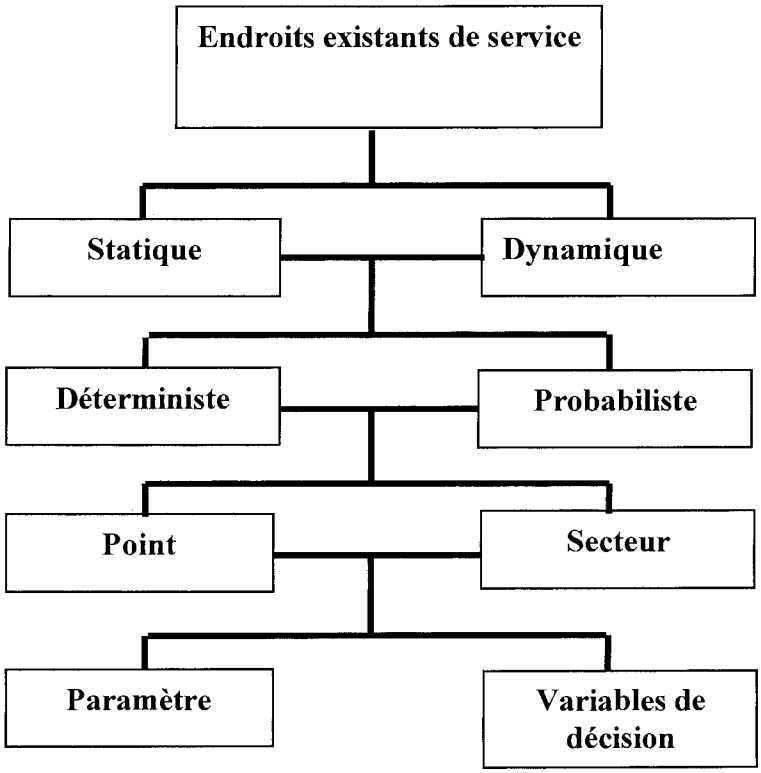


Fig.1.2 - Classification du problème d'aménagement (endroits existants de service)

Lorsqu'il existe une relation quantitative ou qualitative entre les équipements, on considère que les équipements interagissent. Dans certain cas, le degré d'interaction est une fonction de la position des équipements. Il est vrai qu'il y a un nombre de situation où ce type d'interaction est indépendant des localisations des départements. En outre, l'étendue

de l'interaction peut être soit statique ou dynamique, soit déterministe ou probabiliste, et soit un paramètre ou une variable de décision.

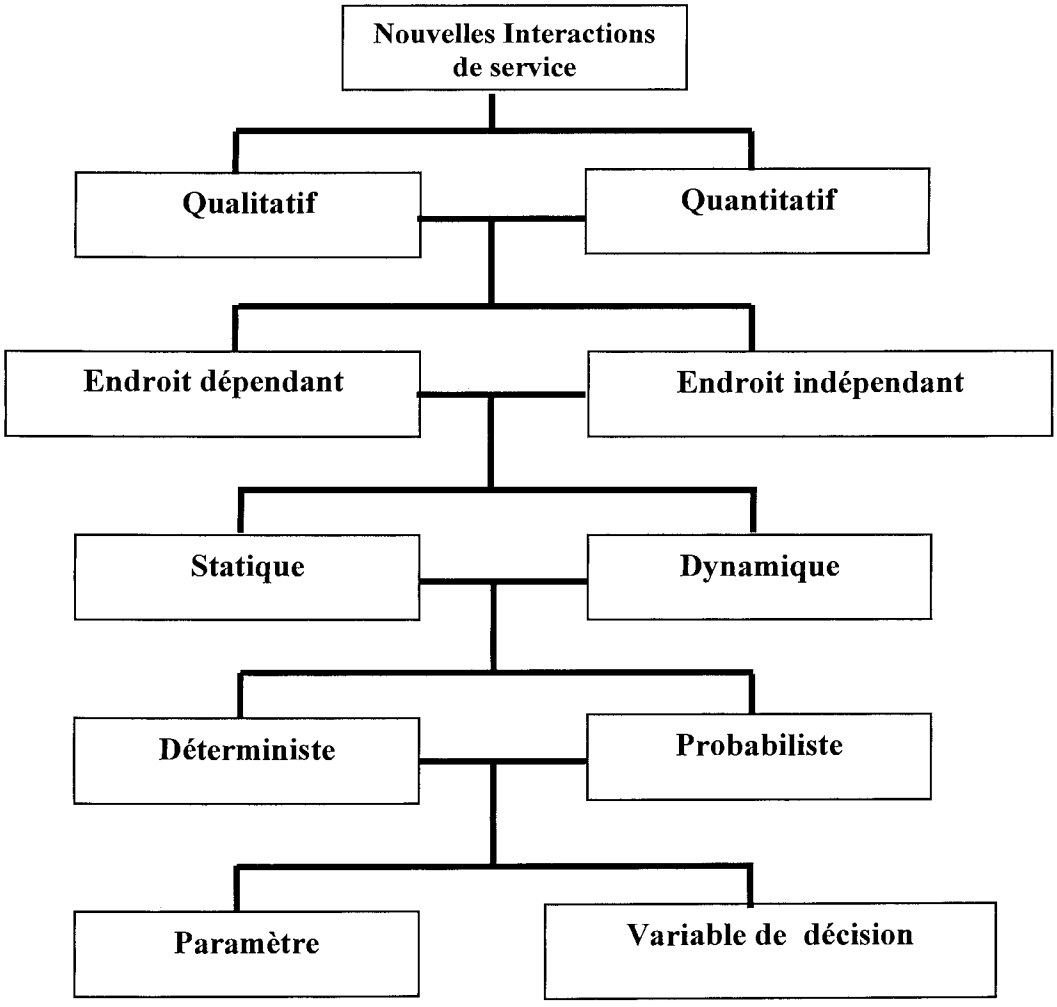


***Fig.1.3* - Classification du problème d'aménagement (nouvelles interactions et interactions existantes de service.)**

La quatrième catégorie considérée pour classifier le problème de localisation concerne l'espace de solution pour le problème de localisation. Dans un certain problème de localisation de département, l'espace de solution est dimensionnel; cependant, il existe

généralement un espace de solution à deux ou trois dimensions. Aussi, l'espace de solution peut être soit discret ou continu. L'espace de solution discret consiste en un nombre fini de localisations possible. Tandis que l'espace de solution continue est un nombre infini de localisations possible. Dans un autre cas, l'espace de solution peut être limité par une ou plusieurs contraintes. L'aménagement discret implique des modules discrets, tels que les compartiments dans un entrepôt, tandis que l'aménagement continu considère les modules comme des points.

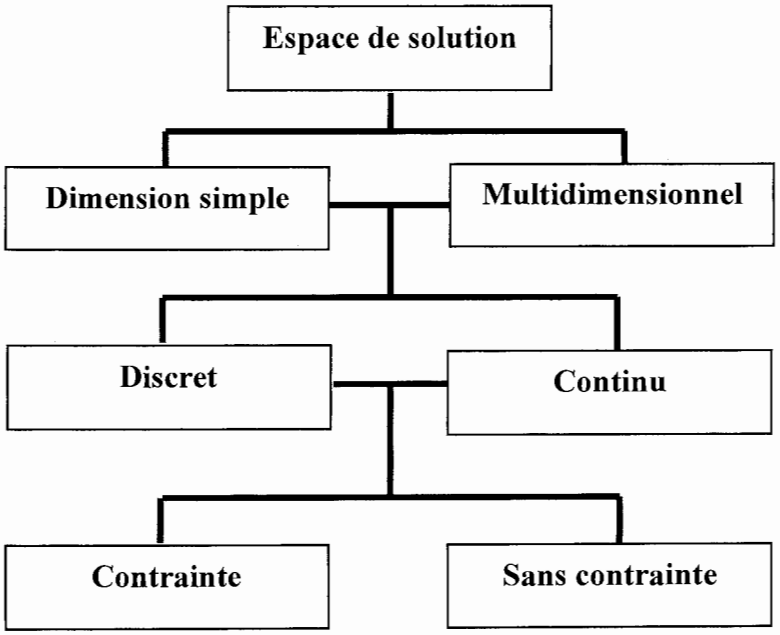


Fig.1.4 - Classification du problème d'aménagement (Caractéristiques de l'espace de solution.)

La mesure de distance impliquée dans les problèmes de localisation fournit une autre classification. Les distances euclidienne et rectiligne sont souvent utilisées. Cependant, il existe un nombre de problèmes de localisation de département dont les distances entre les départements ne peuvent pas être raisonnablement représenter. C'est le cas d'un problème de localisation urbain dans lequel les rues ne forment pas une grille rectangulaire.

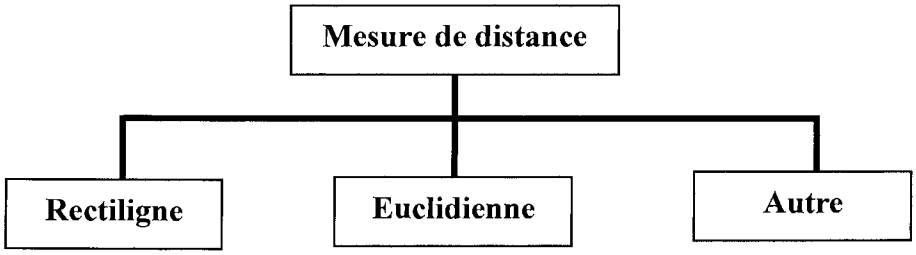


***Fig.1.5** - Classification du problème d'aménagement (mesure de distance)*

La dernière catégorie, souvent utilisée pour classifier le problème de la localisation, est la fonction objective employée pour évaluer des solutions alternatives. Dans le cas de problème d'aménagement, un nombre d'objectifs qualificatifs est souvent utilisé, tandis que les objectifs quantitatifs sont utilisés pour résoudre le problème de localisation.

Les objectifs quantitatifs rencontrés dans les problèmes de localisation sont la minimisation de la somme totale de la fonction coût ou la minimisation de coût entre deux équipements. On peut rencontrer d'autres objectifs utilisés dans la littérature. Toutefois,

quelques problèmes d'aménagement ne pourraient pas être complètement classifiés en utilisant les catégories que nous avons suggérées.

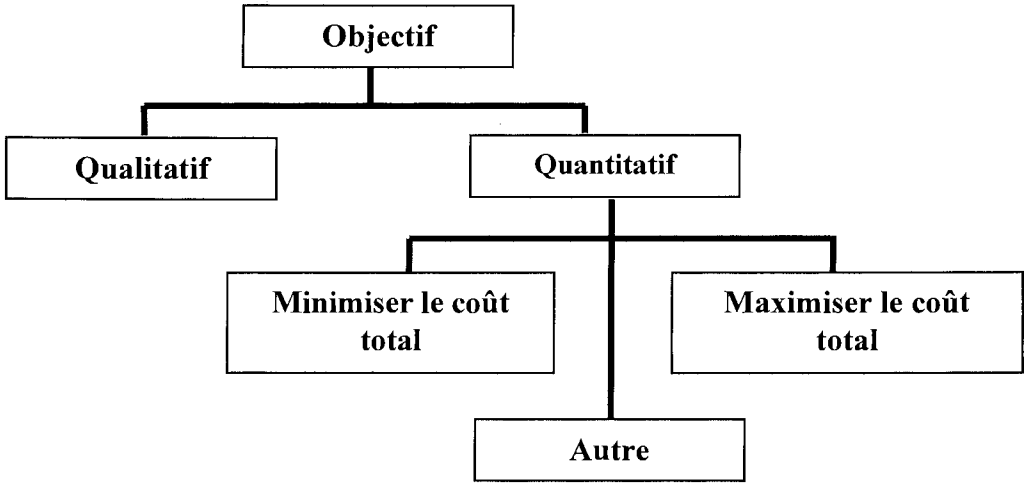


Fig.1. 6 - Classification du problème d'aménagement (Objectif)

1.4 LES PARAMÈTRES DE L'AMÉNAGEMENT

Dans un aménagement d'usines, il existe un ensemble de paramètres à déterminer tel que l'interaction entre les centres (le flux) et la distance entre deux centres.

1.4.1 Calcul des distances

Géométriquement, un centroïde peut être défini comme étant le centre de gravité, ou centre de la masse d'un objet. Dans le domaine de l'aménagement d'usines, le centroïde d'un département est le centre du département. En outre, les distances rectilinéaires et euclidiennes sont généralement, les mesures le plus utilisées (***Figure.1.7***).

La distance euclidienne correspond à une distance à vol d'oiseau entre deux points. Elle est définie par la formule suivante :

$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2} \qquad \textit{Equ. 1.1}$$

La distance rectilinéaire entre deux points quelconques $(x_1, y_1) et (x_2, y_2)$ correspond à la somme des déplacements selon chacun des axes cartésiens. Elle se calcule par la formule suivante:

$$d = |x_2 - x_1| + |y_2 - y_1| \qquad \textit{Equ. 1.2.}$$

La distance rectiligne entre deux centres est souvent la plus utilisée dans le cas de l'aménagement d'usines.

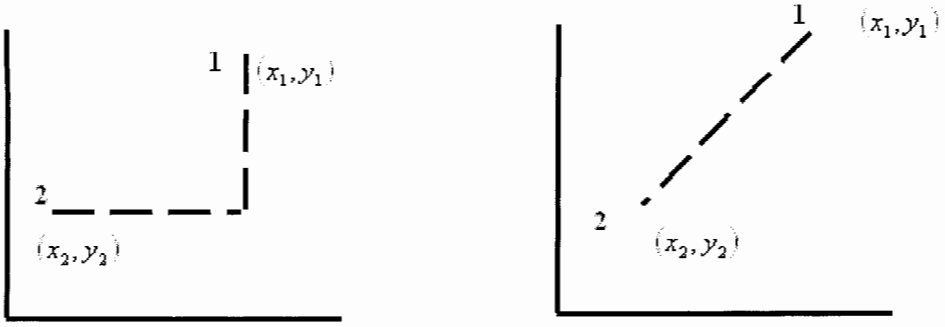


Fig.1.7 - Distances euclidiennes et rectilinéaires

1.4.2 Le flux et le modèle d'acheminement

Il existe plusieurs types d'interactions entre les centres de départements dont la matrice de flux. Cette matrice se présente généralement sous l'une des deux formes suivantes :

Des flux directionnels: ***n*** unités se déplacent du centre ***i*** vers le centre ***j***

Des flux non directionnels: ***n*** unités circulent entre les centres ***i*** et ***j***

Dans la modélisation mathématique d'un aménagement, le terme ***''analyse des flux''*** se réfère principalement aux flux des matières entre départements. Cette analyse permette de relier l'intensité de flux au positionnement relatifs des départements.

Par exemple, si le flux entre les départements ***A*** et ***D*** sont de ***500*** alors que les flux entres les départements ***A*** et ***F*** sont de ***1500***, alors les flux entre ***A*** et ***F*** sont plus importants, donc, il est préférable de placer ***F*** plus près de ***A*** que de placer ***D*** plus près de ***A***.

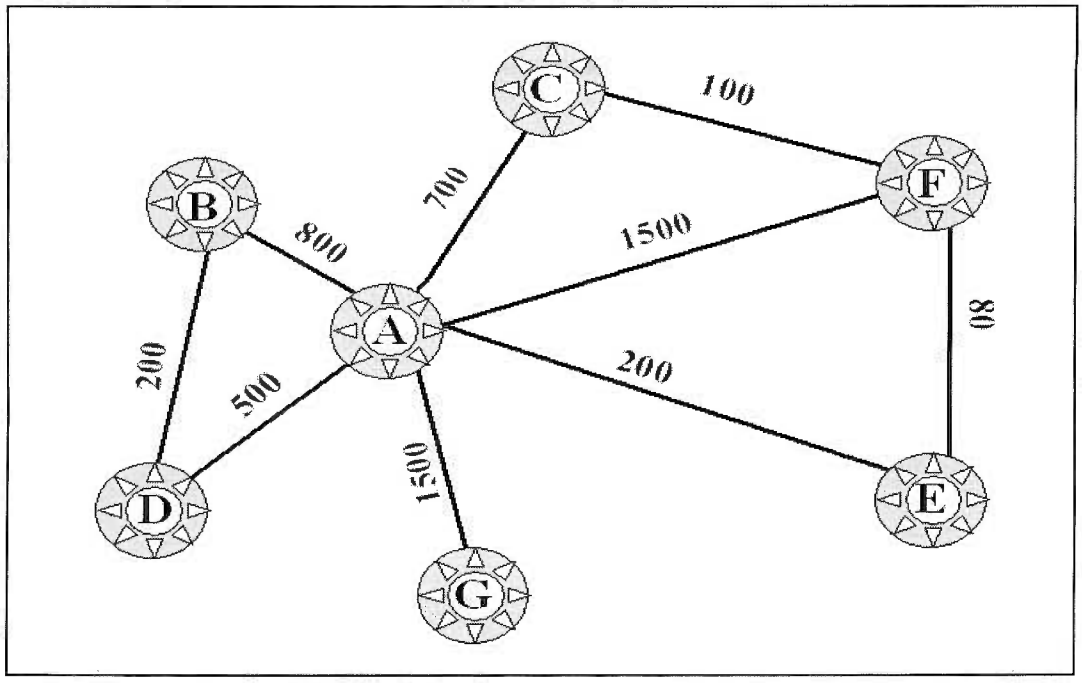


Fig.1. 8 - Distances euclidiennes et rectilinéaires

En règle générale, le modèle d'acheminement le plus simple et le plus direct sera le plus efficace. Ceci ne veut pas nécessairement dire que le déplacement doit se faire en ligne droite. Assez souvent, il faut refaire le chemin parcouru. Dans ce cas, l'acheminement des matières entraîne des coûts élevés. L'existence d'une telle situation révèle généralement que l'on parcourt des distances inutiles et il en résulte que les frais de manutention sont trop élevés.

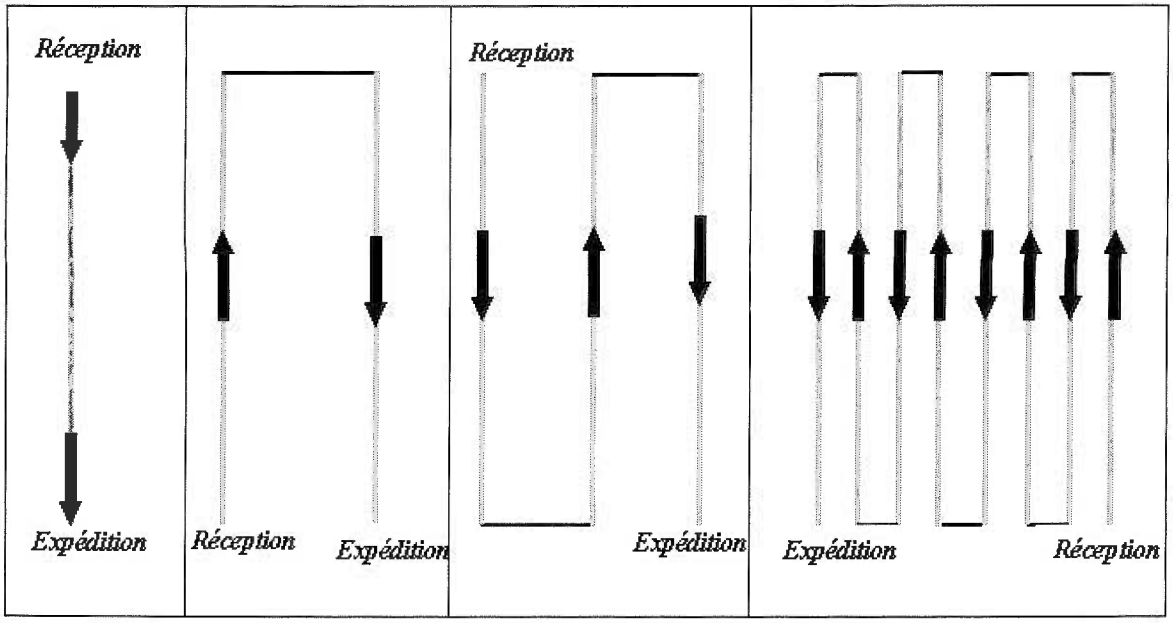


Fig.1. 9 **- Modèles d'acheminement**

Modèle en forme de I :

Dans ce cas, les matières entrent par une extrémité de l'usine et se déplacent en ligne droite. Les produits finis sont entreposés à l'autre extrémité de l'usine et expédiés de cet endroit. Au fur et à mesure que le nombre d'opérations s'accroît, l'acheminement se fera de plus en plus en zigzag, mais les matières se déplacent quand même dans la même direction.

Modèle en forme de U :

Selon ce modèle, le service de l'expédition et celui de la réception sont situés à la même extrémité de l'usine. Les deux services peuvent être l'un près de l'autre et peuvent

utiliser la même zone de chargement. Les produits en cours vont d'une extrémité à l'autre de l'usine puis reviennent au point de départ. Ce modèle peut être en usage dans toute usine, quelle que soit sa taille. Il convient, toutefois, davantage aux petites usines et à leur plan d'aménagement. En effet, les fonctions réception et expédition peuvent être sous la supervision d'une même personne, ce qui est souvent indispensable dans les petites usines.

Forme convolutée :

Les usines dont l'aménagement est complexe doivent avoir de nombreux ateliers. L'acheminement des matières se fait selon une forme convolutée (il y a ainsi dire un mouvement de va et vient d'une extrémité a l'autre de l'usine), ce qui permet le déplacement en ligne droite à l'intérieur d'un espace restreint. Souvent, le modèle en forme ***" I"*** à tendance à se modifier et à devenir un modèle convoluté. Dans ce cas, le service de la réception et celui de l'expédition peuvent être situés à la même extrémité de l'usine ou à l'opposé l'un de l'autre.

1.5 LE PROBLÈME D'AMÉNAGEMENT STATIQUE

Le problème d'aménagement statique consiste à déterminer l'arrangement le plus efficace des départements à l'intérieur de l'usine lorsque le flux entre les départements reste constant durant la planification horizontale.

Cette recherche se concentre sur la représentation d'un aménagement discret qui consiste à diviser l'usine avec des cases pré-localisées de tailles prédéterminées. La discrétisation peut prendre la forme d'une grille de cases carrées de dimension égales formant les distances unitaires.

1.5.1 Modélisation comme un problème d'affectation quadratique

Le problème d'affectation quadratique *(PAQ)* ou bien ''*Quadratic Assignment Problem* (QAP)'' a été formulé pour la première fois en 1957 par *KOOPMANS et BECKMANN*. Dès lors, il s'est révélé comme ayant de très nombreuses applications pratiques étant les plus intéressantes. Entre autres, le placement de circuits électroniques et la réparation des services médicaux dans les grands centres hospitaliers.

Cependant, étant particulièrement difficile à résoudre, le problème d'affectation quadratique a fait l'objet de diverses publications. En effet, jusqu'à ce jour, les méthodes exactes '' l'algorithme par séparation et évaluation ***branch-and-bround''*** les plus performantes ne permettent pas de résoudre la plupart des problèmes de taille supérieure à 30. Ainsi, des applications pratiques, comme celle du placement des circuits électroniques, requièrent souvent de placer plus d'une centaine d'éléments. D'ailleurs, c'est pour cette raison que de nombreuses méthodes approchées furent adaptées pour résoudre le problème d'affectation quadratique. Parmi ces adaptations, on trouve des méta-heuristiques, telles

que l'algorithme à colonie de fourmis, le recuit simulé, les algorithmes génétiques et la recherche avec tabou.

Dans ce projet, afin de résoudre le problème d'affectation quadratique, la présentation de la conception d'un algorithme de résolution basé sur une hybridation de deux méta-heuristiques sera élaborée : l'algorithme à colonies de fourmis et une variante du recuit simulé (algorithme à plafond dégradé).

1.5.2 Présentation du problème d'affectation quadratique

Le problème d'affectation quadratique est un problème classique d'optimisation combinatoire dans lequel il convient de trouver le placement optimal de ***n*** objets ***''usines, matériels''*** sur ***n*** locations, tout en minimisant le coût qui dépend à la fois des distances inter locations et des flux inter objets. Il s'agit de trouver l'affectation qui permettra de minimiser le coût total.

Dans l'exemple qui suit (figure1.9) on dispose de six zones et d'un ensemble six objets. Le but est d'affecter l'ensemble des objets à l'ensemble des zones tout en minimisant la fonction objective qui présente le coût total engendré.

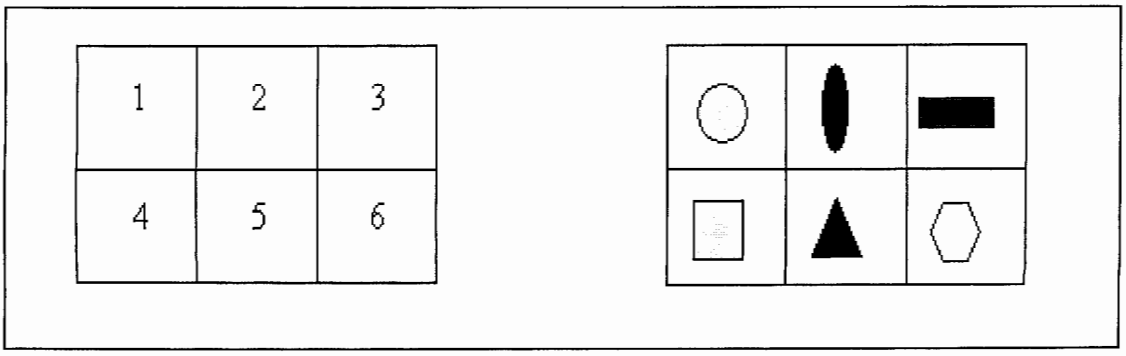


***Fig.1.10** - Exemple d'emplacements et affectation des équipements aux emplacements*

Soient :

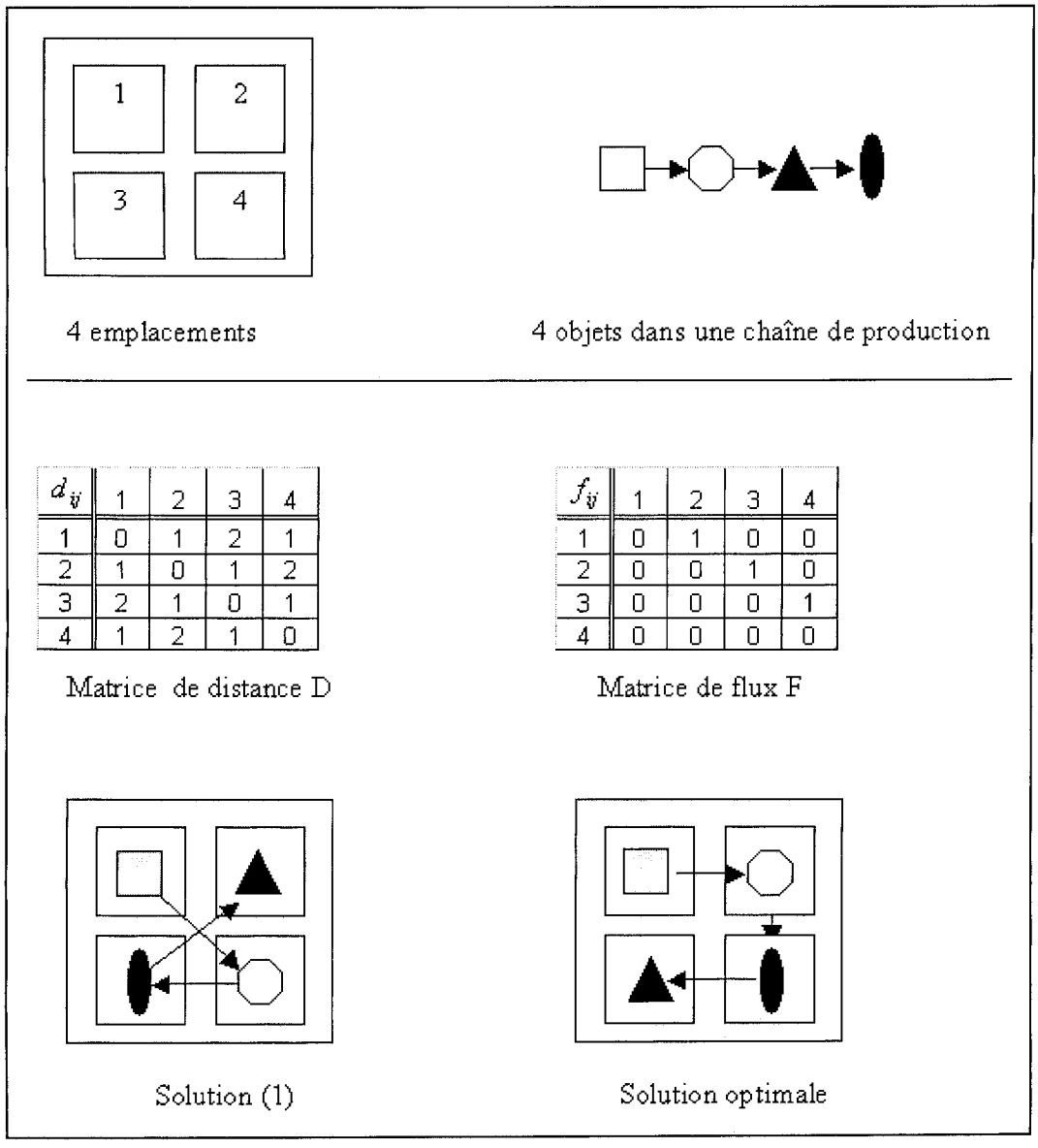
- $D(n \times n)$ La matrice telle que $d(i,j)$ représente la distance entre la location ***i*** et la location ***j.***
- $F(n \times n)$ La matrice de dimension $(n \times n)$telle que $F(k,l)$ représente le flux entre les objets ***k*** et ***l.***

Le problème peut alors s'écrire sous la forme suivante :

La fonction de objectif :

$$Min \quad L = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} f(i,j) \times d(p(i), p(j))$$

Le problème d'aménagement d'usine est formulé pour trouver l'arrangement le plus efficace des départements comme expliqués dans les schémas suivants :



d_{ij}	1	2	3	4
1	0	1	2	1
2	1	0	1	2
3	2	1	0	1
4	1	2	1	0

f_{ij}	1	2	3	4
1	0	1	0	0
2	0	0	1	0
3	0	0	0	1
4	0	0	0	0

Fig.1.11.- Un exemple simple de l'affectation quadratique

Le coût associé à la solution *(1)* est :

$$L = 2+1+2 = 5 \qquad \textit{Equ. 1.4}$$

Alors que le coût de la solution *(2)* est :

$$L = 1+1+1 = 3 \qquad \textit{Equ. 1.5}$$

Ceci démontre que la solution *(2)* est l'une des solutions optimales et alors le meilleur arrangement.

1.5.3 Applications principales

Le problème d'affectation quadratique a de très nombreuses applications en pratique, tant en informatique, qu'en productique électronique ou architecture, mais il est très délicat à résoudre.

Les applications :

- **Les services hospitaliers :**

  Le problème a surgi en 1972 en tant qu'élément de la conception d'un hôpital d'université allemand, *Klinikum Ratisbonne*. Ce problème, connu sous le nom de Krarup30a, où 30 équipements doivent être assignés à 30 emplacements.

❖ **Domaine de l'électronique :**

Dans le domaine de l'électronique, le problème (***PAQ***) est appliqué pour optimiser le placement et le routage des circuits électroniques. Figures (1.12, 1.13)

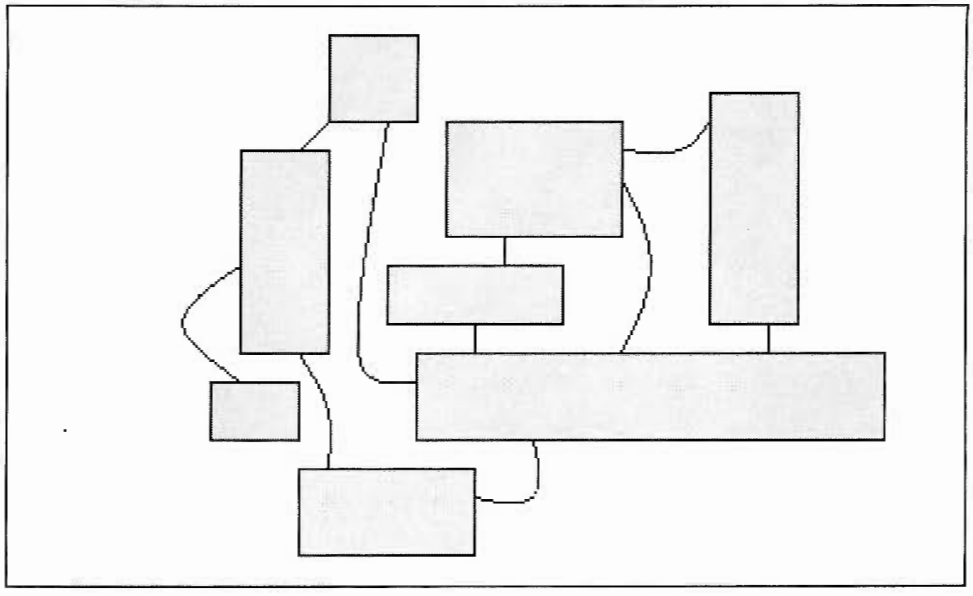


Fig.1.12 - Emplacement des composantes en désordre

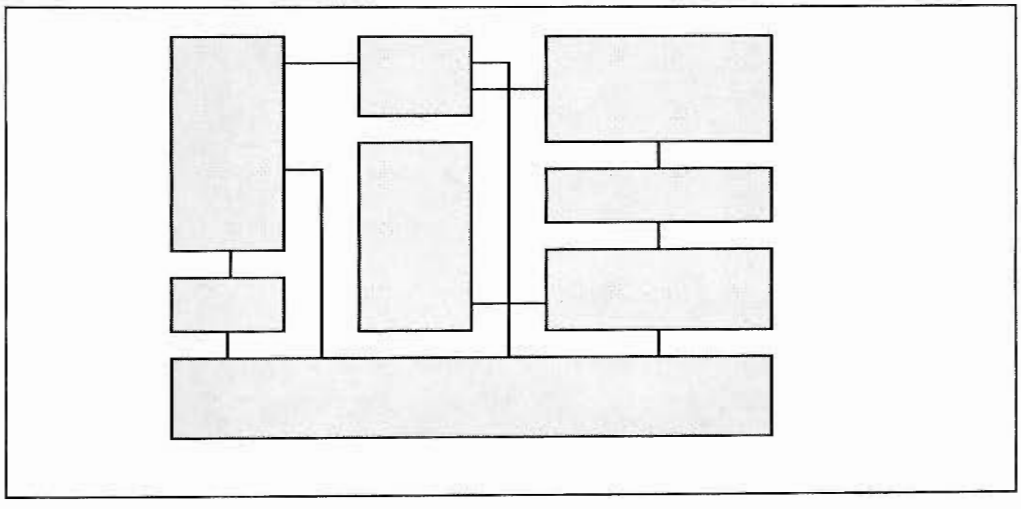


Fig1.13 - Emplacement des composantes en ordre par le (PAQ)

1.6 CONCLUSION

L'optimisation combinatoire occupe une place très importante en recherche opérationnelle, en mathématiques discrètes et en informatique. Son importance se justifie d'une part par la grande difficulté des problèmes d'optimisation et d'autre part par de nombreuses applications pratiques pouvant être formulées sous la forme d'un problème d'optimisation combinatoire (tel que le problème d'affectation quadratique). Bien que les problèmes d'optimisation combinatoire soient souvent faciles à définir, ils sont généralement difficiles à résoudre. En effet, la plupart de ces problèmes appartiennent à la classe des problèmes NP-difficiles et ne possèdent donc pas à ce jour de solution algorithmique efficace valable pour toutes les données.

Etant donnée l'importance de ces problèmes, de nombreuses méthodes de résolution ont été développées en recherche opérationnelle et en intelligence artificielle. Ces méthodes peuvent être classées sommairement en deux grandes catégories : les méthodes exactes (complètes) qui garantissent la complétude de la résolution et les méthodes approchées (incomplètes) qui perdent la complétude pour gagner en efficacité.

Dans le deuxième chapitre du mémoire, une revue de la littérature présentera brièvement les différentes approches de résolution qui ont été proposées pour le problème d'aménagement d'usines modélisé comme un problème d'affectation quadratique. En particulier, nous nous intéresserons aux méta-heuristiques.

CHAPITRE 2

REVUE DE LA LITTÉRATURE

2.1 INTRODUCTION

Le problème d'aménagement d'usines consiste à trouver l'arrangement le plus efficace de N départements indivisibles avec des conditions inégales de surface dans une usine. Comme défini dans la littérature, un tel arrangement est sujet à deux ensembles de contraintes:

- Conditions de département et de surface couverte ;
- Restrictions localisées de départements, c'est-à-dire que les départements ne peuvent pas recouvrir ; ils doivent être placés dans le service et certains doivent être fixés à un endroit où ils ne peuvent pas être placés dans des régions spécifiques.

Le problème d'aménagement d'usines a été étudié intensivement dans la littérature. Pour les aperçus les plus récents, le lecteur est référé à ***Kusiak et Heragu et Meller et Gau***. Nous rappelons que le problème d'aménagement d'usines est un problème d'optimisation combinatoire difficile, et que le problème d'affectation quadratique, où l'optimisation est reportée à un ensemble d'endroits possibles de département, est *NP-difficile* aussi. Donc, des méthodes exactes de solution sont seulement faisables pour des petits problèmes ou des problèmes considérablement limités. Pour cette raison, la plupart des méthodologies de solution dans la littérature sont basées sur des heuristiques.

En général, les problèmes de disposition de service sont trouvés dans la littérature avec l'arrangement des départements rectangulaires. Cependant, il y a des applications dans lesquelles un arrangement orthogonal des départements n'est pas nécessairement une condition (par exemple : la planification du tableau de bord d'un avion, une ville ou un voisinage, une disposition des bâtiments de bureau et les circuits intégrés). D'autre part, le problème statique de l'aménagement *''SFLP : static facility layout problem''* est un problème bien étudié et la littérature remonte à presque cinq décennies. Récemment, une emphase particulière a été remise sur l'utilisation des heuristiques pour la résolution de ce problème d'optimisation combinatoire.

D'autre part, cette littérature est aussi la base pour résoudre le problème dynamique d'aménagement d'usines qui a commencé au milieu des années 80. Ainsi, la littérature de *''SFLP''* fut d'abord publiée et quelques années plus tard, ce fut la littérature de *''DFLP''* qui fut passée en revue.

Généralement, les méthodologies pour le problème statique d'aménagement d'usines ''*SFLP*'' peuvent être classifiées en algorithmes Exacts, donnent des solutions *optimales* et algorithmes heuristiques. Des algorithmes l'algorithme par séparation et évaluation (*Branch and Bound)* et *les plants sécants* sont employés pour résoudre le *SFLP*. Les algorithmes par séparation et évaluation ont été développés et mis en application par *Gilmore (1962)*, *Lawler (1963)*, et *Kaku et Thompson (1986)*, pour ne mentionner que quelques-uns. *Bazaraa et Sherali (1980)* ont développé un algorithme utilisant des plans sécants (*Cutting plane)* et, plus tard, *Burkard et Bonninger (1983)* ont également développé des algorithmes (*Cutting plane)*. Pour un examen des algorithmes exacts pour le *''SFLP''*, voir *Kusiak et le Heragu (1987). Montreuil (1990)* a présenté une formulation de programmation de nombre entier pour le ''*SFLP''*. De même, *Heragu et Kusiak (1991)* ont présenté deux nouveaux modèles pour le ''*SFLP''* :

- Modèle continu ;
- Modèle linéaire en nombres entiers.

L'inconvénient principal de ces algorithmes exacts est qu'ils nécessitent des conditions informatiques lourdes même s'ils sont appliqués à des problèmes de petite taille.

Afin d'obtenir de bonnes solutions dans un temps raisonnable, des heuristiques ont été développées. Une heuristique peut être définie comme un ensemble d'étapes pour identifier rapidement des solutions de bonne qualité.

La qualité d'une solution est définie par un critère d'évaluation (par exemple, minimiser le coût de manutention matérielle) et la solution doit satisfaire les contraintes du problème.

Fondamentalement, les algorithmes heuristiques pour le *''SFLP''* peuvent être classifiés selon quatre classes, *Heragu, (1992)*:

- Algorithmes graphique ;
- Algorithmes constructifs ;
- Algorithmes d'amélioration ;
- Algorithmes hybrides.

La section *2.3* est consacrée aux algorithmes heuristiques, alors que la prochaine section présente des algorithmes exacts auparavant utilisées pour résoudre le problème d'aménagement d'usines.

2.2 ALGORITHMES EXACTS

Les méthodes optimales de résolution sont des approches qui permettent d'obtenir la meilleure solution à un problème selon un ensemble de critères. Les logiciels commerciaux, tel que CPLEX, contiennent un ensemble de ces méthodes de résolution.

En faisant face au problème d'aménagement d'usines, quelques algorithmes exacts ont été proposés : *Gilmore* (1962), *Lawler* (1963), *Seppanen et Moore* (1970), *Foulds* et *Robinson* (1976), *Kaku et Thompson* (1986) et *Ketcham* (1992).

Un algorithme exact peut donner une meilleure solution pour un problème donné ; il peut être divisé en deux classes :

- L'algorithme par séparation et évaluation (*Branch and bound*);
- Les plans sécants *(cutting plane).*

2.2.1 Les algorithmes par séparation et évaluation Branch and Bound

Selon le terme anglo-saxon, un algorithme par séparation et évaluation est également appelé *Branch et Bound (B&B)*. C'est une méthode générique de résolution de problèmes d'optimisation et plus particulièrement d'optimisation combinatoire ou discrète. Elle a été introduite pour la première fois par *A. H. Land et A. G. Drigo* (1960) qui l'avaient décrite comme une méthode de construction d'un arbre de décision. Elle permet de résoudre les problèmes en nombres entiers dans plusieurs logiciels commerciaux.

Dans les méthodes par séparation et évaluation, la *séparation* permet d'obtenir une méthode générique pour énumérer toutes les solutions tandis que *l'évaluation* évite l'énumération systématique de toutes les solutions.

Si tout le nombre de solutions faisables pour le problème à résoudre est petit, nous pouvons étudier chaque solution faisable individuellement et choisir l'optimum en les comparant l'un par rapport à l'autre. Cependant, dans la plupart des situations réelles, une telle méthode d'énumération totale est impraticable, car le nombre de solutions s'avère être très grand. L'algorithme *B&B* fournit une méthode pour rechercher une solution optimale en conduisant une énumération qui n'est pas une énumération totale. L'ensemble de solutions faisables est divisé en plusieurs sous-ensembles simples. Dans chaque étape, un sous-ensemble prometteur est choisi et un effort est fait pour trouver la meilleure solution pour lui. Si la solution obtenue est la meilleure jusqu'ici, alors nous mettons à jour la limite. Autrement, nous devons encore diviser ce sous-ensemble en deux sous-ensembles simples ou plus et que le même processus soit répété jusqu'à ce que la meilleure solution soit obtenue.

Un algorithme général de *B&B* peut être énoncé comme suit :

- **Étape 1**: Résoudre une relaxation (détendez les contraintes ou les variables de restrictions) du problème original. Ceci donne une limite supérieure. Si la solution est intégrale, ARRÊT.
- **Étape 2**: Créer deux nouveaux sous-problèmes en s'embranchant sur une variable partielle.
- **Étape 3**: Un sous-problème n'est plus en activité si l'un des points suivants se produit:
  - ✓ Le sous-problème a été embranché dessus ;

- ✓ Toutes les variables dans la solution sont des nombres entiers ;
- ✓ Le sous-problème n'est pas faisable ;
- ✓ Le sous-problème est sondé (terminé) par un argument de bondissement.

❖ **Étape 4** : Choisir un sous-problème actif et le relier à une variable partielle. Répéter jusqu'à ce qu'il n'y ait aucun sous-problème actif.

L'exécution de la branche et de la méthode attachée est facilitée par le calcul d'une limite dans chaque sous-ensemble de la partition produite. Ces limites sont employées en choisissant les sous-ensembles prometteurs dans la recherche pour l'optimum et pour éliminer quelques sous-ensembles qui ne peuvent probablement pas contenir la solution faisable optimale. Dans les problèmes de minimisation, des limites inférieures peuvent être employées pour décider si nous exécutons l'embranchement à un certain nœud dans l'arbre de recherche. Si nous avons une limite inférieure pour un sous-ensemble qui est supérieur ou égal à la limite supérieure courante, il n'y a pas d'avantage à l'embranchement dans ce nœud.

La réussite de la méthode de *B&B* dépend de la stratégie d'embranchement, de la stratégie de recherche et de la qualité de la limite inférieure (dans la maximisation) ou des limites supérieures (dans la minimisation) produites. Dans cette technique de recherche, un sous-ensemble choisi parmi la liste active est exploré jusqu'à ce qu'une solution faisable améliorée ou une violation de la limite inférieure soit trouvée. Un autre sous-ensemble est

choisi si la solution n'est pas faisable. Nous relions alors le nœud parent dans la branche où l'exploration n'a pas été faite. Des limites inférieures peuvent être obtenues à chaque nœud en employant une méthode heuristique. Dans la méthode de *B&B,* une recherche alternative dans laquelle le sous-ensemble avec la plus basse limite (LB) choisie pour s'embrancher est la meilleure.

Des algorithmes de *B&B* ont été développés la première fois par *Gilmore (1962) et Lawler (1963)* pour résoudre le QAP. Ces deux algorithmes affectent un service simple à un endroit simple et calculent toutes les solutions potentielles ainsi que les limites inférieures.

Différent de *Gilmore (1962)* et de *Lawler (1963); Land* (1963), *Cravett* et *Plyter* (1966) ont proposé l'algorithme par séparation et évaluation pour affecter des paires d'équipements aux paires d'endroits.

Pierce et *Crowston* (1971) ont proposé un algorithme qui procède sur la base de l'exclusion étape par étape des paires de tâches d'une solution aux problèmes basés sur le concept de la réduction carrée de forces. *Burkard* (1973) a développé un algorithme exact pour résoudre le QAP. Dans cet algorithme, une matrice est transformée en une autre matrice de sorte que la matrice résultante ait seulement les éléments non négatifs, et que dans chaque rangée et colonne, il y ait au moins un élément zéro.

Dans l'algorithme de *Branch and Bound* développé par *Bazaraa* (1975), un arrangement partiel à chaque étape est construit et une limite pour tous les

accomplissements possibles de cet arrangement partiel existant est calculée. La disposition partielle existante est alors développée en assignant un nouveau service à condition que la tâche choisie ait un meilleur avantage comparé à la limite courante. Si la tâche a pour conséquence un avantage inférieur, une autre recherche est considérée. Si c'est le cas, une nouvelle tâche différente doit être trouvée. Le processus est suivi jusqu'à ce que l'arrangement complet soit obtenu. *Bazzara* et *Elshafei* (1979) ont proposé l'algorithme par séparation et évaluation *(Branch and Bound)* basé sur l'attribution étape par étape des équipements simples aux endroits inoccupés pour le QAP.

2.2.2 Les plans sécants

Il y a une alternative de l'algorithme par séparation et évaluation nommée les *plans sécants* qui peut également être employée pour résoudre des programmes de nombre entier. L'idée fondamentale derrière ces plans sécants est d'ajouter des contraintes à un programme linéaire jusqu'à ce que la solution faisable de base optimale prenne des valeurs de nombre entier. Naturellement, nous devons faire attention que les contraintes ajoutées ne changent pas le problème. Nous ajouterons un type spécial de contrainte appelé *coupe*.

Une coupe, relativement à une solution partielle courante, satisfait aux critères suivants:

- Chaque solution faisable de nombre entier est faisable pour la coupe ;
- La solution partielle courante n'est pas faisable pour la coupe.

Ceci est illustré sur le schéma qui suit :

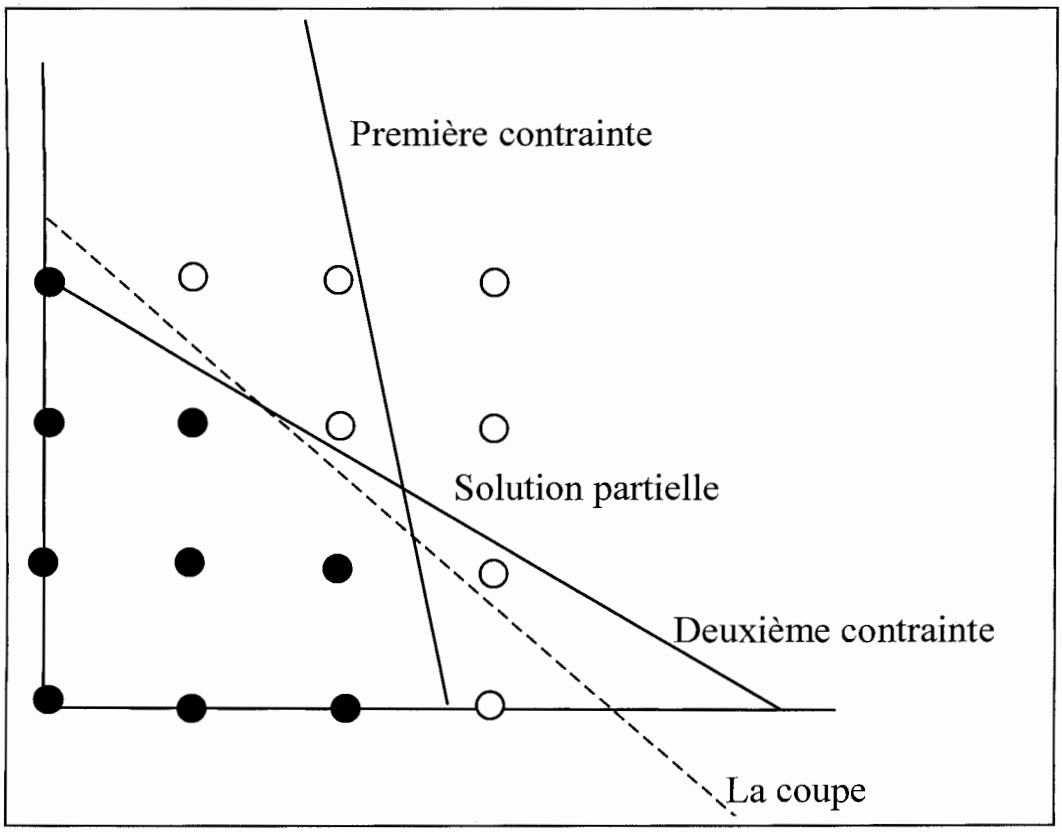


Fig. 2. 1- Une coupe

Il y a deux manières de produire des coupes. La première, appelée les coupes de *Gomory*, produit des coupes de n'importe quel tableau de programmation linéaire. Ceci a l'avantage de « résoudre » n'importe lequel problème, mais a l'inconvénient que cette méthode peut être très lente. La deuxième approche est d'employer la structure du problème pour produire des coupes très bonnes. L'approche a besoin d'une analyse de problème par problème, mais peut fournir des techniques de solutions très efficaces.

L'algorithme plans sécants *(Cutting plane) (CP)* résout des programmes linéaires de nombres entiers en modifiant les solutions de programmation des problèmes détendus,

jusqu'à ce qu'une solution de nombre entier soit obtenue. Cet algorithme raffine un programme linéaire simple en ajoutant de nouvelles contraintes successivement de sorte que la région faisable soit réduite, et ce, jusqu'à ce qu'une solution optimale de nombre entier soit trouvée.

Basés sur la technique de division, *Bazaraa* et *Sherali* (1990) ont proposé un algorithme plan sécant (*Cutting plane)*. En employant cette technique, un programme mélangé de nombres entiers est décomposé en ses nombres entiers et des parties continues, dont chacune peut être résolue par un algorithme spécialisé approprié dans la programmation mathématique. L'algorithme *Cutting plane* est également employé pour résoudre le QAP (*Bukard* et *Bonninger*, 1983). *Kusiak* et *Herugu* (1987) ont indiqué que la méthode plans sécants est complexe et qu'elle a un grand temps de résolution. Par exemple, le plus grand aménagement résolu en employant la méthode *Cutting plan* est le problème d'aménagement avec huit (8) équipements.

2.2.3 Algorithmes graphiques

Seppanen et Moore (1970) ont présenté les concepts de la théorie de graphes appliqués à la conception d'aménagement. *Hassan* et *Hogg* (1989) décrivent l'approche de graphique dans trois étapes: Dans la première étape, un graphique de contiguïté (également appelé un graphique planaire maximal) est développé à partir des rapports entre les

départements; le graphique duel des rapports départementaux est construit dans la deuxième étape; et la troisième étape convertit le graphique duel en disposition de bloc (*Hassan et le Hogg*, 1987). Des algorithmes de la théorie de graphes ont été développés dans les articles de *Seppannen et Moore* (1975), *Foulds et Robinson* (1976), *Foulds et Robinson* (1978), *Carrie et autres* (1978), *Montreuil et autres* (1987), *Goetschalckx* (1992), et *Eades et autres* (1982). Un examen des algorithmes de la théorie de graphes est présenté dans *Hassan et Hogg* (1987). Le procédé de graphique ne garantit pas que les départements ayant des liens forts seront adjacents (*Hassan et Hogg*, 1987). Ajouté à la limitation ci-dessus, il peut produire des départements irréguliers de forme.

2.2.4 Algorithmes de construction

Un algorithme de construction se compose du choix et du placement successifs des départements jusqu'à ce qu'une conception de disposition soit réalisée. Certains des algorithmes de construction sont : HC66 (*Hillier et Connors*, 1966), ALDEP (*Seehof et Evans*, 1967), CORELAP (*Lee et Moore*, 1967), RMA Comp I (*Muther et McPherson,* 1970), MAT (*Edwards et autres*, 1970), PLANETE (*Deisenroth et Apple*, 1972), LSP (*Zoller et Adendorff*, 1972), algorithme linéaire (*Neghabat,* 1974), FATE (*Block*, 1978), INLAYT (*O'brien et Abdel Barr*, 1980), SHAPE (*Hassan et autres*, 1986), NLT (*Van Camp et autres*, 1991), et QLAARP (*Banerjee et autres*, 1992). Ces algorithmes peuvent être employés pour fournir les solutions initiales pour des algorithmes d'amélioration.

2.3 LES DIFFÉRENTS TYPES DE MÉTAHEURISTIQUES

2.3.1 Les approches de recherche locale

Les méthodes de recherche locale sont des algorithmes itératifs qui explorent l'espace ''X'' en se déplaçant pas à pas d'une solution à une autre. Une méthode de ce type débute à partir d'une solution '' $s_0 \in X$ '' choisie arbitrairement. Le passage d'une solution admissible à une autre se fait sur la base d'un ensemble de modifications élémentaires '' Δ ''qu'il s'agit de définir de cas en cas. Une solution '' s' '' obtenue à partir de '' s '' en appliquant une modification '' $\delta \in \Delta$ '' est noté '' $s' = s \oplus \delta$ ''. Le voisinage '' $N(s)$ '' d'une solution '' $s \in X$ '' est défini comme l'ensemble des solutions admissibles atteignables depuis '' s '' en effectuant des modifications élémentaires '' δ ''.

On peut écrire la relation suivante :

$$N(s) = \{s' \in X \mid \exists \delta \in \Delta : s' = s \oplus \delta\} \qquad \textit{Equ. 2.1}$$

Un tel processus d'exploration est interrompu lorsqu'un ou plusieurs critères d'arrêt sont satisfaits.

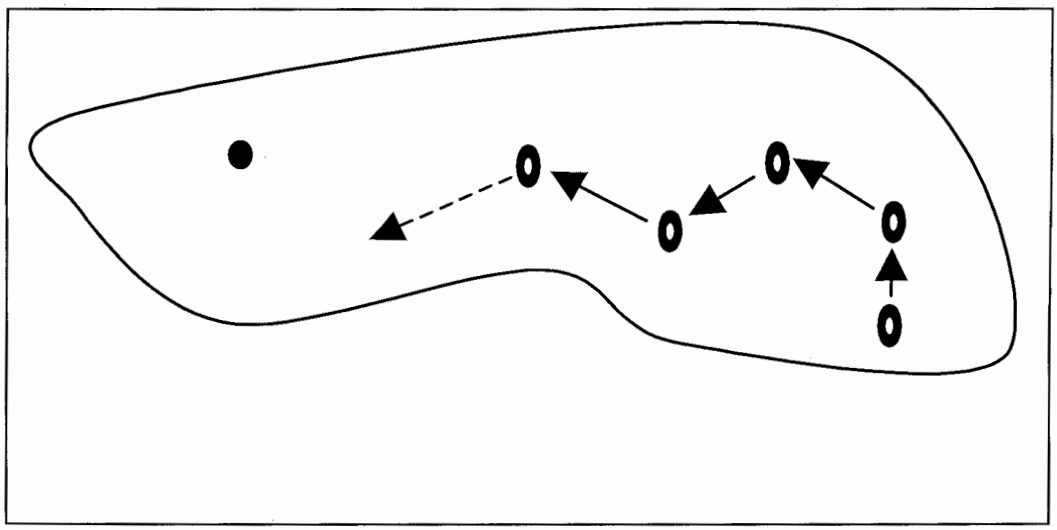


Fig.2.2 - Exploration de X par une approche de recherche locale

Des méthodes de recherche locale plus sophistiquées ont été développées au cours de ces vingt dernières années. Ces méthodes acceptent des solutions voisines moins bonnes que la solution courante afin d'échapper aux minima locaux de la fonction. En règle générale, seule une portion du voisinage courant est explorée à chaque étape. Les méthodes les plus connues seront présentées dans les sections suivantes.

2.3.1.1 Le recuit simulé

Le recuit simulé est une méta-heuristique inspirée d'un processus utilisé en métallurgie. Ce processus alterne des cycles de refroidissement lent et de réchauffage (recuit) qui tendent à minimiser l'énergie du matériau. Elle est aujourd'hui utilisée en

optimisation pour trouver les extrema d'une fonction. Elle a été mise au point par trois chercheurs de la société IBM *S. Kirkpatrick*, C.D. *Gelatt* et *M.P. Vecchi,* en 1983, et indépendamment par *V. Cerny* en 1985.

Afin de transposer les propriétés physiques aux problèmes d'optimisation, on établit l'analogie présentée dans le ***Tableau 2.1.***

Problème d'optimisation	Système physique
Fonction objective	**Énergie libre**
Paramètres du problème	**Coordonnées des particules**
La configuration optimale	**Trouver les états de basse énergie**

Tableau 2. 1- Analogie entre un problème d'optimisation et un système physique

L'algorithme du recuit simulé s'appuie sur deux résultats de la physique statistique. D'une part, lorsque l'équilibre thermodynamique est atteint à une température donnée (T), la probabilité pour un système physique de posséder une énergie donnée (E) est proportionnelle au facteur de Boltzmann : $e^{\frac{-E}{k_B T}}$.

D'autre part, pour simuler l'évolution d'un système physique vers son équilibre thermodynamique à une température donnée (T), on peut utiliser l'algorithme de ***Metropolis*** partant d'une configuration donnée (dans notre cas un placement), on fait subir au système une modification élémentaire (échange deux composants). Si cette transformation a pour effet de diminuer la fonction objective (ou énergie) du système,

elle est acceptée; si elle provoque, au contraire, une augmentation ΔE de la fonction objective, elle est tout de même acceptée, mais avec la probabilité de $e^{\frac{-\Delta E}{T}}$ (en pratique cette condition est réalisée de la manière suivante : on tire au hasard un nombre réel compris entre 0 et 1, et on accepte la configuration dégradant de ΔE la fonction objective, si le nombre tiré est inférieur ou égal à $e^{\frac{-\Delta E}{T}}$). L'itération se poursuit tant que l'énergie du système diminue. Lorsque l'énergie reste stationnaire, on diminue un peu la température et l'on reprend le processus de décroissance de l'énergie. On arrête lorsque les diminutions de température restent inefficaces.

Dans la recherche locale, lorsque nous cherchons une solution et nous restons emprisonné dans une région probablement inférieure, il n'y a aucune manière de sortir de cette région et de rechercher d'autres solutions dans différentes régions.

La méthode du recuit simulé (SA) est employée pour résoudre le problème d'aménagement d'usines. On commence par une solution initiale et on fait une recherche locale pour trouver une solution améliorée. *SA* peut rechercher d'autres solutions dans différentes régions, lorsqu'une recherche locale est emprisonnée dans une région probablement inférieure ou quand il n'y a aucune manière de s'échapper de cette région en laissant le procédé se déplacer à une meilleure solution. Dans l'algorithme de recuit simulé, si la nouvelle valeur de la fonction objective est meilleure que celle de la précédente, la nouvelle solution remplace la solution courante. Quand la nouvelle solution

a une valeur non meilleure que la précédente, la nouvelle solution peut être acceptée selon une certaine probabilité.

Algorithme :

Initialisation

Choisir une solution initiale $s \in X$;

$s^* := 0$;

$K := 0$; (compteur d'itérations globales)

nouveau_cycle := *vrai* ; (variable booléenne indiquant s'il vaut la peine d'effectuer un nouveau cycle d'itérations)

$t := t_0$; (t_0 = température initiale du système)

Processus itératif

tant que *nouveau_cycle* = *vrai* **faire**

nbiter : = 0 ; (compteur il" itérations **interne** à **un cycle**)

nouveau_cycle := *faux* ;

tant que *(nbiter* < *nbiter_cycle)* **faire**

$k := k + 1$; *nbiter* : = *nbiter* + 1 ;

générer une solution $s' \in N(s)$ *aléatoirement* ;

$\Delta f := f(s') - f(s)$;

si $\Delta f \langle 0$ **alors** $s := s'$; *nouveau_cycle* : = *vrai* ;

sinon

$prob(\Delta f, t) := \exp(-\Delta f / t)$;

générer q uniformément dans l'intervalle $[0\ 1[$;

si $q \langle prob(\Delta f, t)$ **alors** s : = s' ; *nouveau_cycle* := *vrai* ;

si $f(s) \langle f(s^*)$ ***alors*** $s^* := s$;

$t := a.t$ $(0 \langle a \langle 1)$: **Coefficient de refroidissement**

Fig.2.3 - Algorithme de recuit simulé

La description de la technique de *SA*, qui est dérivée de la mécanique statistique, peut être indiquée dans l'article de *Johnson et autres* (1989). *SA* a été exploré successivement par *Kirkpatrick et autres* (1983), et *Golden et Skiscim* (1986). L'application de *SA* pour résoudre le problème d'affectation quadratique (QAP) a été lancée par *Burkard et Rendl* (1984). *Wilhelm et Ward* (1987) ont employé *SA* pour résoudre le QAP (les problèmes d'essai de *Nugent et AI* (1968)).

Ces dernières années, il y a un certain nombre d'algorithmes basés sur le recuit simulé pour résoudre des problèmes d'aménagement d'usines. Ceux-ci incluent des algorithmes conçus par *Kouvclis et Kiran* (1991), *Heragu et Alfa* (1992), *Kouvelis et autres* (1992), *Kouvelis et Chiang* (1992), *Suresh et Sahu* (1993), et *Meller et Bozer* (1996).

Une description du recuit simulé (théorie et méthodologie) pour le problème de conception d'aménagement d'usines peut également être trouvée en *Souilah* (1995). Dans sa description, la méthodologie de *SA* générale pour la conception de système de fabrication est discutée avec son application en cellule de production concevant la disposition d'intra-cellule et plaçant les cellules de production sur la surface disponible de faire des emplettes-plancher. Quelques exemples numériques sont présentés sans résultats informatiques. *Heragu et Alfa* (1992) ont proposé des algorithmes de *SA* pour résoudre le problème de phase de conception de disposition de service qu'ils ont considéré l'échange de deux (2) ou trois (3) équipements entre les positions des équipements. Si l'échange

entre la position des équipements a comme conséquence une solution avec une valeur objective plus basse de fonction, alors l'échange est fait.

2.3.1.2 La recherche avec tabous

Une autre approche heuristique disponible pour résoudre le problème d'aménagement d'usine est la recherche de Tabou (TS). Cette approche est une métaheuristique pour résoudre des problèmes d'optimisation combinatoire. Cette approche peut être superposée à d'autres algorithmes, pour éviter d'être emprisonnée à une solution optimale locale.

La recherche avec tabou (TS pour *Tabu Search* en anglais) commence par une solution initiale. Pour produire une liste de solutions candidates, quelques échanges heuristiques locaux sont mis en application. S'il y a beaucoup de solutions candidates disponibles, une restriction de la recherche peut être employé pour créer un sous-ensemble de la solution candidate. Puis, ce sous-ensemble des solutions est évalué, suivi par la sélection de la meilleure solution à partir de l'ensemble de candidat de solutions, qui a la valeur maximum (dans la maximisation) ou la valeur minimum (dans la minimisation). La meilleure solution suivante est choisie si cette solution a un statut interdit selon une règle prescrite. Cette solution choisie devient la nouvelle solution. Pour éviter que la recherche intègre un espace de solution non désiré, une liste tabou est mise en place. Ceci peut éviter le cycle.

Pour éviter de donner un statut tabou à la solution non explorée, des conditions d'aspiration sont mises en application. On dit qu'un échange satisfait une condition d'aspiration, s'il aide l'algorithme à trouver un optimum local et à rechercher d'autres meilleures solutions. Ce procédé est répété jusqu'à ce que le nombre exigé d'itérations aient été exécutés ou lorsque un temps CPU donné a été atteint.

Nous pouvons intensifier la recherche dans la région des solutions ou explorer une autre région. Afin d'intensifier la recherche dans les bonnes régions, nous devrions retourner à une des meilleures solutions que nous n'avons jamais trouvées. Nous pouvons également conduire une division de la région de solution, en résolvant des problèmes secondaires de façon optimale. La combinaison de toutes les solutions partielles mènera à une solution optimale. L'intensification peut également être basée sur la mémoire à long terme. Les bons mouvements et les bonnes solutions sont enregistrés et évalués. La solution meilleure-visitée qui partage un espace commun devrait être prise en considération. La diversification dans la recherche taboue est également employée pour éviter un grand secteur voisin inconnu dans le processus de recherche. La diversification est conduite en exécutant plusieurs solutions initiales différentes aléatoires.

L'idée de la recherche Tabou (RT) a été explorée dans les années 70, et a été présentée sous sa forme courante par *Glover* (1986) qui s'est basé sur les idées fondamentales proposées par *Hansen* (1986). La formalisation a été conduite par *Glover* (1969), et *Werra et Hertz* (1989). L'aspect théorique de *SA* a été étudié par *Faigle et Kem* (1992), *Glover* (1992), et *Fox* (1993). *Faigle et Kem* (1992) ont conçu une version

probabiliste de *RT*. Dans cette technique, la probabilité de se déplacer peut être choisie d'une solution courante à un voisin dans un éventail très grand, et ce, sans perdre les propriétés probabilistes de convergence qui peuvent être faites et qui sont semblables au procédé de recuit simulé affiné.

Algorithme :

Initialisation

Choisir une solution admissible $s \in X$;
$s^* := s$;
$nbiter := 0$; (compteurs d'itérations)
$T :=$; (la liste taboue est vide initialement)
Initialiser la fonction d'aspiration A ;
meil_iter : = 0 ;(itération ayant conduit à la meilleure solution s^* trouvée jusque là)

Processus itératif

tant que ($f(s) \rangle f\_$) et ($nbiter - meil\_iter \langle nb\max$))) **faire**
$nbiter := nbiter + 1$;
Générer un ensemble $N' \subseteq N(s)$ de solution voisines de s ;
Choisir la meilleure solution $s' \in N'$ telle que $f(s') \leq A(f(s))$ ou $s' \notin T$;
Mettre a jour la fonction d'aspiration A ;
Mettre a, jour la liste des solutions taboues ;
$s := s'$;
Si $f(s) \langle f(s^*)$ alors $s^* := s$; meil_iter : nbiter

Fig.2.4 - Algorithme de la recherche de Tabou

La méta-heuristique de recherche avec tabous a été appliquée pour résoudre le problème d'affectation quadratique par *Skorin-Kapov* (1990). Cette technique désignée

sous le nom de l'algorithme de tabou-navigation (T-N) est une recherche de tabou simple qui utilise une liste tabou fixe et un arrangement simple de voisinage (des arrangements voisins sont produits en échangeant l'endroit d'une paire d'équipements). Dans cette méthode ni la stratégie d'intensification, ni la stratégie de diversification n'est appliquée. Des critères d'aspiration pour dépasser le statut tabou de matrice et la stratégie à long terme de mémoire pour choisir de nouveaux arrangements sont mis en application pour remettre en marche la recherche. Ils ont constaté que les résultats informatiques indiquent que l'algorithme du tabou-navigation est meilleur que le recuit simulé en qualité de solutions et le temps d'exécution.

Skorin-Kapov (1991) modifie le programme de tabou-navigation en utilisant une analyse simple pour guider les mouvements non améliorés de l'algorithme de recherche. Dans ce programme l'intensification et la diversification sont également appliquées pour économiser le temps d'exécution.

2.3.1.3 Plafond dégradé

L'algorithme de plafond dégradé est une nouvelle métaheuristique de recherche locale présentée par certains auteurs et il fonctionne de la façon suivante :

- À partir d'une certaine solution initiale, il remplace graduellement une solution courante par un candidat choisi parmi son voisinage.

L'algorithme évalue la fonction de coût des candidats et accepte des solutions avec des coûts inférieurs ou égaux à :

- Le coût de la solution courante ;
- La limite supérieure courante ***L.***

Pendant la recherche, la valeur du plafond est réduite et sa valeur initiale est égale à la fonction de coût de la solution initiale. Ainsi, cela exige la définition d'un seul paramètre - le taux d'affaiblissement ΔL

Algorithme :

1. Générer une solution initiale s

2. Calculer la fonction objective de s $f(s)$

3. Initialiser ' ceiling' $B = f(s)$

4. Initialiser le paramètre ΔB

5. Tant que le critère d'arrêt n'est pas atteint faire :

Définir un voisinage $N(s)$

Sélectionner aléatoirement la solution candidate $s^* \in N(s)$

Si $f(s^*) \leq f(s)$ *ou* $f(s^*) \leq B$

Alors accepter s^*

Diminuer la valeur de $B = B - \Delta B$

Fig.2.5 - Algorithme du Plafond dégradé

Cette nouvelle technique de recherche locale exige seulement un paramètre (qui peut être interprété comme temps de recherche). Généralement, une plus longue recherche fournit un meilleur résultat.

La méthode du *''plafond dégradé''* proposée par *E.K Burke, y . Bykov, J.P Newall and Sanja Petrovic (2002)* a été examinée sur des problèmes d'examen réel d'université *(timetabling)* et les expériences ont confirmé l'efficacité de cette technique proposée. Dans une quantité de temps acceptable, de meilleurs résultats furent produits par cet algorithme que d'autres approches éditées pour résoudre des problèmes de *(timetabling).*

2.3.2 Les approches évolutives

Contrairement aux méthodes constructives et de recherche locale qui font intervenir une solution unique, les méthodes évolutives manipulent un groupe de solutions admissibles à chacune des étapes du processus de recherche. L'idée centrale consiste à utiliser régulièrement les propriétés collectives d'un ensemble de solutions distinguables, appelé population, dans le but de guider efficacement la recherche vers de bonnes solutions dans l'espace de recherche.

2.3.2.1 Les algorithmes génétiques

Pour leurs part, les algorithmes génétiques, ou bien évolutionnaires, utilisent la notion de sélection naturelle développée au 19^{ieme} siècle par le scientifique, Darwin. Selon ce

dernier, les mécanismes à l'origine de l'évolution des êtres vivants reposent sur la compétition qui sélectionne les individus les plus adaptés à leur milieu en leur assurant une descendance, ainsi que sur la transmission aux enfants des caractéristiques utiles qui auront permis la survie des parents. Ce mécanisme d'héritage se fonde notamment sur une forme de coopération mise en œuvre par la reproduction sexuée.

Dans le monde des algorithmes évolutionnaires, les ***individus*** soumis à l'évolution sont des solutions, plus ou moins performantes, à un problème posé. Ces solutions appartiennent à l'espace de recherche du problème d'optimisation. L'ensemble des individus traités simultanément par l'algorithme évolutionnaire constitue une ***population.*** Elle évolue durant une succession d'itérations appelées ***générations*** jusqu'à ce qu'un critère d'arrêt, qui prend en compte à priori la qualité des solutions obtenues, soit vérifié.

Durant chaque génération, une succession d'opérateurs est appliquée aux individus d'une population pour engendrer la nouvelle population à la génération suivante. Lorsqu'un ou plusieurs individus sont utilisés par un opérateur, on convient de les désigner comme des ***parents***. Les individus résultant de l'application de l'opérateur sont des ***enfants***. Ainsi, lorsque deux opérateurs sont appliqués en séquence, les enfants engendrés par l'un peuvent devenir des parents pour l'autre.

Les algorithmes génétiques suivent toujours la même procédure de fonctionnement, afin de faire évaluer la population de solution de manière progressive. Ces algorithmes se

basent sur ces différents principes : sélection, croisement et mutation. Les voici décrits plus en détail :

- ***Sélection :***

  À chaque génération, des individus se reproduisent, survivent ou disparaissent de la population sous l'action de deux opérateurs de sélection :

  - La sélection pour la production ou plus simplement sélection, qui détermine combien de fois un individu sera reproduit en une génération.

  - La sélection pour le remplacement, ou plus simplement remplacement, qui détermine quels individus devront disparaître de la population à chaque génération de sorte que de génération en génération, la taille de la population reste constante, ou plus rarement, soit contrôlée selon une politique définie.

- ***Croisement :***

  Une fois que certains individus seront sélectionnés, on les fait se reproduire entre eux et pour cela, on utilise l'opérateur croisement : c'est l'opérateur essentiel. Il combine les génotypes de deux individus pour en obtenir deux nouveaux. L'opérateur respecte généralement les propriétés suivantes :

  - Le croisement de deux parents identiques donnera des descendants identiques aux parents ;

- Deux parents proches l'un de l'autre dans l'espace de recherche engendreront des descendants qui leur seront proches.

Pour une représentation binaire, il existe trois variantes de croisement classiques :

- ✓ Le croisement « un point » ;
- ✓ Le croisement « deux points » ;
- ✓ Le croisement « uniforme ».

Le croisement un point se déroule en deux étapes :

1. Choix aléatoire d'un point de coupure identique sur les deux chaînes binaires ;
2. Coupure des deux chaînes et échange des deux fragments situés à droite.

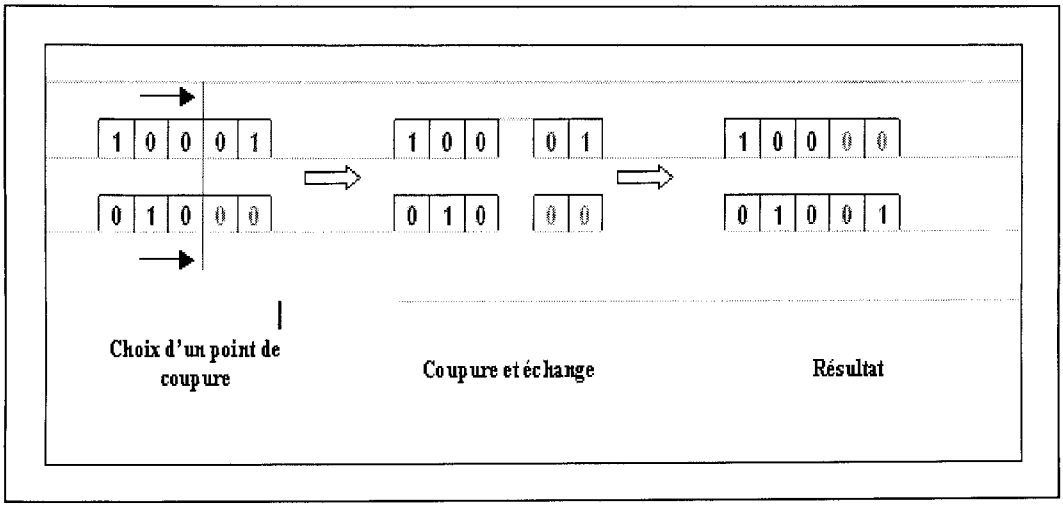


Fig.2.6 - Croisement « un point » de deux génotypes de 5 bits

❖ ***Mutation :***

L'opérateur de mutation modifie aléatoirement la valeur de certains *bits* d'un génotype avec une faible probabilité, typiquement entre 0.01 et 0.001. Si ce dernier est trop élevé, on risque de ne pas voir une bonne convergence, et s'il est trop faible, cela réduit d'autant son effet. L'intérêt de cet opérateur est d'éviter une dérive génétique: certains gènes favorisés par le hasard peuvent se répandre au détriment des autres et sont ainsi présents au même endroit sur tous les génotypes. La mutation permet alors un maintien de la diversité génétique utile pour une bonne exploration de l'espace de recherche.

Un autre intérêt qu'à l'opérateur est de permettre une meilleure recherche locale, notamment quand la plupart des individus ont convergé autour de l'optimum global. À ce moment, le croisement est assez inefficace, car les individus sont souvent identiques. La mutation leur donne alors la chance de s'approcher du maximum globale d'autant que le permet la précision du codage. On a alors principalement le croisement pour explorer globalement et entièrement l'espace de recherche et la mutation pour la recherche locale et l'optimisation de solutions déjà utilisables.

Les différents opérateurs qui interviennent dans un algorithme évolutionnaire sont représentés selon le schéma ci-dessous :

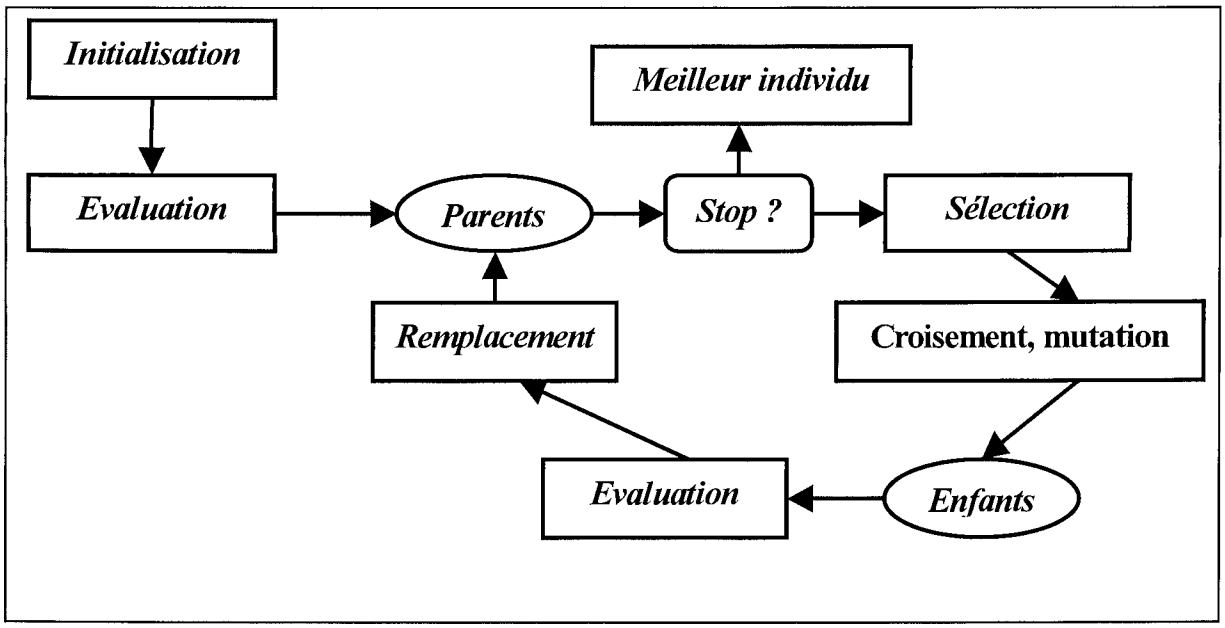


Fig.2.7 -. Algorithmes évolutionnaire

Les auteurs suivants ont employé *GA* pour résoudre le QAP : *Fluerent et Ferland (1994)* ont mis en application *GA*, qui emploie des méthodes de recherche locale pour améliorer la forme physique des individus. *Tate et Smith (1995)* ont mis en application l'algorithme génétique et l'ont examiné sur des exemples de problème donnés par *Nugent et autres (1968). Suresh et autres (1995)* ont examiné trois opérateurs standard de croisement et ont proposé un nouvel opérateur de croisement pour éviter les solutions infaisables dans la population. *Ahuja et autres (2000)* ont utilisé le *GA* avide avec de nouveaux arrangements de croisement et un arrangement d'immigration pour favoriser la diversité.

2.3.2.2 Algorithme à colonies de fourmis

Ce type d'algorithme a été proposé récemment (1992, par *Marc Dorigo*) et il s'inspire des comportements collectifs de dépôt et de suivie de traces. Dans la nature, les fourmis utilisent des phéromones pour marquer des informations dans leur environnement. Elles utilisent ensuite leur capteur de phéromones pour déterminer le chemin le plus marqué en phéromones qu'elles suivront. Les études ont démontrées que les fourmis étaient capables de sélectionner le plus court chemin pour aller du nid à une source de nourriture grâce au dépôt et au suivi de pistes de phéromone. Cette conduite est le résultat d'un mode de communication indirecte, via l'environnement : la « stigmergie ». Chaque fourmi dépose, le long de son chemin, une substance chimique, nommée phéromone. Tous les membres de la colonie perçoivent cette substance et orientent préférentiellement leur marche vers les régions les plus odorantes.

Les algorithmes de colonies de fourmis possèdent plusieurs caractéristiques intéressantes, mentionnons notamment :

- **La flexibilité** : une colonie de fourmis est capable de s'adapter à des modifications de l'environnement.
- **La robustesse** : une colonie est apte à maintenir son activité si quelques individus sont défaillants.
- **La décentralisation** : une colonie n'obéit pas à une autorité centralisée.

- **L'auto-organisation** : une colonie trouve elle-même une solution, qui n'est pas connue à l'avance.

Dorigo et al. ont proposé un nouvel algorithme pour la résolution du problème du voyageur de commerce. Depuis ces travaux, la démarche a été étendue à beaucoup d'autres problèmes d'optimisation combinatoires ou mêmes continus. Considérons l'exemple de la *figure2.9 (Dorigo et al.*, 1996) où une colonie de fourmis se déplace de la source de nourritures vers le nid E en sélectionnant l'un des deux chemins de longueurs différentes (créée par une barrière).

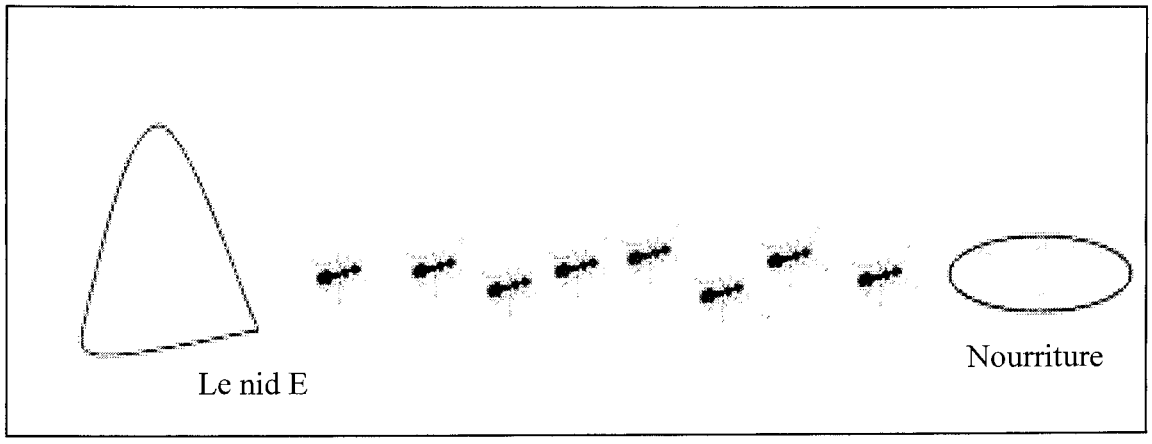


Fig.2.8 - Exemple de fourmis réelles

Des fourmis réelles suivent un chemin entre le nid et une source de nourriture :

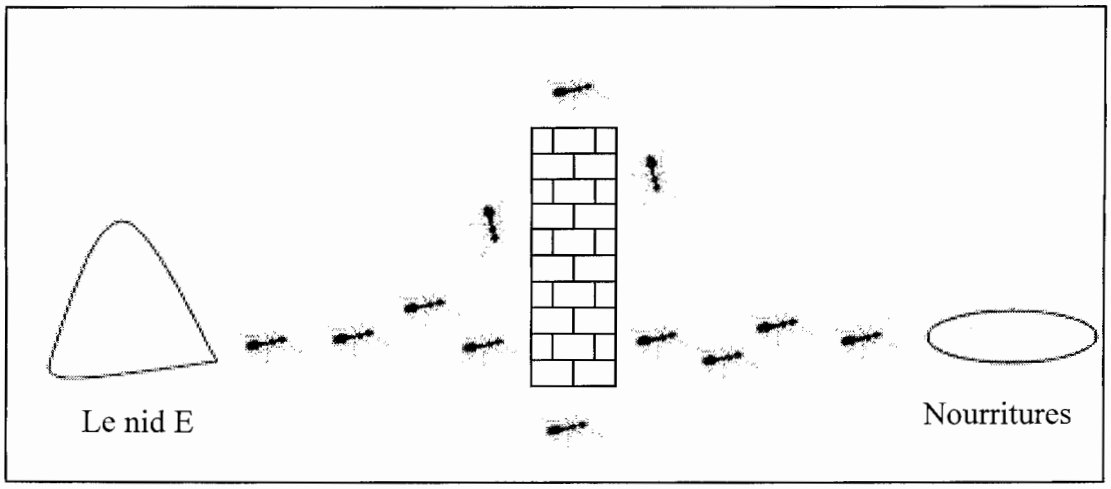


Fig.2.9 - . Cas d'un obstacle dans le chemin

Un obstacle survient sur le chemin ; les fourmis choisissent de tourner vers la gauche ou vers la droite avec des probabilités égales et la phéromone est déposée plus rapidement sur le chemin le plus court.

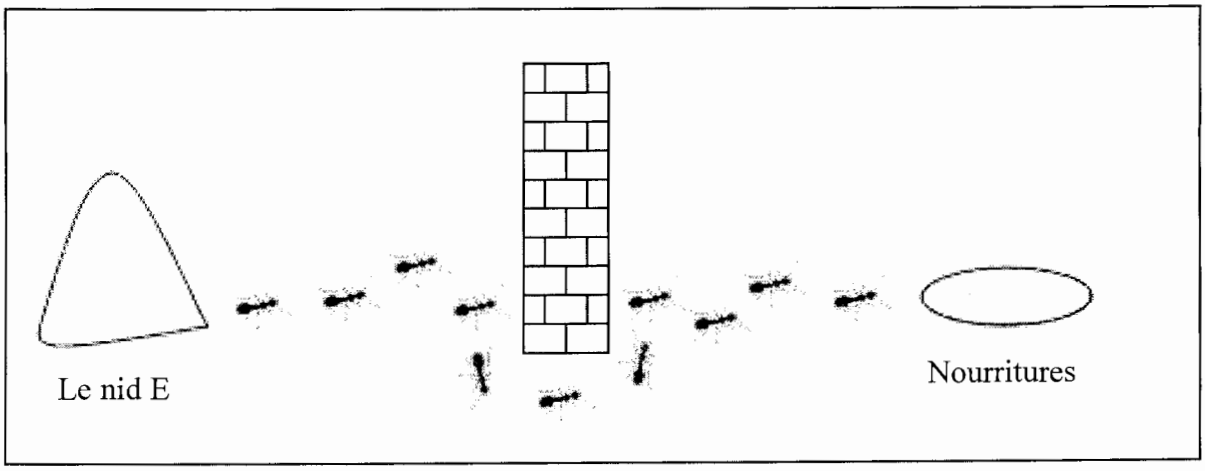


Fig.2.10 - Les fourmis ont choisi le chemin le plus court

Lorsqu'une colonie de fourmis a le choix entre deux chemins de longueurs différentes pour exploiter une source de nourriture, elle sélectionne le chemin le plus court si la différence entre les longueurs des chemins est suffisamment importante. Les fourmis déposent de la phéromone lors de l'aller vers la source de nourriture et au retour vers le nid. Au départ, le choix est aléatoire, mais le chemin le plus court devient vite le plus marqué par la phéromone, car les fourmis qui l'empruntent arrivent plus vite au nid et auront statistiquement plus de chance de l'emprunter lorsqu'elles retourneront vers la source de nourriture.

Les auteurs *M. Dorigo, Luca Maria Gambardella (1996)* ont proposé l'application de *l'Ant System* sur le problème du voyageur de commerce (TSP). Le problème *TSP* se présente comme suit : étant donné un ensemble de *n* villes, l'objectif est de trouver le tour ayant la plus courte distance. La distance séparant une ville *i* à une ville *j* est notée d_{ij} (distance euclidienne). Ce problème peut être représenté par un graphe (N,E), où N est l'ensemble des villes (nœuds) et E est l'ensemble des arêtes entre les villes.

Soit $b_i(t)$ (i=1,.., n) l'ensemble des fourmis dans la ville *i* au temps *t* et soit $m = \sum_{i=1}^{n} b_i(t)$, le nombre total des fourmis. Chaque fourmi est un agent avec les caractéristiques suivantes :

- La fourmi choisit la prochaine ville de destination avec une probabilité qui est en fonction de la distance de la ville et de la quantité de phéromones présente sur l'arête de connexion.
- Les villes déjà visitées par la fourmi sont retirées de la liste des villes qui restent à visiter. On utilisera pour cela une liste taboue.
- Lorsqu'une fourmi a terminé un tour, une quantité de phéromones est déposée sur chaque arête (i, j) visitée.

Soit $\tau_{ij}(t)$ la quantité de phéromones sur l'arête (i, j) au temps t. chaque fourmi au temps t choisit la prochaine ville où elle s'y positionnera au temps t+1. Une itération comprendra donc l'ensemble des m mouvements. Les fourmis compléteront leurs tours après n itérations (un cycle) et la mise à jour des phéromones se fera à cet instant comme suit :

$$\tau_{ij}(t+n) = \rho.\tau_{ij}(t) + \Delta\tau_{ij} \qquad \textbf{Equ.2.2}$$

où ρ est un coefficient tel que (1-ρ) représente l'évaporation de la phéromone entre le temps t et t+n.

$$\Delta\tau_{ij} = \sum_{k=1}^{m} \Delta\tau_{ij}^{k} \qquad \textbf{Equ.2.3}$$

Où $\Delta\tau_{ij}^{k}$ est la quantité de phéromone par unité de longueur déposée sur l'arête (i, j) par la k ème fourmis entre le temps t et $t+n$. Elle est donnée par :

$$\Delta\tau_{ij}^{k} = \begin{cases} \dfrac{Q}{L_k} & \text{si la kème fourmis utilise l'arete (i, j) dans son tour} \\ 0 & \text{sinon} \end{cases} \qquad \textit{Equ.2.4}$$

Q est un nombre positif constant et L_k est la longueur du tour de la k ème fourmis. Une liste tabou (notée $tabu_k$) contenant les villes visitées est associée à chaque fourmis.

La quantité $1/d_{ij}$ est appelée visibilité et noté η_{ij} . Cette quantité n'est pas modifiée durant l'exécution de l'*Ant system* (constante). La probabilité de transition est définie comme suit :

$$p_{ij}^{k}(t) = \begin{cases} \dfrac{[\tau_{ij}(t)]^{\alpha}.[\eta_{ij}(t)]^{\beta}}{\sum_{k \in villes\_restantes_k} [\tau_{ik}(t)]^{\alpha}.[\eta_{ik}(t)]^{\beta}} & \text{si j} \in \text{villes\_Restantes}_{k} \\ 0 & \text{sinon} \end{cases} \qquad \textit{Equ.2.5}$$

Où villes_restantes$_k$ = $\{N - tabu_k\}$. α et β sont des paramètres qui contrôlent l'importance relative de la phéromone versus la visibilité.

Le *QAP* est adressé en tant qu'un des problèmes les plus complexes et qui est très difficile à être résolu de façon optimale dans un temps acceptable. Par conséquent,

plusieurs heuristiques, tels que le recuit simulé (*Connolly*, 1990), la recherche tabou (*Taillard,* 1991), l'algorithme génétique (*Tate et Smith*, 1995), les algorithmes de l'optimisation de colonie de fourmis (ACO) (*Maniezzo et Colorni*, 1999), etc., ont été développés par les chercheurs pour fournir les solutions proche de l'optimal pour le *QAP*. On a montré que les algorithmes d'*ACO* sont parmi les meilleures méthodes pour résoudre le *QAP* (*Stutzle et Dorigo*, 1999). Il y a beaucoup de réalisations avec des algorithmes de fourmis. *Dorigo et autres* (1996) ont appliqué l'algorithme d'*ACO* pour résoudre le problème de voyageur de commerce (TSP) et ont pu prolonger leur approche pour résoudre *TSP* asymétrique. *Taillard et Gambardella* (1997) ont proposé un algorithme rapide de fourmis notamment *FANT* pour le *QAP*. Dans l'algorithme de *FANT,* chaque itération emploie seulement une fourmi et des tâches sont faites sans information heuristique. *Gambardella et autres* (1999) ont proposé un algorithme de fourmis appelé *HAS-QAP* pour résoudre le *QAP*. La différence principale de cette méthode par rapport à d'autres algorithmes de fourmis est que des traînées de phéromones ne sont pas employées pour construire, mais pour modifier les solutions. Ils ont signalé que les *HAS-QAP* et les algorithmes (GH) hybrides génétiques sont parmis les meilleures méthodes pour résoudre le *QAP*. *Maniezzo et Colorni* (1999) ont amélioré l'exécution de l'algorithme proposé par *Dorigo et autres* (1996) et ont employé le concept de limite inférieure pour mesurer l'attraction de chaque affectation. Dans cette approche, la limite inférieure sur la valeur de fonction objective de la solution partielle est calculée pour chaque affectation.

2.4 CONCLUSION

Tel que nous l'avons vu au cours de ce chapitre, il existe deux types d'approches : les méthodes exactes et les approches méta-heuristiques. Dans la section 2.2 on a présenté quelques méthodes exactes trouvées dans la littérature, tels que *branch-and-bound*, les plants sécants. Ces méthodes de résolution sont efficaces que pour les problèmes de petite taille. Dans un second temps, différents types méta-heuristiques pour le PAQ, section (2.3), ont été présentées dont le recuit simulé, la recherche avec tabous, le plafond dégradé, les algorithmes génétiques et les colonies de fourmis. Cette dernière est hybridée avec l'algorithme de plafond dégradé dans ce travail pour résoudre le problème d'aménagement d'usines formulé comme un problème d'affectation quadratique.

CHAPITRE 3

MÉTHODOLOGIE

3.1 INTRODUCTION

Les problèmes d'aménagement concernent l'arrangement des équipements physiques. Historiquement, une grande partie de la recherche fut consacrée au problème statique d'aménagement, qui, dans sa version simpliste, peut être modélisé comme un problème d'affectation quadratique (PAQ). Beaucoup de chercheurs ont essayé de résoudre ce problème avec des techniques combinatoires traditionnelles d'optimisation. Cependant, des solutions optimales furent trouvées seulement pour des problèmes de petites tailles. En raison de la nature combinatoire du PAQ, le nombre de candidats de solution augmente exponentiellement avec le nombre d'équipements à localiser.

En d'autres termes, l'effort informatique de trouver la solution optimale pour ce problème se développe exponentiellement avec la taille du problème. On sait que les

algorithmes exacts nécessitent un effort informatique élevé. Par conséquent, le besoin de l'heuristique pour fournir de bonnes solutions dans un temps raisonnable a augmenté.

Dans cette recherche, une hybridation de deux méta-heuristiques *(l'algorithme à colonies de fourmis et l'algorithme de plafond dégradé)* est employée pour résoudre le problème d'aménagement. La méthodologie sera expliquée dans la section suivante.

3.2 IDÉES DE BASE

- **Développer une méthode de résolution :**

Une méta-heuristique dans laquelle on fait une implémentation de la méthode de recherche locale. On a pensé à utiliser la recherche locale pour améliorer les solutions.

- **Méta-heuristique de base**
  - ✓ ***Colonies de fourmis***

Les **algorithmes à colonies de fourmis** forment une famille de méta-heuristiques d'optimisation qui s'inspirent du comportement collectif des fourmis. Proposé à la fin des années 1980 par *Moyson et Manderick* et largement exploité et développé par *Dorigo* dans les années 90 [Dor92], pour la recherche de chemins optimaux dans un graphe.

L'algorithme original s'inspire du comportement des fourmis recherchant un chemin allant de leur colonie vers une source de nourriture. L'idée originale est depuis déclinée pour résoudre une classe plus large de problèmes.

L'idée originale est issue de l'observation du comportement collectif d'exploitation de la nourriture chez les fourmis. En effet, celles-ci, bien qu'ayant individuellement des capacités cognitives très limitées, sont capables collectivement à résoudre le problème de la découverte du chemin le plus court entre une source de nourriture et leur nid.

- **La recherche locale**
  - ✓ ***Plafond dégradé***

C'est une technique de recherche locale parmi les méthodes les plus efficaces pour l'optimisation. On commence par une solution initiale, on se déplace vers une solution voisine et ainsi de suite jusqu'à l'optimum.

L'algorithme accepte chaque solution dont la fonction objective est inférieure ou égale à la limite supérieure '' B''.

Les méta-heuristiques intègrent généralement des méthodes de recherche locale. En répétant des recherches locales sur des solutions générées aléatoirement, on obtient de bons résultats. Si l'on part de solutions qui sont déjà proches de l'optimum, on devrait trouver la solution plus rapidement, et c'est là que se situe le rôle de la méta-heuristique qui doit générer de bonnes solutions pour la recherche locale.

Pour notre recherche, nous avons choisi l'algorithme à plafond dégradé. Ce choix s'explique par le fait que cette méthode est très performante pour le PAQ.

✓ ***Schéma explicatif :***

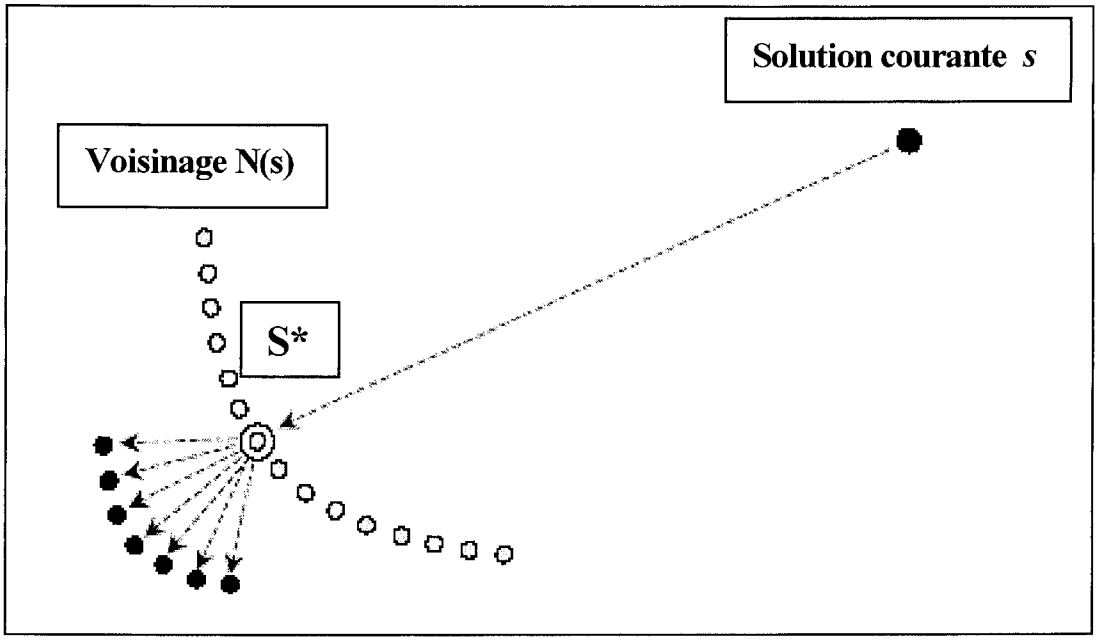


Figure 3.1 - Recherche locale

Successivement, on répète le remplacement de la solution courante s par la nouvelle s^*. Jusqu'à ce qu'une certaine condition d'arrêt ait été satisfaite, la nouvelle solution est choisie parmi un voisinage $N(s)$ et la qualité de la solution est caractérisée par la fonction coût $f(s)$. Donc le but de cette recherche locale est de ***minimiser la fonction coût.***

Le but est d'améliorer la solution obtenue par l'algorithme à colonies de fourmis pour avoir une solution globale

3.3 MÉTHODE DE RÉSOLUTION

En appliquant l'algorithme à colonies de fourmis, on pourra présenter notre problème d'affectation quadratique (QAP) comme suit :

Considérons un problème de ''n'' objets et '' n'' locations tel que :

$D(n \times n)$ Une matrice telle que

$d(i, j)$ Représente la distance entre location '' i '' et location '' j ''

$F(n \times n)$ Une matrice de dimension $(n \times n)$

$F(k, l)$ Représente le flux entre les objets ''k'' et '' l.''

✓ ***Notation :***

Soit $p(i)$ est un nombre compris entre '' 0 '' et '' n '', indiquant la location affectée à l'objet '' i ''

Exemple :

P(x) = y ----» la location (zone) y est affectée à l'objet x.

✓ ***Fonction objective :***

Représente la fonction coût du problème à minimiser

Min: $$L = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} f(i, j) d(p(i), p(j)) \qquad \textit{Equ. 3.1}$$

✓ ***Application de l'algorithme colonies de fourmis_plafond dégradé - au problème***

L'algorithme ne serait pas complet sans le processus d'évaporation des pistes de phéromones. En effet, pour éviter d'être piégé dans des solutions sous optimales, il est nécessaire de permettre au système d'oublier les mauvaises solutions. On contrebalance donc l'additivité des pistes par une décroissance constante des valeurs des arrêtes à chaque itération.

L`algorithme générale se présente de la façon suivante :

1- Initialisations

2- Pour chaque fourmi *k*

- **Construire la solution $s(k)$**
- **Mise à jour locale**

3- **Appliquer la recherche locale sur toutes les solutions S (k=1 : nb fourmis)** ***l'algorithme à plafond dégradé***

4- Mettre à jour la meilleure solution

5- Mise à jour globale de phéromones

-----» cette mise à jour est appliquée sur la meilleure solution

6- Retourner au point deux (2) pour le cycle suivant

Figure 3.2 - Algorithme (colonies de fourmis_plafond dégradé)

✓ ***Structure du «algorithme à colonies de fourmis »***

❖ **Probabilité de transition :**

Chaque fourmi choisit un objet à affecter à une zone (ou location) suivant la formule suivante :

Pour un q_0 donné $(0 \leq qo \leq 1)$

J= $\{\arg\max(\tau_{im})^{\alpha}(\eta_{im})^{\beta}\}$ si q<q0

J : Variable aléatoire sélectionnée suivant la distribution de probabilité :

$$p_{ij} = \begin{cases} \dfrac{(\tau_{ij})^{\alpha}(\eta_{ij})^{\beta}}{\sum_{m \in Z_{disp}} (\tau_{im})^{\alpha}(\eta_{im})^{\beta}} & \text{Si } j \in Z_{disp} \\ p_{ij} = 0 & \text{Sinon} \end{cases} \qquad \textit{Eq 3.2}$$

Z_{disp} = Location encore disponible

❖ Paramètres de problèmes

Dans l'algorithme à colonies de fourmis les paramètres α et β jouent un rôle très important pour avoir une solution optimale. α et β sont donc des paramètres qui contrôlent l'importance relative de la phéromone versus, et de la visibilité.

Avec α=0, seule la visibilité est prise en compte .au contraire, avec β=0, seules les pistes de phéromone jouent. Alors pour éviter une sélection trop rapide (affectation), un compromis entre ces deux paramètres est nécessaire (réglages de paramètres CH 4).

ρ est un coefficient qui représente la quantité de phéromone déposé, appelée l'intensité de la piste. Ce paramètre définit l'attractivité d'une partie du trajet global et change à chaque passage d'une fourmi, c'est en quelque sorte une mémoire globale du système, qui évolue par apprentissage.

❖ Information heuristique (η_{ij})

Information heuristique aussi appelée la visibilité se calcule comme suit :

$$\eta_{ij} = \frac{1}{d_i \, f_j} \quad avec \quad \begin{cases} i = 1,2,...,n \\ \\ j = 1,2,...,n \end{cases} \qquad \textbf{Equ 3.3}$$

avec

$$d_i = \sum_{j=1}^{n} d_{ij} \qquad i = 1, 2, ..., n$$

$$f_i = \sum_{l=1}^{n} f_{il} \qquad i = 1, 2, ..., n$$

Exemple

Pour n = 3

Distance entre les zones : $D = \begin{bmatrix} 0 & 5 & 3 \\ 5 & 0 & 1 \\ 3 & 1 & 0 \end{bmatrix}$

Flux entre objets : $F = \begin{bmatrix} 0 & 1 & 2 \\ 1 & 0 & 6 \\ 2 & 6 & 0 \end{bmatrix}$

L'information heuristique entre objets (j=3) et zones (i=2) est :

$$\eta_{23} = \frac{1}{d_2 \; f_3}$$

avec :

$d_2 = 5+0+1 = 6$ $2^{ème}$ ligne de la matrice D

$f_3 = 2+6+0 = 8$ $3^{ème}$ ligne de la matrice F

- ✓ ***Mise à jour locale :***

À chaque affectation réalisée par une fourmi, on applique la mise à jour locale suivante :

$$\tau_{ij} \leftarrow (1 - \rho)\tau_{ij} + \rho\ \tau_0 \qquad \textit{Equ 3.4}$$

✓ ***Mise à jour globale :***

La mise à jour globale se fait sur la meilleure solution :

Soit S_{best} la meilleure solution, la mise à jour globale se fait sur les arcs (i, j) de la meilleure solution :

$$\tau_{ij} \leftarrow (1 - \rho)\tau_{ij} + \rho\ \Delta\tau_{ij} \qquad \textit{Equ 3.5}$$

avec $\Delta\tau_{ij} = \dfrac{1}{S_{best}}$

Exemple :

Si la solution optimale est :

Zone 1 ← Objet 3

Zone 2 ←Objets 1

Zone 3 ←Objets 2

3	**1**	**2**

Alors la mise à jour se fait sur les arcs : (1,3) ; (2,1) ; (3,2)

✓ ***Algorithme du plafond dégradé***

Après avoir trouvé une solution par l'algorithme à colonies de fourmis, on cherche à améliorer la solution courante $f(s)$ trouvée en appliquant la méthode de recherche locale. Pour réaliser ceci on introduit l'algorithme du Plafond dégradé.

L'algorithme se présente de la façon suivante :

1- Générer une solution initiale S
2- Calculer la fonction objective notée par f(s)
3- Initialiser le paramètre ΔB
Tant que le critère d'arrêt n'est pas atteint, faire :
- **Définir un voisinage N(s)**
- **Sélectionner aléatoirement la solution candidate $S^* \in N(s)$**
- **Si $f(s^*) \leq f(s)$ ou $f(s^*) \leq B$**
- **Alors accepter S^***
- **Diminuer la valeur de B : $B = B - \Delta B$**

Figure 3.3 - Algorithme du plafond dégradé

Avec cette procédure, les bonnes solutions (meilleure que la solution optimale) et les mauvaises solutions (inférieure à B) seront acceptées.

3.4 FONCTIONNEMENT DE L'ALGORITHME PROPOSÉ

(Colonies de fourmis_plafond dégradé)

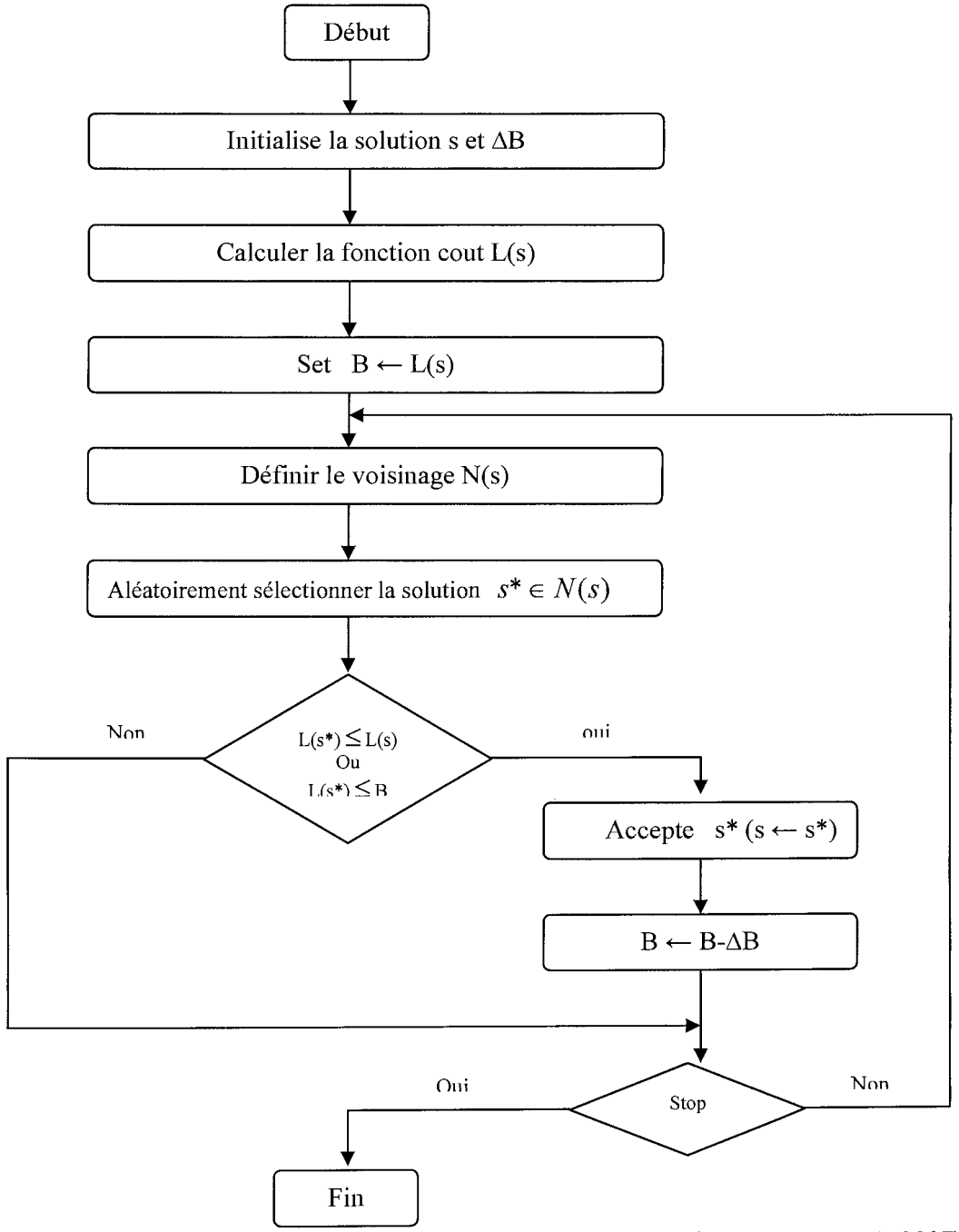


Figure 3.4 - Organigramme de l'algorithme du plafond dégradé **(Nourelfath *et al.*, 2007)**

Voici l'organigramme de l'algorithme de *colonies de fourmis* qu'on va développer pour résoudre le problème d'affectation quadratique.

La recherche locale est la métaheuristique de *plafond dégradé* a été présentée dans la figure3.3.

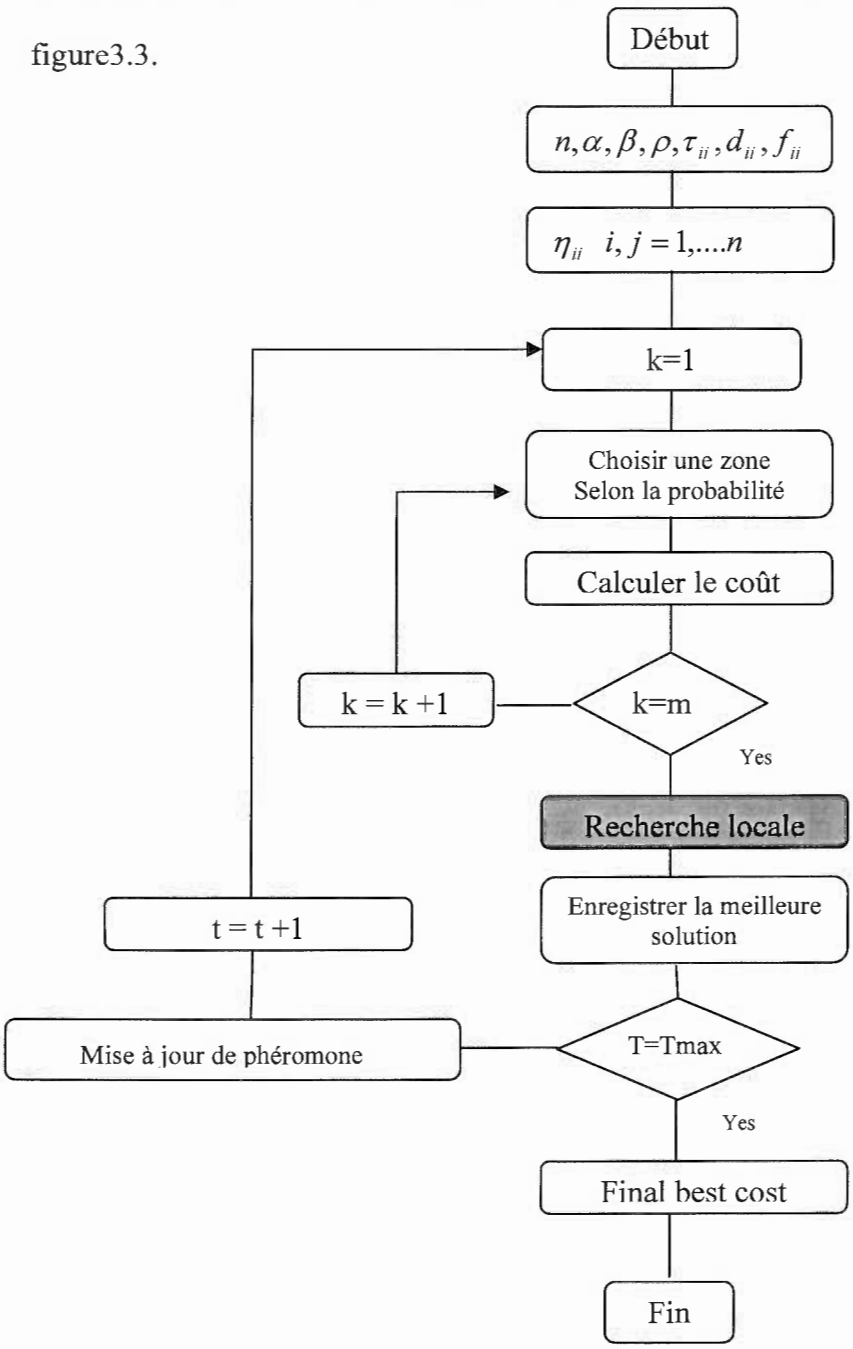


Figure 3.5 - Organigramme du l'algorithme à colonies de fourmis

3.5 Exemple de déroulement de l'algorithme

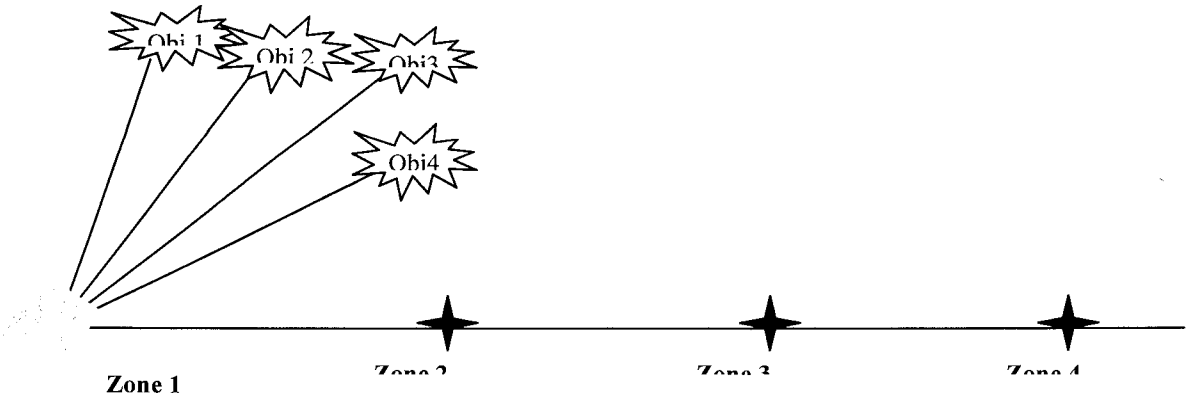


Figure 3.6.a - Exemple simple de 4 zones, 4 locations

Une fourmi choisi un trajet selon l'équation de probabilité, et trace une piste de phéromone.

L'ensemble des fourmis parcourt un certain nombre de trajets, chaque fourmi déposant une quantité de phéromone proportionnelle à la qualité du parcours.

Chaque arête du meilleur chemin est plus renforcée que les autres.

L'évaporation fait disparaître les mauvaises solutions.

Ce qui va expliquer le choix d'affecter l'objet 4 à la zone 1 dans cet exemple

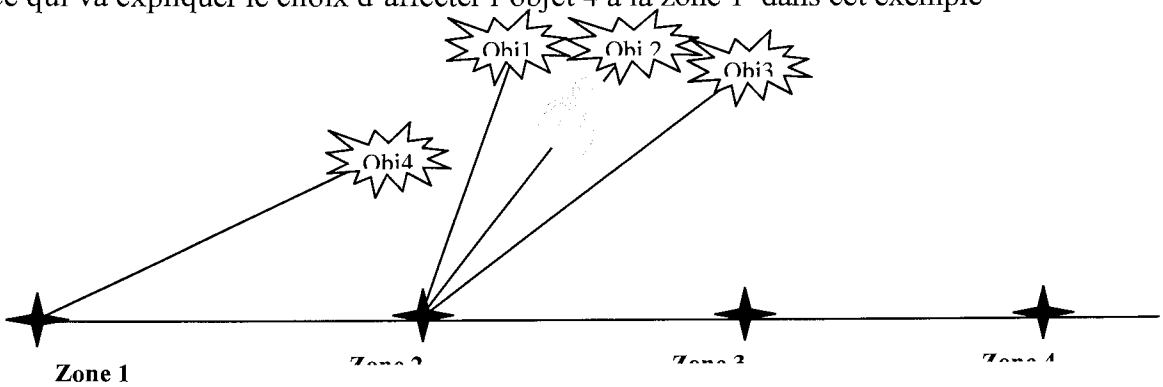


Figure 3.6.a - Exemple simple de 4 zones, 4 locations

De la même manière les fourmis vont finir par affecter les objets 2,1 et 3 aux zones 2,3 et4.

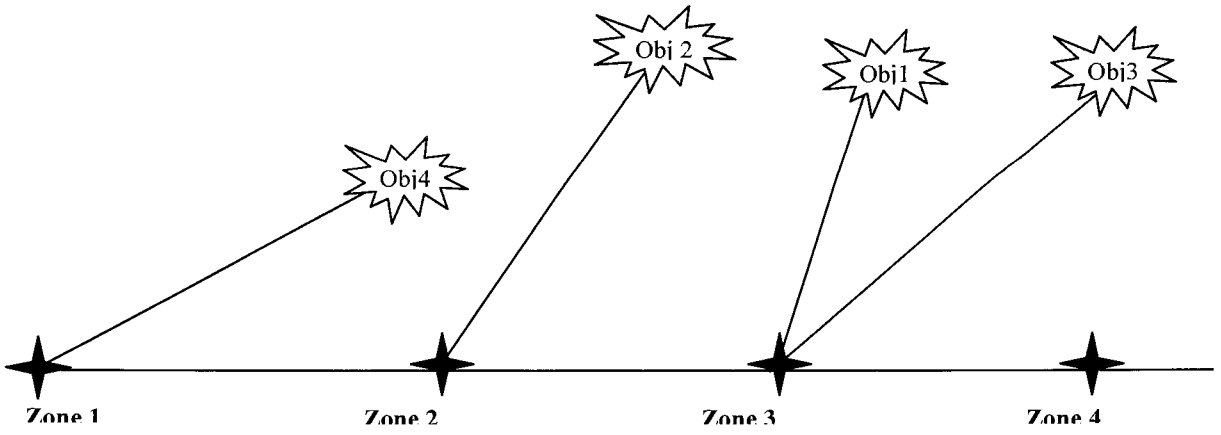


Figure 3.6.c - Exemple simple de 4 zones, 4 locations

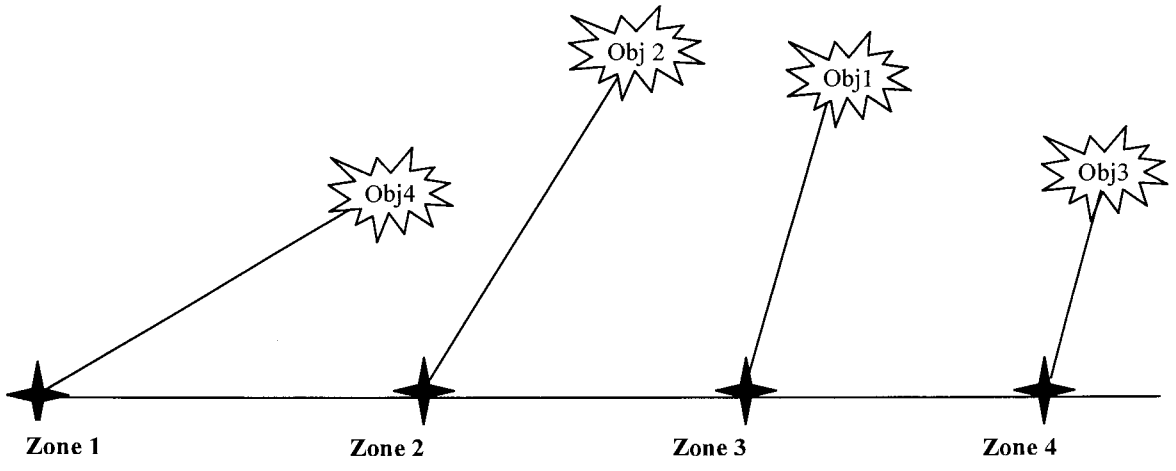


Figure 3.6.d - Exemple simple de 4 zones, 4 locations

Cet optimum est une solution qui ne connaît pas de meilleure solution parmi ses voisines d'où l'utilité de la notion de recherche locale, en faisant une permutation de la solution courante ce qui est l'objectif de l'algorithme de plafond dégradée.

Un exemple numérique est donné dans l'annexe.

3.6 CONCLUSION

Au cours de ce chapitre nous avons présenté l'hybridation employée pour résoudre le problème d'affectation quadratique. L'algorithme hybride que nous avons utilisé, combine l'algorithme à colonies de fourmis et l'algorithme de plafond dégradé. Le principe de cette hybridation est de favoriser une exploration aussi complète que possible de l'espace de recherche et améliorer la solution courante obtenue par l'algorithme à colonies de fourmis. Afin de tester l'efficacité de cette méthode, le prochain chapitre présentera diverses expérimentations sur des instances de problèmes répertoriées dans la littérature.

CHAPITRE 4

EXPÉRIMENTATION ET RÉSULTATS

4.1 INTRODUCTION

Ce chapitre est consacré aux résultats pratique obtenus avec ***l`algorithme de colonies de fourmis_ plafond dégradé.***

Dans un premier temps, ils seront utilisés pour démontrer l'intérêt de cette méthode et des choix effectués pour ses différents constituants. Ensuite, nous comparons ces résultats obtenus par rapport à d'autres méthodes utilisées pour la résolution de *PAQ*.

L'algorithme proposé, est une hybridation entre les deux métaheuristiques « ***Colonies de fourmis*** et ***plafond dégradé*** », a été testé sur un ensemble de problèmes rencontrer dans la littérature. Il a été codé en *MATLAB* et résolu sur un ordinateur (*PC*) avec un processeur de type Pentium III.

4.2 INSATNACES CONSIDÉRÉES

Un benchmark, en anglais est un point de référence servant à effectuer une mesure. En informatique, un benchmark est un banc d'essai permettant de mesurer les performances d'un system pour le comparer à d'autres.

Parmi les problèmes de benchmark que nous avons utilisé on peut citer Sko 42, Sko56 étudiés par *Skorin-Kapov* sont des instances où la taille de problème change n=42, n=56 ainsi que les matrices de flux et de distance

Pour les instances Tai20, Tai25b et Tai30 sont des problèmes de taille n=20 , n=25 et n=30 traités par *Taillard*

4.3 RÉGLAGE DES PARAMÈTRES

Dans ce problème, l'algorithme présente cinq (5) paramètres nécessitant un réglage manuel : '' ρ '', '' φ '', '' β '', '' α '', '' η ''.

On prend '' α '' =1, car les expériences démontrent que c'est la meilleure valeur.

Après avoir fixé '' α '', on définit par la suite la valeur de '' β ''. Après cinq simulations, la valeur optimale définit la valeur de '' β '' et on suit la même procédure pour définir les autres paramètres.

On prend par exemple le problème ***Tai30.***

❖ ***Le problème : Tai30***

On fixe ''α'' =1 et on fait varier ''ρ"

Exécution	ρ							
	0,09	0,1	0,15	0,2	0,25	0,3	0,35	0,4
1	1828392	1827866	1825384	1830988	2E+06	1828392	1830136	1836194
2	1836468	1836590	1840560	**1818146**	2E+06	1818442	1826230	1818442
3	1834538	1844528	1828392	1830468	2E+06	**1818146**	1833216	1834594
4	1825384	1827970	1826230	1825262	2E+06	1825262	1842832	1825262
5	1827970	1828340	1833754	1825384	2E+06	1826436	1839862	1831754
Moyen	**1830550**	**1833059**	**1830864**	**1826050**	**2E+06**	1823336	**1834455**	**1829249**
Min (My)	**1823336**							
Optmum	**1818146**							

Tableau 4.1 - Réglage du paramètre ''ρ''

On définit le coût minimum et aussi le minimum des moyens après cinq simulations.

On prend la valeur du paramètre qui correspond au coût minimum et aussi celle qui correspond au coût minimum des moyens.

Après cinq simulations, la valeur moyenne minimale et la valeur minimale justifient le réglage de paramètre ''ρ'' à **0.2** pour une et **0.3** pour l'autre.

✓ **Moyenne**

On fixe les paramètres suivants :

$$\begin{cases} \alpha = 1 \\ \beta = 0.2 \\ \rho = 0.3 \\ \eta = 7 \end{cases}$$

On définit par la suite la valeur de '' φ '',

Exécution	φ							
	0,3	0,4	0,5	0,6	0,7	0,8	0,9	0,99
1	1827126	1836942	1833052	1837434	1825262	1825384	1832338	1839766
2	1838722	1846940	1837716	1844140	1831528	1839166	1845256	1825384
3	1843228	1844182	1839274	1838356	1831516	1831806	1830136	1838524
4	1837076	1836984	1827126	1839352	1826230	1832168	1830982	1841558
5	1837856	1828992	1840400	1837030	1831578	1826398	1833798	1849590
Moyen	1836802	1838808	1835514	1839262	1829223	1830984	1834502	1838964
Min (My)	1829223							
Optmum	1829223							

Tableau.4.2 - Réglage du paramètre '' φ'' (valeur moyenne)

Après cinq simulations, la valeur moyenne minimale justifie le réglage de paramètre '' φ '' à ***0.7***.

✓ Minimum

On fixe les paramètres suivants :

$$\begin{cases} \alpha = 1 \\ \beta = 0.2 \\ \rho = 0.2 \\ \eta = 7 \end{cases}$$

On définit par suit la valeur de '' φ ''

Exécution	φ							
	0,3	0,4	0,5	0,6	0,7	0,8	0,9	0,99
1	1833152	1840202	1834738	1826230	1834626	1833112	**1825384**	1841460
2	1832168	1841934	1838824	1839386	1831522	1836814	1835232	1833456
3	1846600	1833422	1840492	1839550	1838994	1844778	1832636	1838978
4	1835678	1833684	1838388	1831232	1833594	1834636	1831164	1843280
5	1847704	1837706	1835854	1830702	1839464	1829534	1826398	1845134
Moyen	1839060	1837390	1837659	1833420	1835640	1835775	1830163	1840462
Min (My)								
Optmum	**1825384**							

Tableau.4.3 - Réglage du paramètre '' φ'' (valeur minimale)

Après cinq (5) simulations, la valeur minimale justifie le réglage de paramètre '' φ '' à

0.9

✓ ***Moyenne***

On fixe les paramètres suivants :

$$\begin{cases} \alpha = 1 \\ \beta = 0.2 \\ \rho = 0.3 \\ \varphi = 0.7 \end{cases}$$

On définit par suit la valeur de « η »

Exécution	η							
	1	2	3	5	7	13	17	21
1	1838454	1833896	1836590	1841812	1828392	1834112	1832138	1827934
2	1834264	1839858	1828576	1829864	1841042	1830592	1839180	1843300
3	1835956	1827934	**1818146**	1827234	1839126	1834008	1840976	1840562
4	1844744	1827866	1837602	1837662	1841430	1837722	1838668	1825262
5	1840100	1828576	1827810	1837754	1835726	1829864	1831232	1838622
Moyen	1838704	1831626	1829745	1834865	1837143	**1833260**	1836439	1835136
Min (My	**1833260**							
Optmum	**1833260**							

Tableau.4.4-. Réglage du paramètre '' η''. (valeur moyenne)

✓ *Minimum*

On fixe les paramètres suivants :

$$\begin{cases} \alpha = 1 \\ \beta = 0.2 \\ \rho = 0.2 \\ \varphi = 0.9 \end{cases}$$

Et on définit par suit la valeur de $'' \eta ''$

Exécution	η								
	1	2	3	5	7	13	17	21	25
1	1838386	1844186	1825384	1818442	1828392	1837238	1825384	1846630	1834506
2	1835720	1846626	1826230	1838906	1847494	1835202	1840198	1840198	1833290
3	1831578	1838378	1838156	1825384	1829136	1828392	1840660	1825262	1828576
4	1837060	1825262	1822402	1825262	1841106	1824150	1824150	1824150	1846564
5	1839972	1838422	1826398	1818146	1827970	1818146	1839440	1826230	1841694
Moyen	1836543	1838575	1827714	1825228	1834820	1828626	1833966	1832494	1836926
Min (My									
Optmum	1818146								

Tableau.4.5 -. Réglage du paramètre '' η''. (valeur minimale)

Finalement, on trouve que les bons paramètres qui seront utilisés après le réglage sont :

✓ ***Min***

Paramètre	Valeur
α	*1*
β	*0.2*
ρ	*0.2*
φ	**0.9**
η	*13*

Tableau.4.6 - . Paramètres choisis (min)

✓ ***Moyenne***

Paramètre	Valeur
α	*1*
β	*0.2*
ρ	*0.3*
φ	*0.7*
η	*5*

Tableau.4.7 - . Paramètres choisis (moyen)

Alors, on se base sur ces paramètres pour trouver l'optimum du problème ***Tai 30*** ; la même procédure est appliquée pour tous les problèmes qui seront traités.

4.4 RÉSULTATS NUMÉRIQUES

On a appliqué l'algorithme à colonies de fourmis sur les différents problèmes de la littérature et on a obtenu de bons résultats.Les tableaux suivants montrent les différents paramètres ainsi que les résultats obtenus pour chaque problème.

❖ *Problème : Sko42*

✓ *Les paramètres*

$$\begin{cases} \alpha = 1 \\ \beta = 0.29 \\ \rho = 0.9 \\ \eta = 17 \\ \varphi = 0.93 \end{cases}$$

✓ *Les résultats obtenus*

	Coût	Cycles/3.27
1 er	15812	85
2 ème	15812	48
3 ème	15812	110
4 ème	15812	262
5 ème	15812	60
6 ème	15812	72
7 ème	15812	233
8 ème	15812	45
9 ème	15812	24
10 ème	15812	98

Tableau.4.8 - Résultats obtenus pour le problème Sko42

❑ *Qualité de la solution*

$$\text{Erreur} = \frac{(coûtmoyen - coûtopt)}{coûtopt} * 100 = 0\ \% \qquad \text{Eq 4.1}$$

❖ *Problème : Tai 20*

✓ *Les paramètres*

$$\begin{cases} \alpha = 1 \\ \beta = 0.09 \\ \rho = 0.8 \\ \eta = 13 \\ \varphi = 0.93 \end{cases}$$

✓ *Les résultats obtenus*

	Coût opt	Cycles/2.6
1 er	703 482	4
2 ème	703 482	260
3 ème	703 482	540
4 ème	703 482	14
5 ème	703 482	132
6 ème	703 482	484
7 ème	703 482	112
8 ème	703 482	95
9 ème	703 482	29
10 ème	703 482	153

Tableau.4.9 - Résultats obtenus pour le problème Tai20

❑ *Qualité*

$$\text{Erreur} = \frac{(coûtmoyen - coûtopt)}{coûtopt} * 100 = 0\ \% \qquad \text{Eq 4.2}$$

❖ *Problème : Tai 25b*

✓ *Les paramètres*

$$\begin{cases} \alpha = 1 \\ \beta = 0.29 \\ \rho = 0.1 \\ \eta = 15 \\ \varphi = 0.9 \end{cases}$$

✓ *Les résultats obtenus*

	Coût opta	Cycles/2.6
1 er	**1 167 256**	2
2 ème	**1 167 256**	22
3 ème	**1 167 256**	147
4 ème	1 171 526	100
5 ème	**1 167 256**	15
6 ème	1 169 030	136
7 ème	1 176 312	600
8 ème	1 174 640	145
9 ème	1 171 526	341
10 ème	1 175 564	155

Tableau.4.10 - Résultats obtenus pour le problème Tai 25b

❑ *Qualité de la solution*

$$\text{Erreur} = \frac{(coûtmoyen - coûtopt)}{coûtopt} * 100$$

$$= \frac{11707658 - 1167256}{1167256} * 100 = 0.3\ \% \qquad \text{Eq 4.3}$$

❖ ***Problème : Sko 56***

✓ *Les paramètres*

$$\begin{cases} \alpha = 1 \\ \beta = 0.2 \\ \rho = 0.2 \\ \eta = 5 \\ \varphi = 0.97 \end{cases}$$

✓ *Les résultats obtenus*

	Coût opt	Cycles/2.6
1 er	34 466	583
2 ème	34 462	6
3 ème	**34 458**	52
4 ème	34 466	577
5 ème	34 520	468
6 ème	**34 458**	20
7 ème	34 466	124
8 ème	34 466	314
9 ème	**34 458**	250
10 ème	34 466	609

Tableau .4.11 - Résultats obtenus pour le problème Sko56

❑ ***Qualité de la solution***

$$\text{Erreur} = \frac{(coûtmoyen - coûtopt)}{coûtopt} * 100$$

$$= \frac{34468.6 - 34458}{34458} * 100 = 0.03\ \% \qquad \text{Eq 4.4}$$

❖ *Problème : Els19*

✓ *Les paramètres*

$$\begin{cases} \alpha = 1 \\ \beta = 0.3 \\ \rho = 0.1 \\ \eta = 1 \\ \varphi = 0.99 \end{cases}$$

✓ *Les résultats obtenus*

	Coût opta	Cycles
1 er	**17 212 548**	1
2 ème	**17 212 548**	54
3 ème	**17 212 548**	3
4 ème	**17 212 548**	5
5 ème	**17 212 548**	42
6 ème	**17 212 548**	7
7 ème	**17 212 548**	135
8 ème	**17 212 548**	5
9 ème	**17 212 548**	29
10 ème	**17 212 548**	23

Tableau4.12 - Résultats obtenus pour le problème Els19

❑ *Qualité de la solution*

$$\text{Erreur} = \frac{(coûtmoyen - coûtopt)}{coûtopt} * 100 = 0\ \% \qquad \text{Eq 4.5}$$

❖ *Problème : Wil50*

✓ *Les paramètres*

$$\begin{cases} \alpha = 1 \\ \beta = 0.2 \\ \rho = 0.2 \\ \eta = 13 \\ \varphi = 0.9 \end{cases}$$

✓ *Les résultats obtenus*

	Coût opt	Cycles
1 er	**48 816**	27
2 ème	**48 816**	109
3 ème	**48 816**	33
4 ème	**48 816**	300
5 ème	**48 816**	85
6 ème	48 824	46
7 ème	**48 816**	42
8 ème	48 824	221
9 ème	**48 816**	129
10 ème	48 824	296

Tableau.4.13 - Résultats obtenus pour le problème Wil50

❑ *Qualité de la solution*

$$\text{Erreur} = \frac{(coûtmoyen - coûtopt)}{coûtopt} * 100$$

$$= \frac{48818.4 - 48816}{48816} * 100 = 0.0049\ \% \qquad \text{Eq 4.5}$$

Les tableaux précédents résument les résultats obtenus par l'application de *l'algorithme à colonies de fourmis* avec la méthode du *plafond dégradé* comme recherche locale (une méthode hybride *colonies de fourmis_plafond dégradé*) sur un ensemble de problème de la littérature.

Pour les problèmes **Sko42**, **Tai20** et **Els19**, sur dix simulations, *colonies de fourmis_plafond dégradé* a atteint dix fois l'optimum (**15 812 ; 703 482 ; 17 212 548**) ce qui donne une qualité égale à **0 %**.

En plus de ces problèmes, cette méthode a été appliquée sur :

Le problème **tai25b** : son optimum a été atteint **4** fois.

Le problème **Sko56** : son optimum a été atteint **3** fois.

Le problème **Wil50** : son optimum a été atteint **7** fois.

Ceci nous a permis de trouver des résultats parfois meilleurs que celles obtenus par certaines autres méthodes, tel que résumé par le tableau suivant :

	Best known value	*TS*	*RTS*	*SA*	*GH*	*HAS-QAP*	*ANtabu*	*Degraded ceiling*
Sko42	*15812*	*0.039*	*1.116*	*0.114*	*0.003*	*0.076*	*0*	*0*
Tai20	*703482*	*0.211*	*0.246*	*0.716*	*0.268*	*0.675*	*0*	*0*
Els19								*0*
Tai25b	*1167256*	*0.51*	*0.345*	*1.002*	*0.629*	*1.189*	*0.736*	*0.3*
Sko56	*34458*	*0.08*	*1.082*	*0.11*	*0.06*	*0.101*	*0.002*	*0.03*
Wil50	*48816*	*0.041*	*0.504*	*0.061*	*0.032*	*0.061*	*0.008*	*0.0049*

Tableau.4.14 - Comparaison des résultats obtenus avec des méthodes de la littérature.

4.5 CONCLUSION

L'analyse de résultats montre que l'algorithme hybride utilisé pour résoudre le problème d'affectation quadratique est performant en comparaison avec les approches existantes. La méthode que nous avons utilisée tient son efficacité de la l'hybridation de deux méta-heuristiques, à savoir les colonies des fourmis et l'algorithme du plafond dégradé.

CHAPITRE 5

CONCLUSION

Ce mémoire s'est concentré sur le problème d'affectation quadratique. Ce dernier est lié à l'aménagement d'usines qui implique la détermination d'un arrangement et d'une configuration optimale des équipements. L'objectif est d'optimiser le coût de production et le bénéfice. Ce problème a été l'objet de nombreuses études pendant les 40 dernières années. Depuis, un certain nombre de modèles et de méthodes de résolution ont été proposés, y compris les méthodes exactes, ainsi que des approches à base de méta-heuristiques. Le problème d'affectation quadratique est un NP-difficile. Nous avons proposé d'utiliser les méta-heuristiques pour le résoudre. On s'est basé principalement sur l'algorithme à colonie de fourmis qui s'inspire des comportements collectifs de dépôt et de suivis de traces des fourmis. Cet algorithme a été hybridé à une variante du recuit simulé qui s'appelle l'algorithme du plafond dégradé.

Nous avons présenté la problématique de l'aménagement d'usines en soulignant des approches de résolution face aux situations industrielles. Tel que nous l'avons signalé, les modèles mathématiques existants ne sont pas capables de résoudre des problèmes de grande

taille, ce qui peut rendre ces approches inutilisables dans le monde industriel réel caractérisé par des problèmes de grande taille. En effet, grâce au développement des méta-heuristiques la taille des cas pouvant être résolus augmente considérablement. Les heuristiques sont reconnues pour donner rapidement une bonne solution à un problème donné. Aussi, une revue de la littérature sur les méta-heuristiques et sur les modèles mathématiques a été présentée au deuxième chapitre.

Dans cette recherche, une hybridation de deux méta-heuristiques *(l'algorithme à colonies de fourmis et l'algorithme de plafond dégradé)* a été employée pour résoudre le problème d'affectation quadratique. La méthodologie a été décrite de façon détaillée au troisième chapitre. L'algorithme utilisé a été codé en utilisant le logiciel *Matlab*. Ce dernier a été utilisé pour mener à terme les expérimentations du quatrième chapitre. Les résultats numériques obtenus montrent que l'heuristique hybridant les colonies de fourmis avec l'algorithme du plafond dégradé, donne des bons résultats pour résoudre le problème d'affectation quadratique.

Comme perspective de ce travail, il serait intéressant de traiter le problème de l'aménagement dynamique par une approche hybride similaire à celle utilisée dans ce mémoire. Une autre perspective consiste à explorer d'autres types d'hybridation pour les problèmes d'aménagements statique et dynamique. Enfin, nous allons chercher dans le futur à résoudre le « vrai » problème d'aménagement d'usines en ajoutant des équations de contraintes à la formulation de type PAQ étudié dans ce travail de maîtrise.

BIBLIOGRAPHIE

Ahuja, R.K., Orlin, J.B., and Tiwari, A., 2000, "*A greedy Genetic Algorithm for The Quadratic Assignment Problem,* " *Computers and Operations Research*, Vol. 27, pp. 917-934.

A. Kusiak and S. S. Heragu. *''The facility Layout Problem''.* European Journal of Operational Research, 29:229–251, 1987.

Banerjee, P., Montreuil, B., Moodie, C.L., and Kashyap, R.L., 1992, *''A Modeling of Interactive Facilities Layout Designer Reasoning Using Qualitative Patterns,''* International Journal of Production Research, Vol. 30, No. 3, pp. 433-453.

Burkard, R. E. (1973). Operation Research Verfahren 16,84-108.

Burkard, R. E., and Rendl, F. (1984). *''A Thermodynamically Motivated simulation procedure for Combinatorial Optimization Problems''*. European Journal of Operational Research, 17, 169-174

Bazarra, M. S. (1975). '' *Computerized Layout Desing : A Branch And Aound Approach''.* AIIE Transactions 7(4), 432-437.

Bazarra, M. S., and Elshafei, A. N. (1979). '' *An Exact Branch and Bound Procedure for Quadratic Assignment Problems*''. Naval Research Logistics Quarterly, 26, 109

Bazarra, M.S. and Sherali, M.D., 1980, "*Bender's Partitioning Scheme Applied to A New Formulation of the Quadratic Assignment Problem*," *Naval Research Logistics Quarterly*,Vol. 27, No. 1, pp. 29-41.

Block, T.E., 1978, "*Fate: A New Construction Algorithm For Facilities Layout*," Journal of Engineering Production, Vol. 2, pp. 111-120.

Burkard, R.E. and Bonniger, T., 1983, "*A heuristic for Quadratic Boolean Program With Application to Quadratic Assignment Problems*," European Journal of Operational Research, Vol. 13, pp. 374-386.

Carrie, A.S., Moore, J.M., Roczniak, M., and Seppanen, J.J., 1978, *"Graph Theory And Computer-Aided Facilities Design*," Omega, Vol. 6, No. 4, pp. 353-361.

Connolly, D.T. (1990), ''*An Improved Annealing Scheme For Finding a Maximum Weight Planar Subgraph*'' European Journal of Operational Research, 20(1),102-114.

Deisenroth, M.P. and Apple, J.M., 1972, "*A Computerized Plant Layout Analysis and Evaluation Technique*," Technical Paper, Annual AIIE Conference, Norcross, GA.

Dorigo M., V. Maniezzo & A. Colorni (1996), ''*Ant system: Optimization By a Colony Of Cooperating Agents*''

Eades, P., Foulds, L.R., and Giffin, J., 1982, "*An Efficient Heuristic For Identifying a Maximum Weight Planar Subgraph*," In: Lecture Notes in Mathematics No. 952 (Combinatorial Mathematics IX), Springer, Berlin.

Edwards, H.K., Gillet, B.E., and Hale, M.C., 1970, "*Modular Allocation technique (MAT),*" *Management Science*, Vol.17, No. 3, pp. 161-169.

Faigle, U., and Kern, W. (1992). ''*Some Convergence Results for Probabilistics Tabu Search*''. ORSA Journal on Computing, 4, 32-37.

Fleurent, C. and Ferland, J.A., 1994, "*Genetic Hybrids For The Quadratic Assignment Problem,*" *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, Vol. 16, pp. 173-188.

Foulds, L.R. and Robinson, D.F., 1976, "*A Strategy For Solving The Plant Layout Problem,*" *Operations Research*, Vol. 27, No. 4, pp. 845-855.

Foulds, L.R. and Robinson, D.F., 1978, "*Graph Theoretic Heuristics For The Plant Layout Problem,*" *International Journal of Production Research*, Vol. 16, No.1, pp. 27-37.

Fox, B. (1993). ''*Integrating And Accelerating Tabu Search, Simulated Annealing And Genetic Algorithms*'', F. Glover, T. Taillard, M. Laguna, D. De Werra (Eds.).

Tabu Search, Annals of Operations Research, Baltzar, Basel,47-67.

Gilmore, P.C., 1962, "*Optimal And Suboptimal Algorithms For The Quadratic Assignment Problem,*" *Journal of the Society for the Industrial and Applied Mathematics*, Vol. 10, pp.305-313.

Gavett, J. W., and Plyter, N. V. (1966). ''*The Optimal Assignment Of Facilities To Locations By Branch and Bound. Operation*''s Research, 14, 210-232.

Glover, F. (1986). ''*Future Paths for Integer Programming And Links To Artificial Intelligence. Computers and operations*'' Research, 5, 533-549.

Glover, F. (1989).''*Tabu Search, Part I. ORSA*'' Journal on Computing, 1(3),190-206

Glover, F. (1992). ''*Ejection Chains, Reference Structures And Alternating Path Methods For Travelling Salesman Problems*''. University of Colorado.

Golden, B. L., & Skiscim, C. C. (1986). ''*Using Simulated Annealing To Love Routing & Location Problems*''. Naval Research Logistics Quarterly, 33, 261-279

Goetschalckx, M., 1992, "*SPIRAL: An efficient And Interactive Adjacency Graph Heuristic for Rapid Prototyping Of Facilities Design*," *European Journal Of Operational Research*, Vol. 63, No. 2, pp. 304-321.

Hansen, P. (1986). ''*The Stepest Ascent Mildest Descent Heuristic For Combinatorial Programming*''. Talk Tresented At The Congress on Numerical Methods in Combinatorial Optimization, Capri.

Hassan, M.M.D., Hogg, G.L., and Smith, D.R., 1986, "*SHAPE: AConstruction Algorithm For Area Placement Evaluation*," International Journal of Production Research, Vol. 24, pp. 1283-1295.

Hassan, M.M.D. and Hogg, G.L., 1987, "*A Review Of Graph Theory Applications To The Facilities Layout Problem*," Omega, Vol. 14, No. 4, pp. 291-300.

Hassan, M.M.D. and Hogg, G.L., 1989, "*On Converting a Dual GraphInto a Block Layout*," *International Journal of Production Research*, Vol. 27, No.7, pp. 1149-1160.

Heragu, S.S., and Alfa, A. S. (1992). ''*Experimental Analystis Of Simulated Annealing Fased Algorithms For The Facility Layout Problems*''. *European Journal of Operational Research*, 57(2), 190-202

Heragu, S.S. and Kusiak, A., 1991, "*Efficient Models For The Facility Layout Problem*," *European Journal of Operational Research*, Vol. 53, No. 1, pp. 1-13.

Heragu, S.S., 1992, "*Recent Models And Techniques For Solving The Layout Problem*," *European Journal of Operational Research*, Vol. 57, No. 2, pp. 136-144.

Hillier, F.S. and Connors, M.M., 1966, "*Quadratic Assignment ProblemAlgorithms And The Location Of Indivisible Facilities*," Management Science, Vol. 13, pp. 42-57.

Johnson, D. S., Argaon, C. R., McGeoch, L. A., and Schevon, C. (1989).
''Optimization By Simulated Annealing: An Experimental Evaluation. Part I, Graph Partitioning, Operations Research, 37(6), 865-892

Kirkpatrick, S., Gelatt, C. D., And Veechi, M. P. (1983). ''*Optimization By Simulated Annealing*'', Science, 220(4598), 671-680.

Kaku, B.K. and Thompson,G.L., 1986, "*An Exact Algorithm For The General Quadratic Assignment Problem*," *European Journal of Operational Research*, Vol. 23, pp. 382-390.

Kouvelis, P., Chiang, W. (1992). ''*Simulated Annealing Procedure For Single Row Layout Problems In Flexible Manufacturing Systems*'' European Journal of Operational Research, 57(2), 203-223.

Kouvelis, P., Chiang, W., And Fitzsimmons, J. (1992). *''Simulated Annealing Procedure For Machine Layout Problems In The Presence Of Zoning Constraints''* European Journal of Operational Research, 57(2), 203-223.

Kouvelis, P., and kiran, A.S. (1990). *''The Plant Layouat Problem In Automated Manufacturing Systems. Annals Of Operations''* Research, 26, 379-412.

Lawler, E.L., 1963, *"The Quadratic Assignment Problem,"* *Management Science*, Vol. 9,No. 4, pp. 586-599.

Land, A. H. (1963). *''A Problem Of Assignment With Inter Related Costs. Operations Research Quarterly''*, 14, 185-198

Lee, R. and Moore, J.M., 1967, *"Corelap-Computerized Relationship Layout Planning,"* Journal Of Industrial Engineering, Vol.18, pp. 195-200.

Meller, R. D., and Bozer, Y. A. (1996). *''A new Simulated AnnealingAlgorithm For The Facility Layout Problem''.International Journal of Productionl Research*, 34(6), 1675-1692.

Muther, R. and McPherson, K., 1970, *"Four Approaches To Computerized Layout Planning,"* Industrial Engineering, February, pp. 39-42.

Montreuil, B., Ratliff, H.D., and Goetschalckx, M., 1987, *"Matching Based Interactive Facility Layout," IIE Transactions*, Vol. 19, No. 3, pp. 271-279.

Montreuil, B., 1990, "*A Modelling Framework For Integrating Layout Design And Flow Network Design*," *Proceedings of the Material Handling Research Colloquium*, Hebron,KY, pp. 43-58.

Nourelfath, M., Nahas, N., et Montreuil, B., 2007, "Coupling ant colony optimization and the extended great deluge algorithm for the discrete facility layout problem", *Engineering Optimization*, Vol. 39, Issue 8 pp. 953 - 968.

Neghabat, F., 1974, "*An efficient Equipment Layout Algorithm*," Operations Research, Vol. 22, pp. 622-628.

Nugent, C.E., Vollmann, T.E., and Ruml, J. (1968). ''*An Experimental Comparison Of Techniques For The Assignment Of Facilities To Locations. Operations*'' Research, 16, 150-173.

O'Brien, C. and Abdel Barr, S.E.Z., 1980, "*An Interactive Approach To Computer Aided Facility Layout*," International *Journal of Production Research*, Vol. 18, No. 2, pp. 201 211.

R. D. Meller and K-Y. Gau. ''*The Facility Layout Problem: Recent And Emerging Trends And Perspectives*''. *Journal of Manufacturing Systems*, 15:351–366, 1996.

Pierce, J. F., and Crowston, W. B. (1971). '' *Tree-Search Algorithms For Quadratic Assignment Problems*'' . Naval Research Logistic Quarterly, 18, 1-36.

Seehof, J.M. and Evans, W.O., 1967, "*Automated Layout Design Program*," The Journal Of Industrial Engineering, Vol. 18, No. 2, pp. 690-695.

Seppannen, J.J. and Moore, J.M., 1970, "*Facilities Planning With Graph Theory*,"*Management Science*, Vol. 17, No. 4, pp. 242-253.

Seppannen, J.J., and Moore, J.M., 1975, "*String Processing Algorithms For Plant Layout Problems*," *International Journal of Production Research*, Vol. 13, No. 3, pp. 239-254.

Skorin-Kapov, J. (1990). ''*Tabu Search Applied To The Quadratic Assignment Problem*''. ORSA Journal On Computing 2(1), 33-45

Suresh, G. and Sahu, S. (1993). ''*Multiobjective Facility Layout Using Simulated Annealing*''. *International Journal of Production Economics, 32, 239-254.*

Suresh, G., Vinod, V.V., and Sahu, S., 1995, "*A Genetic Algorithm For Facility Layout,*" *International Journal of Production Research*, Vol. 33, No. 12, pp. 3411-3423.

Taillard, E. (1991). ''*Robust Taboo Search For Thee Quadratic Assignment Parallel Computing*'', 17, 443-455.

Tate, D.E. and Smith, A.E., 1995a, "*A Genetic Approach To The Quadratic Assignment Problem*," Computers and Operations Research, Vol. 22, pp.73-83.

Van Camp, D.J., Carter, M.W., and Vannelli, A., 1991, "*A Nonlinear Optimization Approach For Solving Facility Layout Problems*," European Journal of Operational Research, Vol. 57, pp. 174-189.

Werra, D., and Hertz, A. (1989). ''Tabu Search Techniques : A Tutorial And An Application To Neural Network. OR Spektrum 11, 131-141.

Wilhelm, M. R., And Ward, T. L. (1987). '' *Solving Quadratic Assignment Problems By Simulated Annealing*''. IIE Transactions, 19, 107-119

Zoller, K. and Adendroff, K., 1972, "*Layout Planning By Computer Simulation,*" AIIE Transactions, Vol. 4, No. 2, pp. 116-125.

ANNEXE

Une application numérique de notre algorithme

On prend le problème Els19 avec les deus matrices de données suivantes :

Matrice Flux=

```
[0   12   36   28   52   44 110 126   94   63 130 102   65   98 132 132 126 120 126;
 12    0   24   75   82   75 108   70 124   86   93 106   58 124 161 161   70   64   70;
 36   24    0   47   71   47 110   73 126   71   95 110   46 127 163 163   73   67   73;
 28   75   47    0   42   34 148 111 160   52   94 148 49 117 104 109 111 105 111;
 52   82   71   42    0   42 125 136 102   22   73 125   32   94 130 130 136 130 136;
 44   75   47   34   42    0 148 111 162   52   96 148   49 117 152 152 111 105 111;
110 108 110 148 125 148    0   46   46 136   47   30 108   51   79   79   46   47  14;
126   70   73 111 136 111   46    0   69 141   63   46 119   68 121 121   27   24   36;
 94 124 126 160 102 162   46   69    0 102   34   45   84   23   80   80   69   64   51;
 63   86   71   52   22   52 136 141 102    0   64 118   29   95 131 131 141 135 141;
130   93   95   94   73   96   47   63   34   64    0   47   56   54   94   94   63   46   24;
102 106 110 148 125 148   30   46   45 118   47    0 100   51   89   89   46   40   36;
 65   58   46   49   32   49 108 119   84   29   56 100    0   77 113 113 119 113 119;
 98 124 127 117   94 117   51   68   23   95   54   51 77    0   79   79   68   62   51;
132 161 163 104 130 152   79 121   80 131   94   89 113   79    0   10 113 107 119;
132 161 163 109 130 152   79 121   80 131   94   89 113   79   10    0 113 107 119;
126   70   73 111 136 111   46   27   69 141   63   46 119   68 113 113    0    6 24;
120   64   67 105 130 105   47   24   64 135   46   40 113   62 107 107    6    0   12;
126   70   73 111 136 111   41   36   51 141   24 36 119   51 119 119   24   12    0]
```

Matrice Distance=

```
0 76687     0   415   545   819   135  1368   819  5630     0  3432  9082  1503     0     0
13732  1368  1783;
76687     0 40951  4118  5767  2055  1917  2746  1097  5712     0     0     0   268     0
1373   268     0     0;
    0 40951     0  3848  2524  3213  2072  4225   566     0     0   404  9372     0   972
0 13538  1368     0;
  415  4118  3848     0   256     0     0     0     0   829   128     0     0     0     0
0     0     0     0;
  545  5767  2524   256     0     0     0     0    47  1655   287     0    42     0     0
0   226     0     0;
  819  2055  3213     0     0     0     0     0     0   926   161     0     0     0     0
0     0     0     0;
  135  1917  2072     0     0     0     0     0   196  1538   196     0     0     0     0
0     0     0     0;
 1368  2746  4225     0     0     0     0     0     0     0   301     0     0     0     0
0     0     0     0;
  819  1097   566     0    47     0   196     0     0  1954   418     0     0     0     0
0     0     0     0;
 5630  5712     0   829  1655   926  1538     0  1954     0     0   282     0     0     0
0     0     0     0;
    0     0     0   128   287   161   196   301   418     0     0  1686     0     0     0
0   226     0     0;
 3432     0   404     0     0     0     0     0     0   282  1686     0     0     0     0
0     0     0     0;
 9082     0  9372     0    42     0     0     0     0     0     0     0     0     0     0
0     0     0     0;
 1503   268     0     0     0     0     0     0     0     0     0     0     0     0     0
0     0     0     0;
    0     0   972     0     0     0     0     0     0     0     0     0     0     0     0
99999     0     0     0;
    0  1373     0     0     0     0     0     0     0     0     0     0     0     0 99999
0     0     0     0;
13732   268 13538     0   226     0     0     0     0     0   226     0     0     0     0
0     0     0     0;
 1368     0  1368     0     0     0     0     0     0     0     0     0     0     0     0
0     0     0     0;
 1783     0     0     0     0     0     0     0     0     0     0     0     0     0     0
0     0     0     0];
```

L'exécution du programme *Matlab* donne le résultat suivant pour le (Els19) par exemple :

17	18	19	11
12	9	3	14
1	2	10	13
7	5	16	15
8	6	4	