République Algérienne Démocratique et Populaire

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
Faculté des Sciences Exactes
Département de Recherche Opérationnelle

Mémoire de fin de cycle

En vue de l'obtention du diplôme de Master en Mathématiques Appliquées

Option : Modélisation Mathématique et Téchniques de Décision

Thème

Résolution du problème de cueillette et livraison avec fenêtres de temps à un seul véhicule par des méta-heuristiques

---

Présenté par

M^{lle} Ferroudj Celia M^{lle} Ichallal Lamia ;

Soutenu le 03/07/2019 devant le jury composé de :

Présidente	M.C.A	M^{me} AOUDIA Fazia
Promotrices :	M.C.B	M^{me} HALIMI-YOUSFI Naouel
	M.A.A	M^{lle} AOUDIA Zohra
Examinatrice	M.C.B	M^{me} TOUCHE-ANZI Aicha
Examinatrice	Maitre de Recherche.B.CREAD	M^{lle} IDRES Lahna

Promotion 2018-2019

Remerciements

En premier lieu, nous remercions Dieu tout puissant de nous avoir accordé santé, courage et volonté pour accomplir ce travail.

Nous tenons à exprimer nos vifs remerciements à toutes les personnes qui nous ont soutenues, et qui n'ont pas cessé de nous donner des conseils en signe de reconnaissance et qui ont contribué de prés ou de loin à la réalisation de ce mémoire.

Nous tenons à remercier particulièrement notre promotrice Mme HALIMI-YOUSFI Naouel pour son encadrement.

Nous remercions également Mlle AOUDIA Zohra, notre co-promotrice.

Nous exprimons notre grand respect aux honorables membres de jury qui ont accepté d'évaluer ce travail.

Notre plus grande reconnaissance est pour nos parents pour leur présence et leur soutient.

Dédicaces

***J**e tiens à dédier ce travail à :*

***M**a mère pour le don divin d'amour et d'affection qu'elle ma toujours portée, et sont soutient infini durant toute ma vie ;*
***M**on père qui ma soutenu et ma encouragé durant ma carrière d'études ;*
***A** tous mes fères :Abd ghani, Hocin et a ma soeure Imene qui n'ont cessé d'etre pour moi des exemples ;*
***A** toute ma famille ;*
***A** mes chèr(e)s ami(e)s ;*
***A** ma chère binôme Celia et toute sa famille ;*
***A** toutes personnes qui m'ont soutenues et participées de prés ou de loin à l'élaboration de ce mémoire.*

Lamia

***J**e tiens à dédier ce travail à :*

***M**a mère pour le don divin d'amour et d'affection qu'elle ma toujours portée, et sont soutient infini durant toute ma vie, qui ma soutenu et ma encouragé durant ma carrière d'études ;*
***A** mes deux fères Akli et Maksen et a ma soeure Sonia qui n'ont cessé d'etre pour moi des exemples, je leurs souhaite une bonne continuation dans leurs vie ;*
***A** toute ma famille ;*
***A** mes chèr(e)s ami(e)s : Razika, Fraoussen, Fares, Mitecho, Anis ;*
***A** ma chère binôme Lamia et toute sa famille ;*
***A** toutes personnes qui m'ont soutenues et participées de prés ou de loin à l'élaboration de ce mémoire.*

Celia

TABLE DES MATIÈRES

TABLE DES FIGURES

CHAPITRE 1

GÉNÉRALITÉS SUR LES PROBLÈMES D'OPTIMISATION

Dans ce chapitre nous allons présenter les problèmes d'optimisation combinatoire et le principe des algorithmes génétiques en détaillant leurs caractéristiques et leurs mécanismes. On s'intéresse ausssi a une nouvelles méta-heuristiques colonie de fourmis.

1.1 Introduction sur les problèmes d'optimisation combinatoire

L'optimisation est un thème central en recherche opérationnelle, un grand nombre de problèmes d'aide à la décision peuvent en effet être décrits sous la forme de problèmes d'optimisation. Les problèmes d'identification, d'apprentissage supervisé de réseaux de neurones ou encore la recherche du plus court chemin sont, par exemple, des problèmes d'optimisation.
Nous définissons dans ce qui suit l'espace de définitions d'un problème d'optimisation ainsi que ces différentes caractéristiques et la marche à suivre face à un problème d'optimisation.

1.1.1 Définition d'un problème d'optimisation

1. **Un espace de recherche (de décision)** : est un ensemble de solutions ou de configurations finis ou dénombrables constitué des différentes valeurs prises par **les variables de décision**. Ces derniers peuvent être de nature diverse (réelle, entiere, booléenne ...etc.) et exprimer des données qualitatives ou quantitatives.
2. Une ou plusieurs **fonction(s)** dite **objectif(s)** à optimiser (minimiser ou maximiser).
3. Un ensemble de **contraintes** à respecter. Cet ensemble définit des conditions sur l'espace d'état que les variables doivent satisfaire. Ces contraintes sont souvent des contraintes d'inégalité ou d'égalité ($<>$ *ou* $=$) et permettent en général de limiter l'espace de recherche (solutions réalisables).

La résolution optimale du problème consiste à trouver un point ou un ensemble de points de l'espace de recherche qui réalise la meilleur valeur de la fonction objective. Le résultat est appelé **valeur optimale** ou **optimum** [24].

1.1.2 Classification d'un problème d'optimisation

Les problèmes d'optimisation sont classées selon :

1. Le domaine des variables de décision est :
   - Soit Continu et on parle alors de problème continu.
   - Soit discret et on parle donc de **problème combinatoire**.
2. La nature de la fonction objective à optimiser :
   - Soit linéaire et on parle alors de problème linéaire.
   - Soit non linéaire et on parle donc de problème non linéaire.
3. Le nombre de fonctions objectifs à optimiser :
   - Soit une fonction scalaire et on parle alors de problème **mono-objectif**.
   - Soit une fonction vectorielle et on parle donc de problème **multiobjectif**.
4. La présence ou non des contraintes :
   - on parle de problème **sans contraintes**.
   - on parle de problème **avec contraintes**.
5. Sa taille :
   - problème de **petite taille**.
   - problème de **grande taille**.
6. L'environnement :
   - **dynamique**.
   - **statique**.

1.1.3 Définition d'un problème d'optimisation combinatoire

Le domaine de l'optimisation combinatoire est un domaine très important, il est situé au carrefour de la recherche opérationnelle, mathématique et informatique. Son importance s'explique par la grande difficulté posée par les problèmes d'optimisation d'une part, et par le nombre important des applications pratiques pouvant être formulées sous forme de problèmes d'optimisation combinatoires.
On qualifie généralement de "combinatoires" les problèmes dont la résolution se ramène à l'examen d'un nombre fini de combinaisons. Bien souvent cette résolution se heurte à une explosion du nombre de combinaisons à explorer.
Un problème d'optimisation combinatoire peut être défini comme suit :
Etant donné un ensemble X de combinaisons, et une fonction $f : X \longrightarrow \mathbb{R}$ il s'agit de trouver la combinaison de X minimisant f ,i.e. ;$x^* \in X$.

$$f(x^*) \leq f(x_i) \quad ,\forall x_i \in X$$

1.2 Approches de résolution des problèmes d'optimisation combinatoire

Résoudre un problème d'optimisation combinatoire nécessite l'étude de trois points suivants[32] :

- La définition de l'ensemble des solutions réalisables des contraintes à respecter.
- L'expression de l'objectif à optimiser.
- Le choix de la méthode d'optimisation à utiliser.

Les deux premiers points relèvent de la modélisation du problème, le troisième de sa résolution. Ceci ne peut être fait qu'avec une bonne connaissance du problème étudié et de son domaine

d'application.
Le choix de l'objectif à optimiser requiert également une bonne connaissance du problème.
La définition de la fonction objective mérite toute l'attention de l'analyste car rien ne sert de développer de bonnes méthodes d'optimisation si l'objectif à optimiser n'est pas bien défini.
Les méthodes d'optimisation combinatoire peuvent être réparties en deux grandes classes [16] :

- Les méthodes exactes.
- Les méthodes approchées.

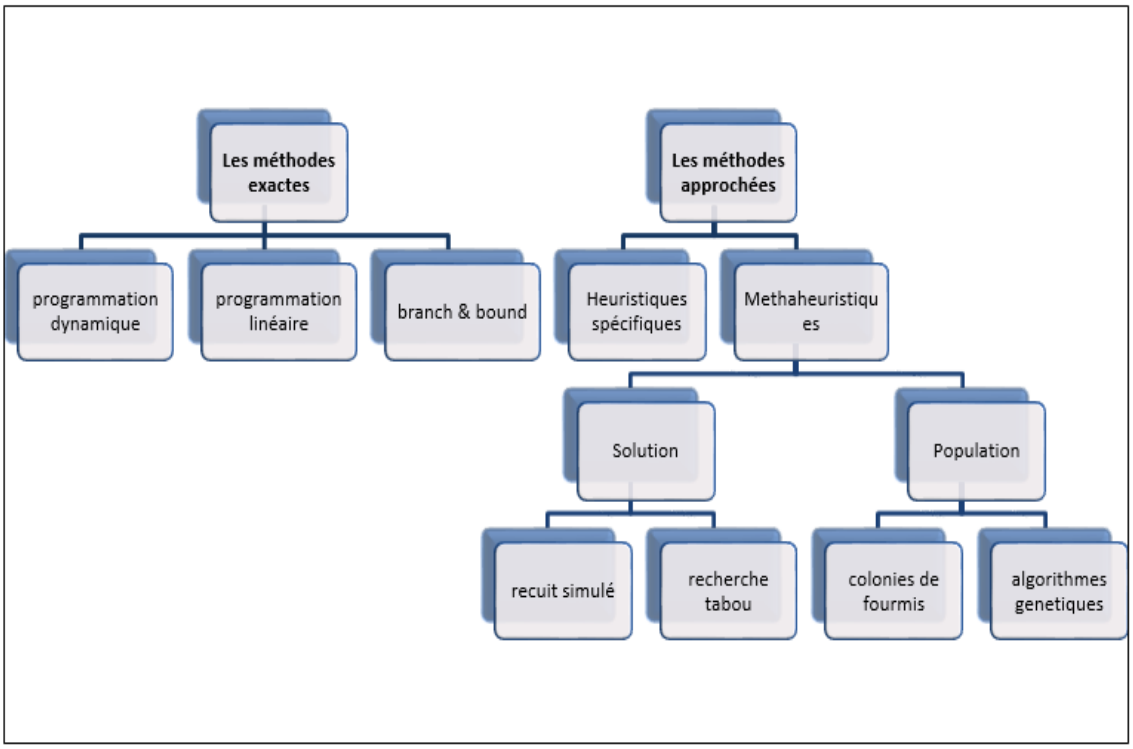

FIGURE 1.1 – Classification des méthodes d'optimisation combinatoires.

1.2.1 Les méthodes exactes

Les algorithmes exacts sont utilisés pour trouver au moins une solution optimale d'un problème. Les algorithmes exacts les plus réussis dans la littérature appartiennent aux paradigmes des trois grandes classes suivantes [27] :

- La programmation dynamique.
- La programmation linéaire.
- Les méthodes de recherche arborescente (Branch and bound).

1.2.1.1 Branch and bound

La méthode branch and bound (procédure par évaluation et séparation progressive) consiste à énumérer ces solutions d'une manière intelligente en ce sens que, en utilisant certaines propriétés du problème en question, cette technique arrive à éliminer des solutions partielles. De ce fait, on arrive souvent à obtenir la solution recherchée à réduire le temps de résolution en des temps raisonnables. Bien entendu, dans le pire cas, on retombe toujours sur l'élimination explicite de toutes les solutions du problème [16].
Pour ce faire, cette méthode se dote d'une fonction qui permet de mettre une borne sur certaines solutions pour soit les exclure soit les maintenir comme des solutions potentielles. Bien entendu,

La performance d'une méthode branch and bound dépend, entre autres, de la qualité de cette fonction (de sa capacité d'exclure des solutions partielles tôt).

✓ **Algorithme général :** [16]

Début
1 Définir $L = X$ et initialiser $\hat{x}$.
2 Tant que $L \neq \emptyset$.
3 Sélectionner un sous-problème S de L à explorer.
4 Si une solution $\hat{x}' \in x \in S \mid f(x) < f(\hat{x})$ peut être trouvée $\hat{x} = \hat{x}'$.
5 Si S ne peut pas être élagué :
6 Partitionner S en S_1,S_2,...,S_r.
7 Insérer S_1,S_2,...,S_r dans L.
8 Retirer S de L.
9 Finsi
10 Finsi
11 Fintant que
12 Retour $\hat{x}$.
Fin

1.2.1.2 La programmation dynamique

La programmation dynamique a été introduite depuis 1940 par Richard Bellman. Le concept de base est simple : une solution optimale est le résulta obetenie à partir des solution des sous-problèmes résolus de façon optimale. Il faut donc diviser un problème donné en sous-problèmes et les résoudre un par un [16].

✓**Algorithme général de programmation dynamique [16] :**

Début
1 Caractériser la structure d'une solution optimale.
2 Définir récursivement la valeur d'une solution optimale.
3 Calculer la valeur d'une solution optimale en remontant progressivement jusqu'à L'énoncé du problème initial.
4 Construire une solution optimale pour les informations calculées.
Fin

1.2.2 Les méthodes approchées ou heuristiques

Une méthode heuristique ou approchée est une méthode d'optimisation qui a pour but de trouver une solution réalisable de la fonction objectif en un temps raisonnable, mais sans garantire l'optimalité. L'avantage principale de ces méthodes est qu'elles peuvent s'appliquer à n'importe quelle classe de problèmes, faciles ou très difficiles. D'un autre côté les algorithmes d'optimisation tels que les algorithmes de recuit simulé, les algorithmes de recjerche tabous et les algorithmes génétiques ont démontré leurs robustesses et efficacités face à plusieurs problèmes d'optimisation combinatoires.
Les méthodes approchées englobent deux classes [31] :

- Les méthodes constructives.
- Les métaheuristiques.

1.2.2.1 Les méthodes constructives

Ce sont des méthodes itératives qui construisent pas à pas une solution. Partant d'une solution partielle initialement vide, elles cherchent à étendre à chaque étape la solution partielle de l'étape précédente, et ce processus se répète jusqu'à ce que l'on obtienne une solution complète[16].

- **Utilisation** : Les méthodes constructives sont généralement utilisables quand la qualité de solution n'est pas un facteur primordial ou la taille de l'instance est raisonnable, en l'occurrence pour générer une solution initiale dans une métaheuristique ou une méthode exacte, ces méthodes sont rapides et faciles a mettre en oeuvre.

1.2.2.1.1 L'Algorithme glouton

Il consiste à constuire une solution réalisable en se ramenant à une suite de décisions qu'on prend à chaque fois au mieux à l'aide d'un critère local sans remettre en question les décisions déjà prises. Généralement, la solution obtenue est approchée[16].

- **Avantage :**algorithmes simples à implémenter.
- **Inconvénient :** solutions approchées obtenues plus ou moins bonnes (critère local).

1.2.2.1.2 Heuristiques $2-$opt, $3-$opt et $k-$opt

Les heuristiques d'échange d'arcs comme $2-$opt et $3-$opt sont largement utilisées pour l'amélioration des solutions dans les problèmes de routage de véhicules. Généralement, ce genre d'heuristiques d'échange sont incorporé dans des structures de recherche locale de la manière suivante [1] :

Début

1 Commencer avec une solution initiale définie aussi comme solution courante.
2 Générer le voisinage total de la solution courante en appliquant l'heuristique d'échange considérée.
3 Sélectionner la meilleure solution du voisinage et la définir comme solution courante.
4 Revenir à l'étape 2.

Fin

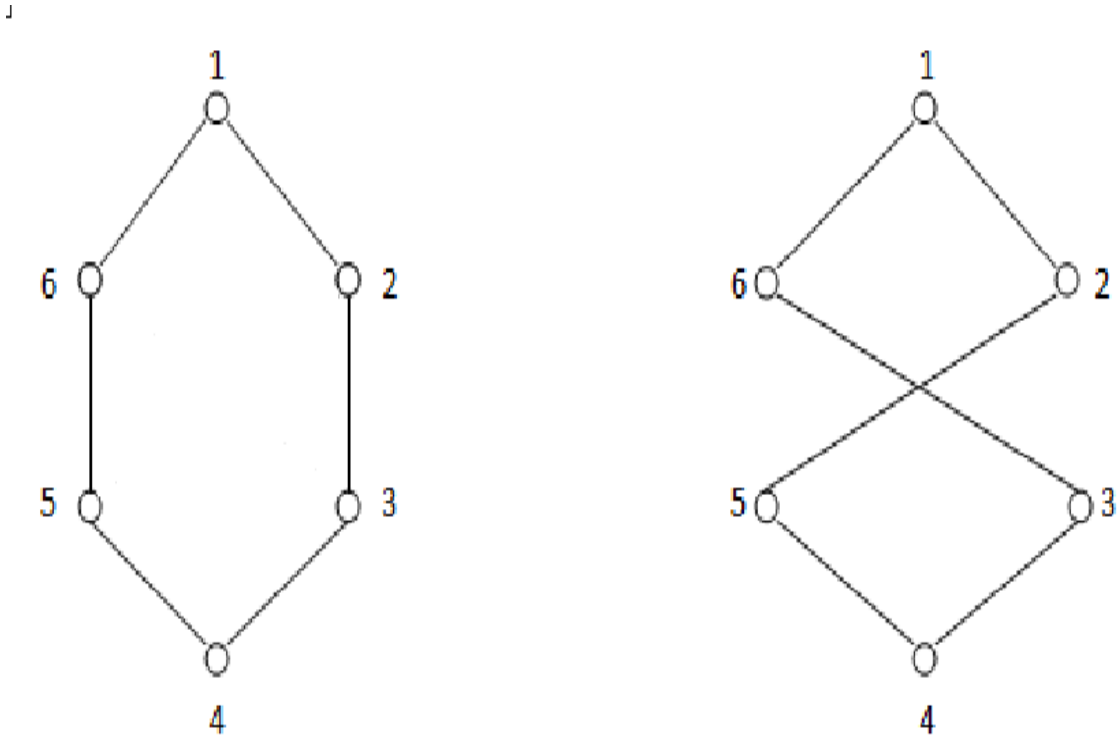

FIGURE 1.2 – Exemple d'échange 2−opt appliqué à deux cycles.

1.2.2.2 Les Métaheuristiques

Le mot métaheuristique est dérivé de la composition de deux mots grecs [16]) :

- **heuristique** qui vient du verbe heuriskein et qui signifie 'trouver'.
- **méta** qui est un suffixe signifiant 'au -delà', 'dans un niveau supérieur'.

Les métaheuristiques sont des méthodes inspirées de la nature, ce sont des heuristiques modernes dédiées à la résolution des problèmes et plus particulièrement des problèmes d'optimisation, qui visent à atteindre un optimum global généralement enfoui au milieu de nombreux optima locaux. Les métaheuristiques se subdivisent en deux sous-classes :

- Les méthodes de voisinage.
- Les méthodes évolutives.

1.2.2.2.1 Les méthodes de voisinage

Ces méthodes partent d'une solution initiale (obtenue de façon exacte, ou par tirage aléatoire) et s'en éloignent progressivement, pour réaliser une trajectoire, un parcours progressif dans l'espace des solutions.
On retrouve entre autre :

A. Le recuit simulé.

B. La recherche Tabou.

A. Le recuit simulé

L'idée principale du recuit simulé tel qu'il a été proposé par Metropolis en 1953 est de simuler le comportement de la matière dans le processus du recuit très largement utilisé dans la métallurgie. Le but est d'atteindre un état d'équilibre thermodynamique, cet état d'équilibre (où l'énergie est minimale) représente - dans la méthode du recuit simulé – la solution optimale

d'un problème ; L'énergie du système sera calculé par une fonction coût (ou fonction objectif). La méthode va donc essayer de trouver la solution optimale en optimisant une fonction objectif, pour cela, un paramètre fictif de température a été ajouté par Kirk patrick, Gelatt et Vecchi. En gros le principe consiste à générer successivement des configurations à partir d'une solution initiale S_0 et d'une température initiale T_0 qui diminuera tout au long du processus jusqu'à atteindre une température finale ou un état d'équilibre (optimum global)[7].

B. La recherche tabou

L'algorithme de Recherche Tabou a été introduit par Glover en 1986. Le but de cette méthode est d'inculquer aux méthodes de recherche locale un surcroît d'intelligence. L'idée est d'ajouter au processus de recherche une mémoire qui permette de mener une recherche plus "intelligente" dans l'espace des solutions. Comme l'algorithme de recuit simulé, la méthode de recherche tabou fonctionne avec une seule configuration courante, qui est actualisée au cours des itérations successives. La nouveauté est que, pour éviter le risque de retour à une configuration déjà visitée, on tient à jour une liste de mouvements interdits, appelée "liste tabou". Cette liste contient m mouvements $(t \rightarrow s)$ qui sont les inverses des m derniers mouvements $(s \rightarrow t)$ effectués. L'algorithme modélise ainsi une forme primaire de mémoire à court terme [10].

1.2.2.2.2 Les méthodes évolutives

Ces méthodes se distinguent de celles déjà présentées par le fait qu'elles opèrent sur une population de solutions, pour cela, elles sont souvent appelées des méthodes à base de population. Certaines d'entre elles ont des principes inspirés de la génétique et du comportement des insectes. La complexité de ces deux phénomènes biologiques a servi de modèle pour des algorithmes toujours plus sophistiqués ces vingt dernières années. Nous allons nous intéresser principalement aux algorithmes génétiques et aux algorithmes de colonie de fourmis.
Etant donnée que dans notre approche nous allons utiliser les méthodes de colonies de foumis ainsi que la méthode des Algo génétiques, nous allons donc leur consacrer la section (1.5) pour bien les détailler.

1.3 Optimisation multi-objectifs

Dans la plupart des problèmes du monde réel, il ne s'agit pas d'optimiser seulement un seul critère mais plutôt d'optimiser simultanément plusieurs critères et qui sont généralement conflictuels. Dans les problèmes de conception, par exemple, il faut le plus souvent trouver un compromis entre des besoins technologiques et des objectifs de coût. L'optimisation multi-objective consiste donc à optimiser simultanément plusieurs fonctions. La notion de solution optimale unique dans l'optimisation uni-objectif disparait pour les problèmes d'optimisation multi objectif au profit de la notion d'ensemble de solutions Pareto optimales[4].

1.3.1 Les problèmes d'optimisation multiobjectifs

Un problème d'optimisation avec objectifs multiples peut être représenté par [10] :

$$\begin{array}{ll} Minimiser & F(x) = (f_1(x), f_2(x), ...f_k(x)). \\ s.c. & x \in \{X \subseteq \mathbb{R}^{\mathbb{D}}\} \end{array} \tag{1.1}$$

où $x = (x_1, x_2, ..., x_D)$ est une position dans l'espace de recherche $f_i : \mathbb{X} \rightarrow \mathbb{R}$, $i = 1, ..., k$; sont les fonctions objectifs du problème. La figure suivante représente le problème d'optimisation multiobjectif [24] :

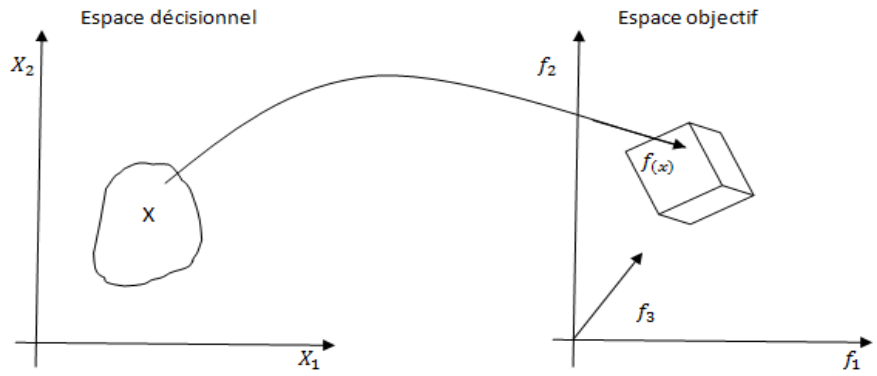

FIGURE 1.3 – Problème d'optimisation multiobjectif (2 variables de décision et 3 fonctions objectifs)

Les problèmes multiobjectifs ont la particularité d'être beaucoup difficiles à traiter. La difficulté réside dans l'absence d'une relation d'ordre total entre les solutions car $\mathbb{R}^k$ ne possède pas de relation d'ordre total si $k \neq 1$. Une solution peut être meilleure qu'une autre sur certains objectifs et moins bonne sur les autres. Donc il n'existe généralement pas une solution unique qui procure simultanément la solution optimale pour l'ensemble des objectifs. Voilà pourquoi le concept de solution optimale devient moins pertinent en optimisation multiobjectif. Dans ce cas la solution optimale ou de bonne qualité n'est plus une solution unique mais, un ensemble de **solutions compromis** entre les différents objectifs à optimiser. Il est vital pour identifier ces meilleurs compromis de définir une relation d'ordre entre ces éléments. La plus célèbre et la plus utilisée est la relation de **dominance au sens Pareto**. L'ensemble des meilleurs compromis est appelé le **front Pareto**, la surface de compromis ou l'ensemble des **solutions efficaces**. Cet ensemble de solutions constitue un équilibre, dans le sens qu'aucune amélioration ne peut être faite sur un objectif sans dégradation d'au moins un autre objectif. La **solution Pareto** consiste à obtenir le**front de Pareto** PO ou d'approximer la **frontière de Pareto** PO^* [24]. Pour bien comprendre les notion d'optimalité dans le cas multiobjectif on introduit les définitions suivantes, basées sur les travaux de Pareto [10] :

Definition 1. Étant donné $x, y \in \mathbb{R}^k$ on dit que :

$$\begin{cases} x \leq y & Si\ x_i \leq y_i,\ i = 1...k \\ x < y & Si\ x \leq y\ et\ x \neq y \end{cases}$$

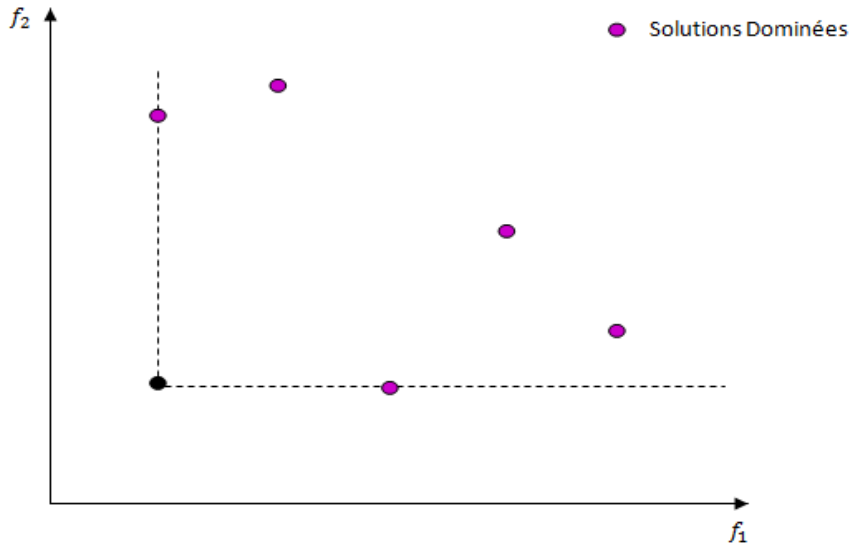

FIGURE 1.4 – Relation de dominance

Definition 2. On dit qu'un vecteur solution $x \in X \subseteq \mathbb{R}^D$ est **non-dominé** dans X s'il n'existe pas d'autre vecteur $x^{'}$ tel que $F(x^{'}) < F(x)$.

Definition 3. On dit qu'un vecteur solution $y^* \in F \subset \mathbb{R}^D$ est **Pareto-optimal** s'il est non-dominé dans F.

Definition 4. L'ensemble optimal de Pareto est défini par :

$$P* = \{x \in \mathcal{X}, x\ est\ pareto - optimal\} :$$

Definition 5. Le front optimal de Pareto est défini par :

$$F(p^*) = \{F(x) \in \mathbb{R}^k \mid x \in P^*\}$$

La figure suivante represente le front de pareto dans le cas biobjectif [10] :

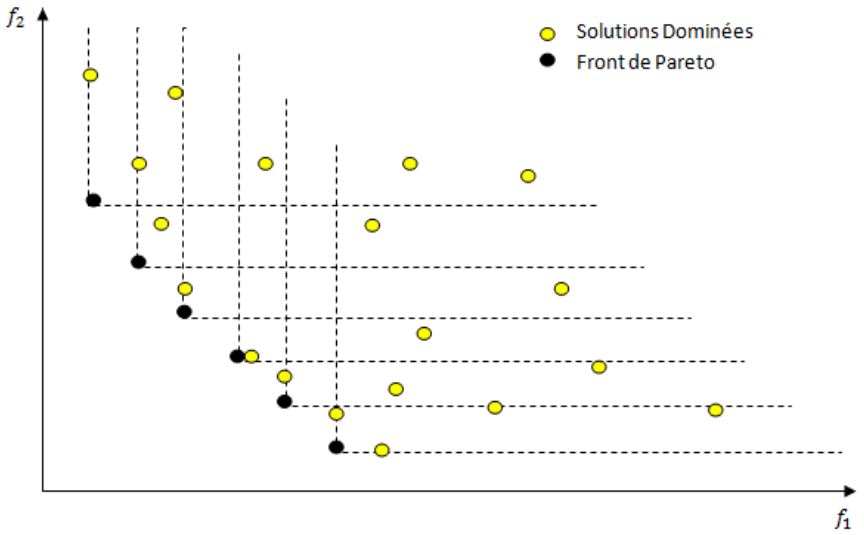

FIGURE 1.5 – Front de Pareto dans le cas bi-objectif

L'espace de recherche qui correspondent au front de Pareto $F(P^*)$ est le but des algorithmes d'optimisation multiobjectifs.

Definition 6. Le point idéal défini par $F^* = (f_1^*, f_2^*, ...f_k^*)$ avec $f_i^* = \min_{x \in X} f_i(x)$, $i = 1...k$.

De manière évidente, il s'agit bien ici d'une borne inférieure puisque toute solution est nécessairement dominée (au pire faiblement pour les points extrêmes) par ce point.[24]

Une visualisation de l'ensemble de ces définitions est donnée sur la figure suivante [24] :

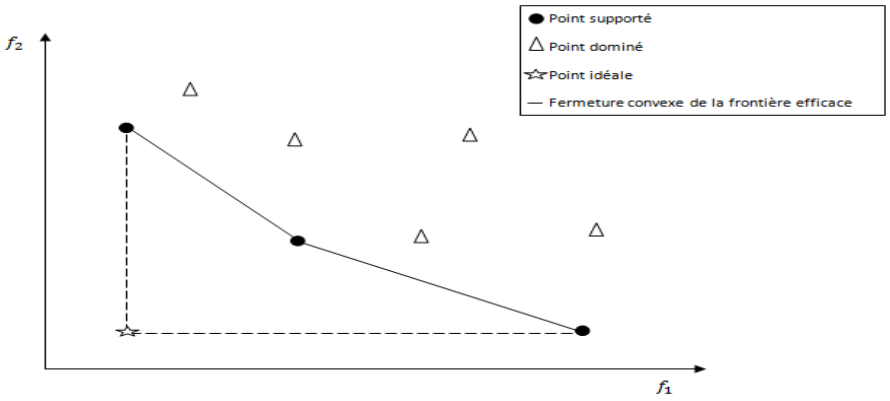

FIGURE 1.6 – Points caractéristiques d'un problème de minimisation biobjectif .

1.3.2 Classification des méthodes d'optimisation multi-objectif.

La classification des méthodes d'optimisation multiobjectifs est donnée comme suit [39] :

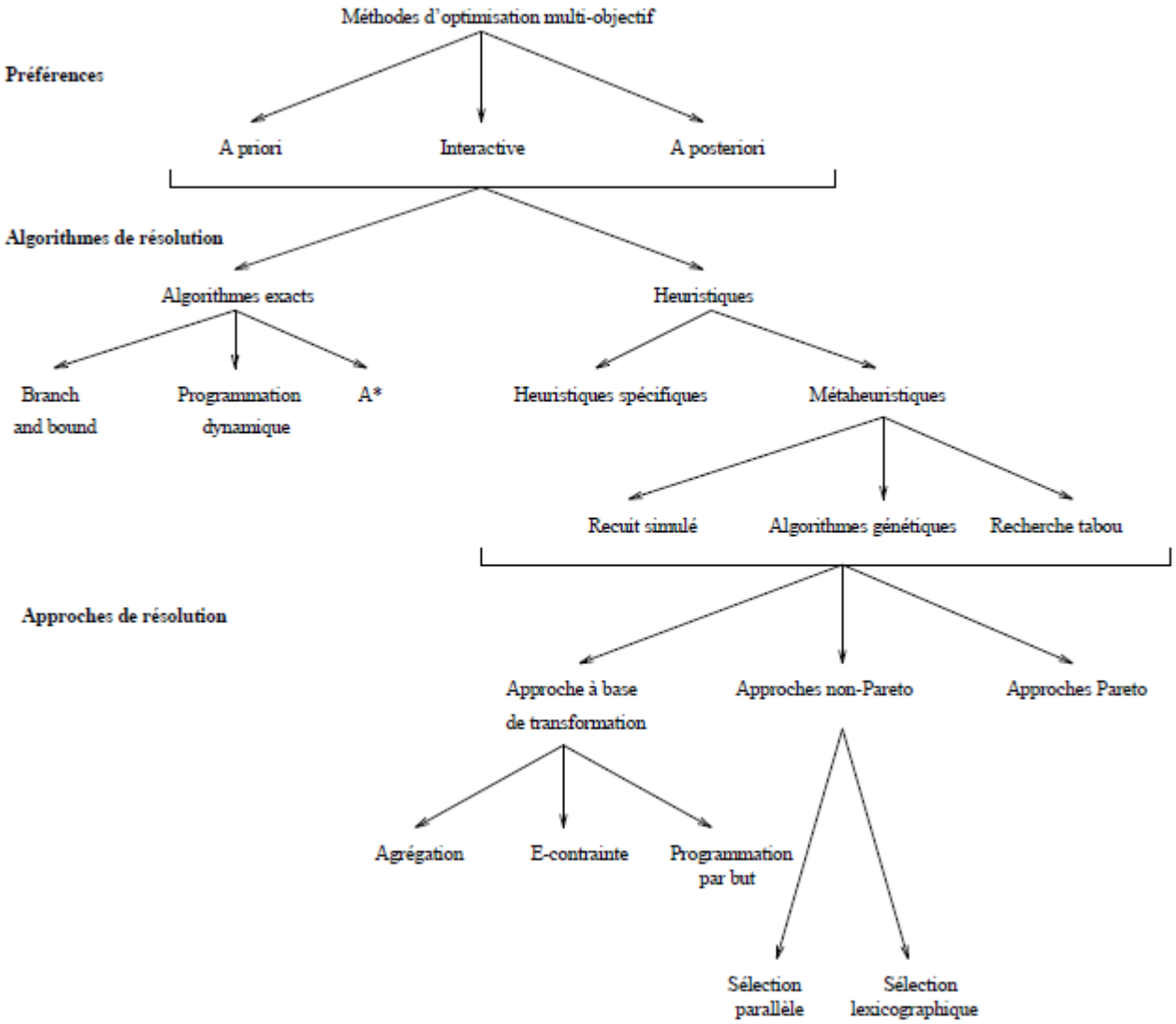

FIGURE 1.7 – Classification des méthodes d'optimisation multi-objectif

1.3.3 Les approches de résolution des problèmes multiobjectif

La résolution d'un problème multiobjectif peut être divisé en deux phases [24] :

1. La recherche **des solutions de meilleur compromis** : C'est la phase d'optimisation multiobjectif.
2. Le **choix de la solution** à retenir : C'est la tâche du **décideur** qui parmi l'ensemble des solutions de compromis doit extraire celle(s) qu'il utilisera. On parle alors ici de décision multiobjectif et cela fait appel à la théorie de la décision.

Dans les différentes publications, nous rencontrons deux classifications différentes des approches de résolution de problèmes multiobjectifs :

► Le premier classement adopte un point de vue décideur, les approches sont classées en fonction de l'usage que l'on désire en faire.

► Le deuxième classement adopte un point de vue concepteur, les approches sont triées de leur façon de traiter les fonctions objectifs.

Point de vue du décideur[24]

On distingue à cet égard trois schémas possibles. Soit le décideur intervient dés le début de la définition du problème, en exprimant ses préférences, afin de transformer un problème multiobjectif en un problème mono objectif. Soit le décideur effectue son choix dans l'ensemble des solutions proposées par le solveur multiobjectif :

a) Les approches à priori : dans ce cas le décideur est supposé connaître à priori le poids de chaque objectif afin de les mélanger dans une fonction unique. Cela revient à résoudre un problème mono-objectif. le décideur ne peut pas exprimer clairement sa fonction d'utilité, parce que les différents objectifs sont non commensurables dans la plupart des cas.

b) Les approches interactives : cette approche permet de bien prendre en compte les préférences du décideur . Ce dernier intervient de manière à modifier certaines variables ou contraintes afin de diriger le processus d'optimisation et modifie ainsi interactivement le compromis entre ses préférences et les résultats.

c) Les approches à posteriori : permet de fournir au décideur un ensemble de bonnes solutions bien réparties. Ensuite, sélectionner celle qui lui semble la plus appropriée. Ainsi, il n'est plus nécessaire de modéliser les préférences du décideur.

Point de vue concepteur[24]

Les approches utilisées pour la résolution de problèmes multiobjctifs peuvent être classées en deux catégories : les approches non Pareto et les approches Pareto.
La figure suivante représente les approches de résolution de problèmes multiobjectifs [24] :

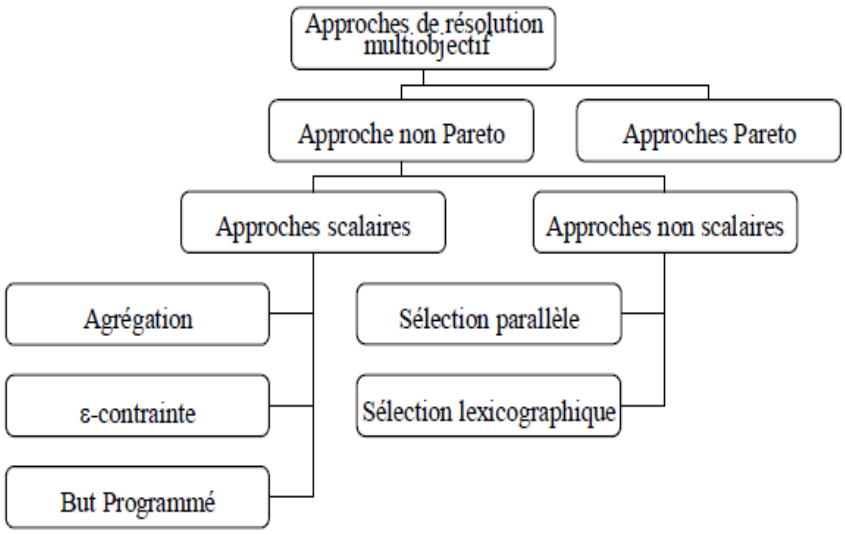

FIGURE 1.8 – Les approches de résolution de problèmes multiobjctifs

1.3.3.1 Les approches non Pareto

Ses approches sont classées en deux catégories : les approches scalaires qui transforment le problème multiobjectif en problème mono-objectif et les approches non scalaires qui gardent l'approche multiobjectif, mais en traitant séparément chacun des objectifs[24].

1.3.3.1.1 Les approches scalaires

Plusieurs approches différentes ont été mises au point pour transformer les problèmes multiobjectifs en problèmes mono-objectifs : les approches agrégées, programmation par but, et les approches ξ-contraintes.

A. Approche d'agrégation

C'est l'une des premières approches utilisée pour résoudre les problèmes multiobjectifs. Elle consiste à le transformer à un problème monoobjectif en définissant une fonction objectif unique F comme étant la somme pondérée des différentes fonctions objectifs du problème initial. En affectant à chaque objectif un coefficient de poids qui représente l'importance relative que le décideur attribue à l'objectif[24] :

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

Ou les poids $\lambda_i \in [0, 1]$ avec $\sum_{i=1}^{k} \lambda_i = 1$

Cette approche a l'avantage de pouvoir réutiliser tous les algorithmes classiques dédiés aux problèmes d'optimisation à un seul objectif. Elle a aussi deux inconvénients importants. Le premier est dû au fait que pour avoir un ensemble de points bien repartis sur le front Pareto, les différentes valeurs λ_i doivent être choisies judicieusement. Il est donc nécessaire d'avoir une bonne connaissance du problème. Le deuxième inconvénient provient du fait que cette méthode ne permet pas de calculer intégralement la surface de compromis lorsque celle-ci n'est pas convexe.

B. But programmé

Le décideur doit définir des buts T_i ou références qu'il désire atteindre pour chaque objectif f_i . Ces valeurs sont introduites dans la formulation du problème en le transformant en un problème mono-objectif. La nouvelle fonction objectif est modifiée de façon à minimiser les écarts entre les résultats et les buts à atteindre :

$$\min(|\Sigma_{i=1}^{k} f_i(s) - T_i|)$$

avec $s \in \Omega$.

Critique : nous pouvons reprendre la critique faite pour la somme pondérée. La méthode est facile à mettre en œuvre mais la définition des poids et des objectifs à atteindre est une question délicate qui détermine l'efficacité de la méthode.

C. Approches $\varepsilon-$contraintes

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

Critique : la connaissance a priori des intervalles appropriés pour les valeurs de ϵ_j est exigée pour tous les objectifs.

1.3.3.1.2 Les approches non scalaires

Ces approches ne transforment pas le problème multiobjectif en un problème mono-objectif, mais utilisent des opérateurs qui traitent séparément les différents objectifs, elles n'utilisent pas non plus la notion de dominance Pareto il s'agit notament du : la sélection parallèle, la sélection lexicographique.

A. Sélection parallèle

C'est la première méthode ou un algorithme génétique a été proposé pour la résolution de problèmes multiobjectifs.
Cette algorithme est appelé VEGA (Vector Evaluated Genetic Algorithm) permet de sélectionner les individus selon chaque objectif de manière indépendante.
L'idée est simple : pour k objectifs et une population de n individus une sélection de n/k meilleurs individus est effectuée pour chaque objectif. Ainsi k sous-populations vont être créées et ensuite mélangées afin d'obtenir une nouvelle population de taille n. le processus se termine par l'application des opérateurs génétiques (croisement et mutation). (Nous allons détaillé cet algo dans la section 1.5).

B. Sélection lexicographique

Fourman a proposé une méthode dans laquelle les objectifs sont préalablement rangés par ordre d'importance par le décideur. Ensuite, l'optimum est obtenu en minimisant tout d'abord la fonction objectif la plus importante puis la deuxième et ainsi de suite. Soient les fonctions objectifs f_i avec $i = 1, \ldots, k$, supposons un ordre tel que : $f_1 > f_2 > ... > f_k$ il faut :

$$\begin{cases} Minimiser \quad f_1(x), \\ avec \quad g_j(x) \quad satisfait : \quad j = 1, ..., m \end{cases}$$

Soit x_1^* la meilleure solution trouvée avec : $f_1^* = f_1(x_1^*) f_1$ devient alors une nouvelle contrainte. L'expression du nouveau problème est donc :

$$\begin{cases} Minimiser \quad f_2(x), \\ avec \quad g_j(x) \quad satisfait : \quad j = 1, ..., m \quad et \quad f_1(x) = f_1^* \end{cases}$$

Soit x_2^* la solution de ce problème. Le $i^{ème}$ problème sera le suivant :

$$\begin{cases} Minimiser \quad f_i(x), \\ avec \quad g_j(x) \quad satisfait : \quad j = 1, ..., m \quad et \quad f_1(x) = f_1^*, f_2(x) = f_2^*, ..., f_{i-1}(x) = f_{i-1}^* \end{cases}$$

La procédure est répétée jusqu'à ce que tous les objectifs soient traités. La solution obtenue à l'étape k sera la solution du problème. L'inconvénient essentiel de cette méthode est la grande importance attribuée aux objectifs classés en premier. La meilleure solution f_1^* trouvée pour l'objectif le plus important va faire converger l'algorithme vers une zone restreinte de l'espace d'état et enfermer les points dans une niche (ensemble d'individus situés dans un espace restreint).

1.3.3.2 Les approches Pareto

Ces approches sont de type a posteriori. Au 19éme Siècle, Vilfredo Pareto un mathématicien Italien formule le concept suivant : dans un problème multiobjectif il existe un équilibre tel que l'on ne peut pas améliorer un objectif sans détériorer au moins un des autres objectifs.
Les approches Pareto utilisent directement la notion de dominance dans la sélection des solutions générées. [24]

1.4 Méthodes de resolution des problèmes d'optimisation multi-objectifs

Tout comme pour l'optimisation combinatoire , deux classes de méthodes de résolution pour traiter les problèmes multiobjectifs : les méthodes exactes dédiées à résoudre optimalement

les petites instances et les méthodes approchées : les heuristiques et en particulier les métaheuristiques (génériques) permettant d'approximer les meilleures solutions sur les plus grandes instances. Enfin, le choix de la méthode de résolution à mettre en œuvre dépendra souvent de la complexité du problème. En effet, suivant sa complexité, le problème pourra ou non être résolu de façon optimale. Dans le cas de problèmes classés dans la classe P, un algorithme polynomial a été mis en évidence. Il suffit donc de l'utiliser. Dans le cas de problèmes NP-difficiles, deux possibilités sont offertes. Si le problème est de petite taille, alors un algorithme exact permettant de trouver la solution optimale peut être utilisé (Branch and Bound, programmation dynamique...). Malheureusement, ces algorithmes par nature énumératifs, souffrent de l'explosion combinatoire et ne peuvent s'appliquer à des problèmes de grandes tailles (même si en pratique la taille n'est pas le seul critère limitant). Dans ce cas, il est nécessaire de faire appel à des heuristiques permettant de trouver de bonnes solutions approchées. Parmi ces heuristiques, on trouve les métaheuristiques qui fournissent des schémas de résolution généraux permettant de les appliquer potentiellement à tous les problèmes.

1.4.1 Méthodes exactes

Dans la littérature, plusieurs méthodes exactes multiobjectifs baséessur le branch and bound, sur l'algorithme A^* et la programmation dynamique , la méthode à deux phases ont été proposées pour résoudre de petits problèmes à deux objectifs. Pour les problèmes à plus de deux critères ou de grandes tailles, il n'existe pas de procédures exactes efficaces, étant données les difficultés simultanées de la complexité NP-difficile, et le cadre multiobjectif des problèmes. Ainsi, on assiste ces dernières années à un accroissement de intérêt porté sur l'utilisation de métaheuristiques pour la résolution de Problèmes Multiobjectifs, et spécialement de problèmes réels.

1.4.1.1 Algorithme A^*

Stewart et White proposèrent une version multiobjectif de l'algorithme A^* (A^*MO).
L'algorithme A^* est maintenu dans sa forme originale. Le poids d'un arc (u, v) dans le graphe de recherche, correspond à un tuple $(c_1, c_2, ..., c_n)$ où c_i désigne le coût relatif à l'objectif f_i induit par le passage de u à v. le coût d'un chemin, correspond à la somme vectorielle des poids des arcs qui le composent. Les fonctions k^*, g^*, h^* prennent une forme non scalaire où $k^*(u, v)$ désigne l'ensemble des coûts non dominés des chemins reliant u à v, $g^*(u)$ représente le coût de la recherche ayant aboutit à la solution intermédiaire u, $h^*(u)$ désigne l'ensemble des vecteurs coût non dominés parmi l'ensemble $k^*(u, v)/v \in T$ où T représente l'ensemble des états terminaux. Les nœuds du graphe sont triés et parcourus comme pour l'algorithme A* classique. L'ordre de tri est décrit par les relations de dominance entre les vecteurs coût f [6].

1.4.1.2 Programmation dynamique

La technique de programmation dynamique multiobjectif a été implémentée par Carraway , Morin et Moskowitz pour la résolution du problème de routage dans les réseaux.
L'application de la programmation dynamique pour l'optimisation multiobjectif est rare car ceci est difficile quand le nombre d'objectifs à optimiser est élevé (> 2). La difficulté est liée au principe de monotonie exigée par la méthode. Ce principe réclame un grand volume d'espace de stockage qu'il faudra utiliser pour sauvegarder l'ensemble des résultats des étapes antérieures, en plus d'un temps de calcul très élevé [30].

1.4.2 Méthodes approchées

1.4.2.1 Recuit simulé multiobjectif :

L'utilisation du recuit simulé dans le cadre de l'optimisation multiobjectif a premièrement été étudiée par Serafini. L'idée principale dans l'application du recuit simulé pour l'optimisation multiobjectif consiste en l'utilisation d'une norme pondérée des composants du vecteur objectif, afin d'accepter ou rejeter une solution de coût inférieur [40].

1.4.2.2 Recherche Tabou multiobjectif

Les premiers travaux portants sur l'utilisation de la Recherche Tabou pour l'optimisation multiobjectif opéraient par transformation vers le mono-objectif. Des approches ultérieures, étendent les principes de la Recherche Tabou afin de produire une bonne approximation de la frontière Pareto.
Dans d'autre travaux on se base sur une approche lexicographique. la recherche tabou est utilisée pour résoudre le problème de formation de cellules de fabrication dans les ateliers. La méthode est appliquée pour optimiser une séquence de sous-problèmes mono objectif sous contraintes. Les objectifs sont traités de façon séquentielle suivant l'ordre d'importances des critères. Le premier sous-problème résolu consiste à optimiser la fonction objectif la plus prioritaire. Le deuxième problème revient alors à optimiser la deuxième fonction objectif la plus importante. Ceci en maintenant satisfaite une contrainte supplémentaire qui consiste à ne pas détériorer, avec une certaine marge, la valeur obtenue pour la première fonction. Les autres critères sont ainsi traités en réitérant ce procédé [40].

1.5 Les algorithmes génétiques

1.5.1 Algorithmes génétiques pour les problèmes mono-objectifs

Un algorithme génétique recherche le ou les extrema d'une fonction définie sur un espace de données. Pour l'utiliser, on doit disposer des cinq éléments suivants [18] :

1. Un principe de codage de l'élément de population.
2. Un mécanisme de génération de la population initiale.
3. Une fonction à optimiser appelée fitness ou fonction de l'évaluation des individus.
4. Des opérateurs permettant de diversifier la population au cours des générations et d'explorer l'espace d'état. L'opérateur de croisement recompose les gènes d'individus existant dans la population, l'opérateur de mutation a pour but de garantir l'exploration de l'espace d'état.
5. Des paramètres de dimensionnement : taille de la population, nombre total de générations ou critère d'arrêt, probabilités d'application des opérateurs de croisement et de mutation.

La figure suivante presente le principe de fonctionnement des algorithmes génetiques [18] :

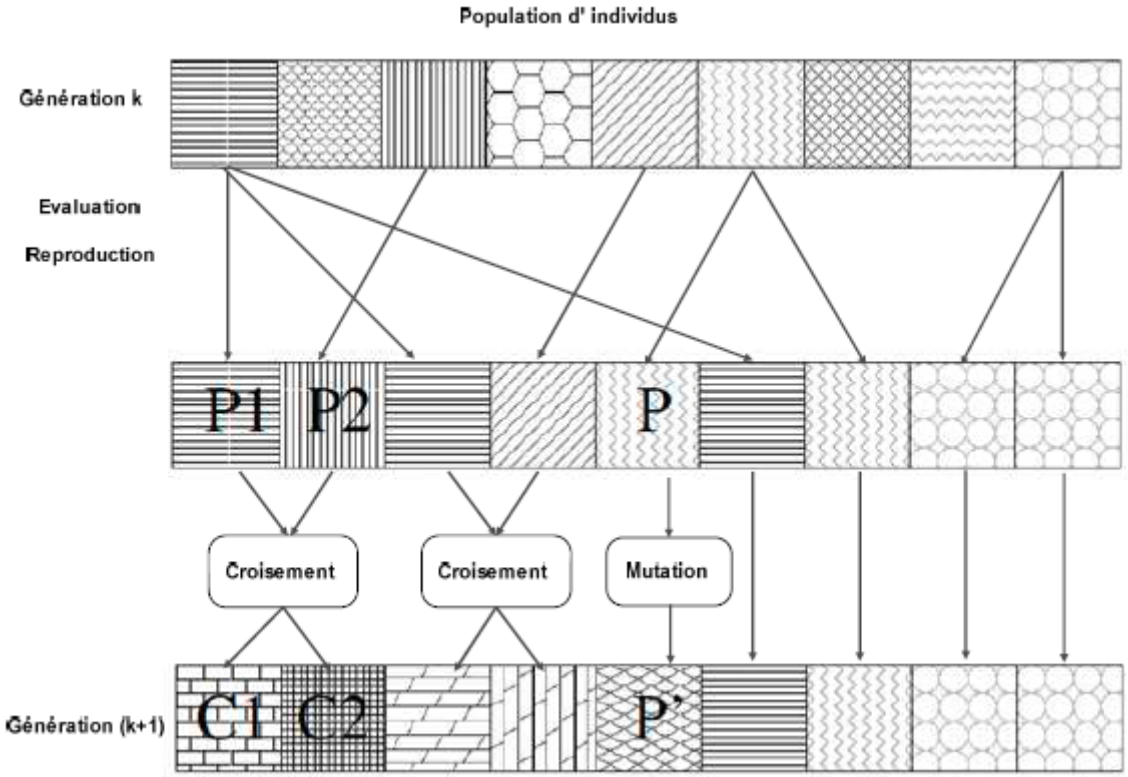

FIGURE 1.9 – Principe des algorithmes génétiques

Afin de trouver la solution optimale d'un problème en utilisant l'algorithme génétique, on commence par générer un ensemble de solutions (population d'individus) de façon aléatoire. L'évolution d'une génération à la suivante, utilise-les trois opérations (Sélection, Croisement et Mutation) qui sont appliqués à tous les éléments de populations. Des couples de parents P_1 et P_2 sont sélectionnés en fonction de leurs adaptations. L'operateur de croisement est applique avec une probabilité P_c et génère des couples d'enfants E_1 et E_2. Ensuite, l'operateur de mutation est applique aux enfants avec la probabilité P_m et génère des individus mutes $E_1^{'}$ et $E_2^{'}$ qui vont être insérés dans la nouvelle population. Parmi les critères d'arrêt qui peuvent être choisis pour l'algorithme, on peut citer :

• L'obtention après un certain nombre de générations, d'un degré d'uniformité des individus de la population supérieur a un seuil souhaite.

• L'atteinte d'un nombre maximal de générations fixé à priori.

La mise en œuvre d'un algorithme génétique : est réalisée suivant les étapes suivantes [18] :

1. Création d'une population initiale.
2. Evaluation des individus de la population.
3. Sélection des meilleurs individus.
4. Reproduction (Croissement et mutation).
5. Formation d'une nouvelle génération.

✓ **L'organigramme de fonctionnement d'un algorithme génétique :**

Début
1 Initialisation de la population.
2 Evaluation de la population.
3 Sélection(génération intermédiaire).
4 Reproduction(croisement et mutation,nouvelle génération courante).
5 Evaluation des individus.
6 **Si** le ritère d'arret et vérifié.
6.1 Le résultat est un meilleurs individus.
7 **Sinon**.
7.1 Revenir a 3.
Fin Si
Fin

Vocabulaires des algorithmes génétiques : dans cette section nous introduisons quelques vocabulaires utilisés dans la mise en œuvre des algorithmes génétiques[18].

- Individu : représenté par un chromosome (génome).
- Un chromosome est une chaîne de gènes.
- Génotype : l'ensemble des gènes représentés par un chromosome.
- Phénotype : l'ensemble des valeurs observables prises par chaque gène.
- Fonction d'adaptation : fitness.
- Opération de reproduction :
  - Le croissement.
  - La mutation.
- Génération : l'ensemble de la population à un moment donné du processus.

1.5.1.1 Le codage

Le codage est une modélisation d'une solution d'un problème donné sous forme d'une séquence de caractères appelée chromosome où chaque caractère, dit aussi gène, représente une variable ou une partie du problème. La tâche principale consiste à choisir le contenu des gènes qui facilite la description du problème et respecte ses contraintes[18].
Par exemple pour le problème du voyageur de commerce on peut préférer utiliser l'alphabetal-lélique c_1, c_2, c_3, ..., c_n où c_i représente la ville de numéro i [15].

1.5.1.1.1 Le codage binaire

Le codage binaire consiste à utiliser des bits (0 où 1) pour représenter les différentes solutions. Dans le sens où les gènes sont représentés par des bits et les chromosomes sont représentés par des chaines de bits. Le type de ce codage s'adapte bien aux problèmes de type binaire, comme le problème max SAT ou les problèmes du sac à dos. Prenons l'exemple du problème du sac à dos dans sa forme la plus simple (unidimensionnelle). Supposant que nous devons sélectionner un sous ensemble d'objets parmi un ensemble de 10 objets. Les solutions peuvent être facilement représentées par des chaines binaires de taille 10, où chaque bit représente l'état d'un objet : le bit 1 pour dire que l'objet est sélectionné, le bit 0 pour dire que l'objet est écarté. La figure suivante représente une des solutions du problème tel que les objets 2, 3, 5, 8, 9, et 10 sont sélectionnés. Tandis que les autres (i.e. 1, 4, 6 et 7) sont écartés [15].

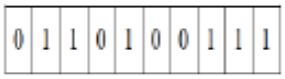

FIGURE 1.10 – Codage binaire d'un chromosome.

1.5.1.1.2 Le codage réel

Ce type de codage est beaucoup plus efficace pour représenter des problèmes de type continu. Il représente les solutions par des suites de type réel [15].
Comme le montre la figure suivante :

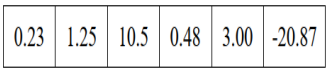

FIGURE 1.11 – Codage réel d'un chromosome.

On peut distinguer un autre type spécial du codage réel, c'est le codage entier ou discret. Il utilise des entiers au lieu de réels. Ce type est beaucoup plus efficace pour la représentation de certain type de problèmes discrets, comme les problèmes d'ordonnancement et le problème du voyageur de commerce. Prenons l'exemple du problème du voyageur de commerce où un voyageur de commerce doit parcourir un ensemble de 10 villes en choisissant le plus court chemin permettant de passer par chacune une seule fois. La figure suivante représente une solution du problème où chaque gène représente une ville et l'ensemble des gènes représente l'ordre de parcours de l'ensemble des 10 villes.

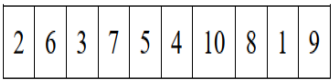

FIGURE 1.12 – Codage entier d'un chromosome.

1.5.1.1.3 Le codage à caractère

Il s'agit d'utiliser une suite de caractères différents pour représenter le chromosome, comme le montre la figure suivante :

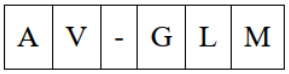

FIGURE 1.13 – Codage à caractère d'un chromosome.

1.5.1.1.4 Le codage arborescent

Ce type de codage utilise une représentation arborescente des individus avec un ensemble de nœuds, où l'ensemble des nœuds représente un individu [15].
Comme le montre la figure suivante :

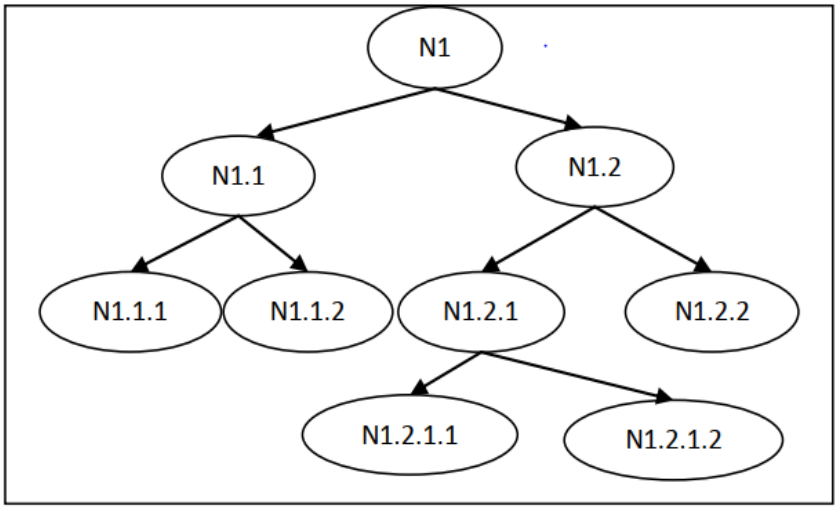

FIGURE 1.14 – Codage arborescent d'un chromosome.

1.5.1.1.5 La population initiale

Le choix de la population initiale d'individus conditionne fortement la rapidité de l'algorithme. Pour générer la population initiale, nous avons, deux possibilités. Dans le cas où aucune information sur la position de la solution n'est disponible, le but est de recouvrir au mieux l'espace d'état. Une génération aléatoire est donc engendrée par des tirages uniformes sur chaque gène du chromosome. Dans le cas où l'utilisateur connaîtrait un sous-domaine de l'espace d'état où la solution se trouve, la population initiale est générée dans ce sous-espace.
Dans le cas d'une génération engendrée de façon aléatoire, il existe deux éventualités. Soit il est possible de savoir à l'avance si un point respecte les contraintes du problème d'optimisation, la population est alors générée aléatoirement dans le domaine admissible. Soit il n'est pas possible de le savoir, le respect des contraintes sera alors assuré via l'ajout d'une pénalisation dans la fonction objectif et la population est uniformément distribuée sur tout l'espace d'état [18].

1.5.1.2 Fonction d'évaluation (fitness)

L'évaluation de l'adaptation de chaque individu à l'environnement est réalisée au moyen d'une fonction d'adaptation (fitness). Cette fonction attribue à chaque individu une valeur qui représente son niveau d'adaptation. La fonction d'adaptation peut affecter directement la qualité des résultats obtenus par l'AG ainsi que le temps d'exécution. Elle est fonction du critère à minimiser. En raison de l'analogie avec la théorie de l'évolution (survie des individus les mieux adaptés à leur environnement), l'algorithme génétique est naturellement formulé en terme de maximisation. Etant donné une fonction h réelle à une ou plusieurs variables [18], le problème d'optimisation sur l'espace de recherche S s'écrit de la manière suivante :

$$\max_{x \in S} h(x) \tag{1.2}$$

Dans beaucoup de problèmes, l'objectif est exprimé sous forme de minimisation d'une fonction coût f :

$$\min_{x \in S} f(x) \tag{1.3}$$

Le passage du problème de minimisation à un problème de maximisation est obtenu par transformation de la fonction f selon la relation suivante :

$$h(x) = \frac{1}{1 + f(x)} \tag{1.4}$$

1.5.1.3 Les opérateurs de base d'un AG

Les opérateurs sont fondamentaux pour implanter le processus reproductif caractéristique d'un AG. De nombreux opérateurs génétiques existent ainsi que différentes stratégies d'implantation.

1.5.1.3.1 La sélection

L'opérateur de sélection est chargé de " favoriser" les meilleurs individus. Plus formellement, l'opérateur de sélection va générer à partir de la population courante une nouvelle population par copie des individus choisis de la population courante. La copie des chaînes s'effectue en fonction des valeurs de la fonction d'adaptation. Ce procédé permet de donner aux meilleures chaînes, une probabilité élevée de contribuer à la génération suivante. Cet opérateur est bien entendu une version artificielle de la sélection naturelle, la survie darwinienne des chaînes les plus adaptées. Plusieurs stratégies sont possibles pour effectuer une telle sélection parmi lesquelles nous abordons [18] :

A. La sélection par classement

Elle consiste à ranger les individus de la population dans un ordre croissant (ou décroissant selon l'objectif) et à retenir un nombre fixé de génotypes. Ainsi, seuls les individus les plus forts sont conservés. L'inconvénient majeur de cette méthode est la convergence prématurée de l'algorithme génétique [18].

B. La sélection par roulette

Selon cette méthode, chaque chromosome est copié dans la nouvelle population proportionnellement à sa fitness. On effectue en quelque sorte, autant de tirages avec remise que d'éléments existant dans la population [17].
Ainsi pour un chromosome particulier ch_i de fitness $f(ch_i)$, la probabilité de sa sélection dans la nouvelle population de taille N est :

$$P(ch_i) = \frac{f(ch_i)}{\sum_{j=1}^{N} f(ch_j)}$$

C. La sélection par tournois

Cette méthode est celle avec laquelle on obtient les résultats les plus satisfaisants. Le principe de cette méthode est le suivant : on effectue un tirage avec remise de deux individus de P (P : l'ensemble de la population), et on les fait "combattre". Celui qui a la fitness la plus élevée l'emporte avec une probabilité p (p :probabilité) comprise entre 0.5 et 1. On répète ce processus n fois de manière à obtenir les n individus de la population P' qui serviront de parents [3].
La variance de cette méthode est élevée et le fait d'augmenter ou de diminuer la valeur de la p permet respectivement de diminuer ou d'augmenter la pression de la sélection.

1.5.1.4 Le croisement

La naissance d'un nouvel individu, nécessite la prise aléatoire d'une partie des gènes de chacun des deux parents. Ce phénomène, issu de la nature est appelé croisement (crossover). Il s'agit d'un processus essentiel pour explorer l'espace des solutions possibles. Une fois la sélection terminée, les individus sont aléatoirement répartis en couples. Les chromosomes parents sont alors copiés et recombinés afin de produire chacun deux descendants ayant des caractéristiques issues des deux parents. Dans le but de garder quelques individus parents dans la prochaine population, on associe à l'algorithme génétique une probabilité de croisement, qui permet de décider si les parents seront croisés entre eux ou s'ils seront tout simplement recopiés dans la population suivante. Il existe plusieurs types de croisement parmi lesquels on trouve : le

croisement en 1 point, le croisement en deux points et le croissement en N points [3][17], ces types sont résumés dans la figure suivante :

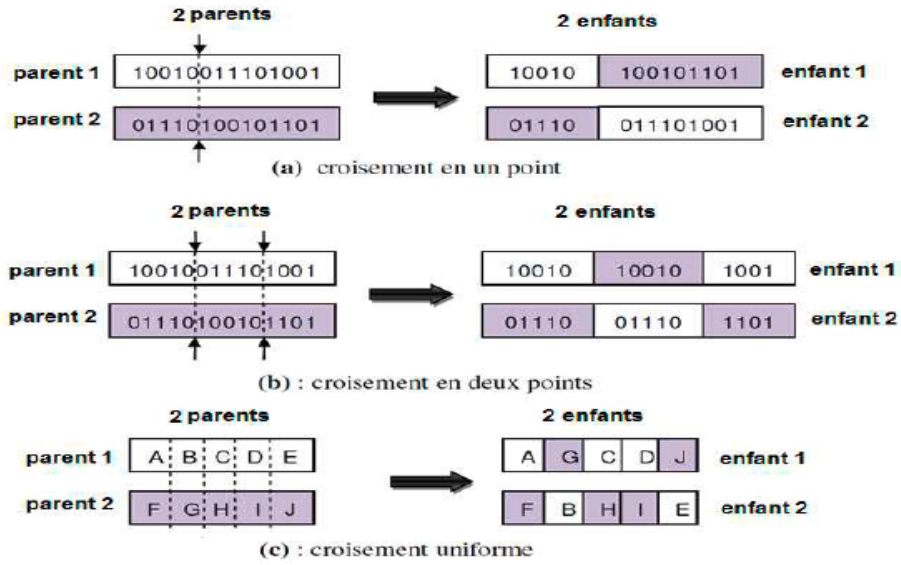

FIGURE 1.15 – Exemples d'opération de croisement

1.5.1.5 La mutation

Le rôle de cet opérateur est de modifier aléatoirement la valeur d'un gène dans un chromosome. Dans le cas du codage binaire, chaque bit $a_i \in \{0, 1\}$ est remplacé par son complémentaire $a_i = 1 - a_i$. Dans le cas d'un codage réel, on utilise principalement deux opérateurs de mutation : la mutation uniforme et la mutation non uniforme. En supposant fixée la probabilité de mutation p_m, un tirage au sort pour chaque gène x_k d'un chromosome ch permet de décider si ce gène doit être modifié ou non. Nous supposons que le gène prend ses valeurs dans un intervalle $[x_k^{\min}, x_k^{\max}]$.
La mutation uniforme est une simple extension de la mutation binaire, on remplace le gène x_k sélectionné par une valeur quelconque x'_k tirée aléatoirement dans l'intervalle $[x_k^{\min}, x_k^{\max}]$ [17].
Pour la mutation non uniforme, le calcul de la nouvelle valeur d'un gène est un peu plus complexe. Le gène x_k subit des modifications importantes durant les premières générations, puis graduellement décroissantes au fur et à mesure que l'on progresse dans le processus d'optimisation. Pour une génération t, on tire au sort une valeur binaire qui décidera si le changement doit être positif ou négatif. La nouvelle valeur x'_k du gène x_k est donnée par :

$$\begin{cases} x_k + \Delta(t, x_k^{\max} - x_k) & si \quad randam = 0 \\ \forall x_k - \Delta(t, x_k - x_k^{\min}) & si \quad randam = 1 \end{cases}$$

où $\Delta(t, y)$ est une fonction qui définit l'écart entre la nouvelle valeur et la valeur initiale à la génération t et rand est un nombre aléatoire qui prend les valeurs 0 ou 1. La figure suivante représente le schéma de mutation.

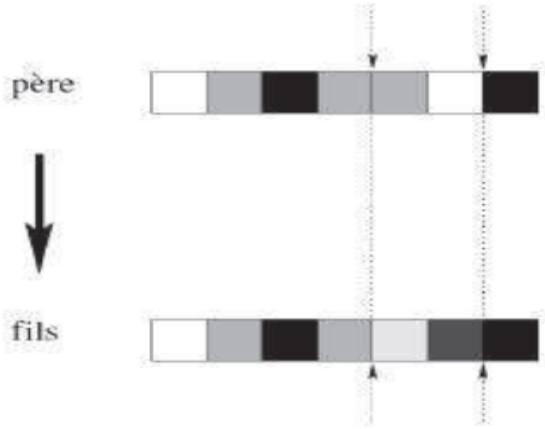

FIGURE 1.16 – Schéma de mutation.

1.5.1.6 La convergence d'un AG

Le cycle de génération et de sélection de population est répété jusqu'à ce qu'un critère d'arrêt soit satisfait ; ce critère peut être notamment un nombre maximum de générations, un temps maximal de calcul, une valeur de fitness minimale, ou/et une convergence vers une solution satisfaisante. Les valeurs de tels paramètres dépendent fortement de la problématique étudiée. Ainsi il n'existe pas de paramètres qui soient adaptés à la résolution de tous les problèmes qui peuvent être posés à un algorithme génétique. Cependant, certaines valeurs sont souvent utilisées (définies dans la littérature) et peuvent être de bons points de départ pour démarrer une recherche de solutions à l'aide d'un AG [18].

- La probabilité de croisement est choisie dans l'intervalle [0.7, 0.99].
- La probabilité de mutation est choisie dans l'intervalle [0.001, 0.01].

1.5.2 Algorithmes Génétiques pour les problèmes multi-objectifs

1.5.2.1 Problèmes de conception et composants de l'AG multiobjectifs

Etant donné l'existence d'un grand nombre d'extension de l'algorithme multiobjectifs nous allons nous intéresser dans ce qui suit aux extensions qui ferons l'objet de la résolution de notre problème d'étude .

1.5.2.1.1 Approches de la somme pondérée

L'approche classique pour résoudre un problème d'optimisation multiobjectif est d'attribuer un poids w_i à chaque fonction objectif normalisée $f'_i(x)$ de sorte que le problème soit converti en un problème a objectif unique avec une fonction objectif scalaire comme suit :

$$\min z = w_1 f'_1(x) + w_2 f'_2(x) + \ldots\ldots + w_k f'_k(x) \tag{1.5}$$

où $f'_i(x)$ est la fonction objectif normalisée $f_i(x)$ et $\sum w_i = 1$ (cette approche s'appelle l'approche a priori étant donné que l'utilisateur est censé fournir les poids [2]).
Résoudre un problème avec la fonction objectif (1.5) pour le vecteur de poids $w = \{w_1, w_2, w_3 \ldots w_k\}$ donne une seule solution, et si plusieurs solutions sont souhaitées, le problème doit être résolu plusieurs fois avec différents combinaisons de poids.
La principale difficulté avec cette approche consiste à sélectionner un vecteur de poids pour

chaque série. À automatiser ce processus ; Hajela et Lin ont proposé la WBGA pour l'optimisation multiobjectifs (WBGA-MO).
Dans la WBGA-MO, chaque solution x_i de la population utilise un vecteur de poids différent $w_i = \{w_1, w_2, w_3 \dots w_k\}$ dans le calcul de la fonction objectif additionnée 1.5. le vecteur de poids w_i est intégré dans le chromosome de solution x_i Par conséquent, plusieurs solutions peuvent être simultanées trouvé en une seule fois. De plus, le poids les vecteurs peuvent être ajustés pour promouvoir la diversité des population.
D'autres chercheurs ont proposé un modèle basé sur la MOGA. Sur une somme pondérée de fonctions objectives multiples où Le vecteur de poids normalisé w_i est généré aléatoirement pour chaque solution x_i au cours de la phase de sélection à chaque génération.
Cette approche vise à préciser plusieurs directions de recherche en une seule fois sans utiliser de paramètres supplémentaires. Le procédure générale du RWGA en utilisant des poids aléatoires est donné comme suit :
Procédure RWGA :

- E = archive externe pour stocker des solutions non dominées trouvé lors de la recherche jusqu'à présent.

- n_E = nombre de solutions élitistes immigrant de E à P à chaque génération.

✓ **L'algorithme associé :**

Début

1 Générez une population aléatoire.

2 Attribuez une valeur de remise en forme à chaque solution $x \in P_t$ en effectuant les étapes suivantes :

2.1 Générez un nombre aléatoire u_k dans $[0, 1]$ pour chaque objectif k ,k=1...K.

2.2 Calculez le poids aléatoire de chaque objectif k comme $w_k = (1/u_k) \sum_{i=1}^{K} u_i$.

2.3 Calculez l'adéquation de la solution comme $F(x) = \sum_{k=1}^{K} (w_k f_k(x))$.

3 Calculer la probabilité de sélection de chaque solution $x \in P_t$ comme suit :

$$p(x) = ((F(x) - F^{min})^{-1} \sum_{y \in P_t} (F(y) - F^{min})$$

$$ou \quad F^{min} = minF(x)|x \in P_t$$

4 Sélectionner les parents en utilisant les probabilités de sélection calculé à l'étape 3.

5 Appliquer le croisement sur la sélection paires de parents pour créer N progéniture. Muter la progéniture avec un taux de mutation prédéfini. Copiez tous les descendants dans P_{t+1} Mettez à jour E si nécessaire.

6 Éliminer aléatoirement n_E solutions de P_{t+1} et ajoutez le même nombre de solutions de E à P_{t+1}.

7 **Si** la condition d'arrêt n'est pas remplie, définissez $t = t + 1$ et passez à l'étape 2.

Sinon revenez à E.

Fin

Le principal avantage de l'approche de la somme pondérée est la facilité de la mise en œuvre. Puisque dans l'approche de la somme pondérée un seul objectif est utilisé, son adaptation par les GA peut être utilisé avec un minimum de modifications. De plus, cette approche est efficace sur le plan informatique. Le principal inconvénient de cette approche est que tous les états de l'ensemble Pareto-optimaux des solutions ne peuvent être étudiées lorsque le vrai front de

Pareto est non convexe. Par conséquent, une AG à objectifs multiples basée sur la méthode de la somme pondérée a du mal à trouver des solutions uniformément réparties sur un marché non convexe hors surface.

1.5.2.1.2 Sélection parallèle (VEGA) (Vector Evaluated Genetic Algorithm)[2]

Comme mentionné précédemment, VEGA est le premier GA utilisé pour se rapprocher de l'ensemble de Pareto-optimal par un ensemble de solutions dominées. Dans VEGA, la population P_t est aléatoirement divisé en K sous-populations de taille égale ; P_1, $P_2...P_K$. Ensuite, chaque solution de la sous-population P_i se voit attribuer une valeur de fitness basée sur la fonction objectif z_i. Les solutions sont sélectionné parmi ces sous-populations en utilisant des sélection pour croiser et muter croiser et muter sont effectuées sur la nouvelle population dans le même manière que pour un seul objectif GA.

Procédure VEGA :

N_s :taille de la sous-population tel que :

$N_s = N/k$.

✓ **L'algorithme associé :**

Début

1 Commencer avec une population initiale aléatoire P_0 poser $t = 0$.

2 Si le critère d'arrêt est satisfait, renvoyez P_t.

3 Trier aléatoirement la population P_t.

4 **Pour** chaque objectif k ;$k = 1...K$, effectuez les etapes suivantes :

4.1 **Pour** $i = 1 + (k-1)N_s$,...kN_s assigner la valeur de $F(x_i) = f_k(x_i)$ à la $i^{ème}$ solution de la population triée.

4.1.1 Basé sur les valeurs de condition physique assignées à l'étape passer.

4.1.2 Sélectionner N_s solutions entre le $(1 + (k-1)N_s)$eme et le kN_s eme solutions de la population triée à créer une sous-population P_t et appliquez le croisement et mutation sur la population combinée à pour créer P_{t+1} de taille N poser $t = t + 1$ aller a 2.

4.2 **Finpour**

5 **Finpour** 6 Combinez toutes les sous-populations $P_1...P_k$ et appliquez le croisement et

la mutation sur la population combinée à créer P_{t+1} de taille Poser $t = t + 1$ aller a 2.

Fin

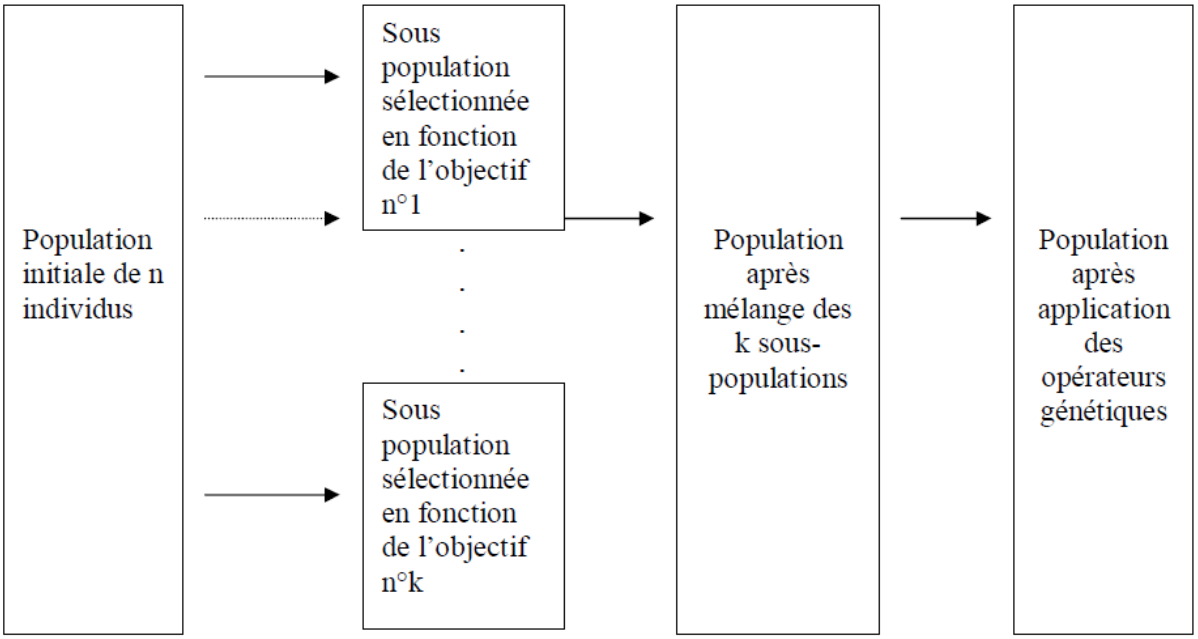

FIGURE 1.17 – Schéma de fonctionnement de VEGA

Une approche similaire à VEGA consiste à utiliser un seul fonction objective qui est déterminée au hasard chaque fois en phase de sélection Le principal avantage de l'approche par objectifs alternés est facile à mettre en œuvre et aussi efficace qu'un calculateur à objectif unique. En fait, cette approche est une extension directe d'un GA à objectif unique pour résoudre des problèmes multiobjectifs. L'inconvénient majeur du basculement objectif est que la population a tendance à converger vers des solutions supérieures en termes d'un objectif, mais pauvre chez les autres.

1.5.2.1.3 La méthode M.O.G.A (Multiple Objective Genetic Algorithm)[40]

En 1993 Fonseca et Flemming ont proposé une méthode dans laquelle chaque individu de la population est rangé en fonction du nombre d'individus qui le dominent. Ensuite, ils utilisent une fonction de notation permettant de prendre en compte le rang de l'individu et le nombre d'individus ayant même rang.
Soit un individu x_i à la génération t, dominé par $p_i(t)$ individus. Le rang de cet individu est :

$$Rang(x_i, t) = 1 + p_i(t)$$

Tous les individus non dominés sont de rang 1.
Fonseca et Fleming calculent la fitness de chaque individu de la façon suivante :

1. Calcul du rang de chaque individu.
2. Affectation de la fitness de chaque individu par application d'une fonction de changement d'échelle (on parle aussi de fonction de scaling) sur la valeur de son rang. Cette fonction est en général linéaire suivant le problème. D'autres types de fonction pourront être envisagés afin d'augmenter ou de diminuer l'importance des meilleurs rangs ou d'atténuer la largeur de l'espace entre les individus de plus fort rang et de plus bas rang.

L'utilisation de la sélection par rang a tendance à répartir la population autour d'un même optimum. Or cela n'est pas satisfaisant pour un décideur car cette méthode proposera qu'une seule solution. Pour éviter cette dérive, les auteurs utilisent la fonction du sharing. Ils espèrent ainsi répartir la population sur l'ensemble de la frontière Pareto.

Cette méthode obtient des solutions de bonne qualité et son implémentation est facile. Mais les performances sont dépendantes de la valeur du paramètre utilisé dans le sharing.

✓ **Algorithme de la méthode M.O.G.A :**

Début
1 Initialisation de la population.
2 Evaluation des fonctions objectif.
3 Assignation d'un rang basé sur la dominance.
4 Assignation d'une efficacité à partir du rang.
5 **Pour** i=1 to genmax (nombre maximal de générations).
5.1 Sélection (aléatoire proportionnelle à l'efficacité).
5.2 Croisement.
5.3 Mutation.
5.4 Evaluation des fonctions objectif.
5.5 Assignation d'un rang basé sur la dominance.
5.6 Assignation d'une efficacité à partir du rang.
6 **Finpour**
Fin.

Il existe comme cité précédemment beaucoup d'autre extensions de l'algorithme génétiques multiobjectifs.

1.6 Optimisation par colonies de fourmis(OCF)

1.6.1 Colonies de fourmis pour les problèmes mono-objectifs

La méthode de colonies de fourmis est apparue d'une constatation simple : les insectes sociaux, et en particulier les fourmis, résolvent naturellement des problèmes complexes. Un tel comportement est possible car les fourmis communiquent entre elles de manière indirecte par le dépôt de substances chimiques, appelées phéromones, sur le sol. Ce type de communication indirecte est appelé stigmergie. En effet, si un obstacle est introduit sur le chemin des fourmis, ces dernières vont, après une phase de recherche, avoir tendance toutes à prendre le plus court chemin entre le nid et l'obstacle. Plus le taux de phéromone à un endroit donné est important, plus une fourmi va avoir une grande probabilité à être attirée par cette zone. Les fourmis qui sont arrivées le plus rapidement au nid en passant par la source de nourriture sont celles qui ont pris la branche la plus courte du trajet. Il en découle donc que la quantité de phéromones sur ce trajet est plus importante que sur l'arc le plus long. De ce fait, le plus court chemin a une probabilité plus grande d'être pris par les fourmis que les autres chemins, et il sera donc pris par toutes les fourmis.

1.6.1.1 Procédure

Plusieurs problèmes combinatoires peuvent être résolus par un algorithme OCF exige de définir [38] :

- Une représentation graphique du problème avec une recherche collective des solutions par plusieurs agents simples (fourmis).
- Le procédé auto-organisationnel (positive feedback process).
- Une heuristique pour guider la recherche.

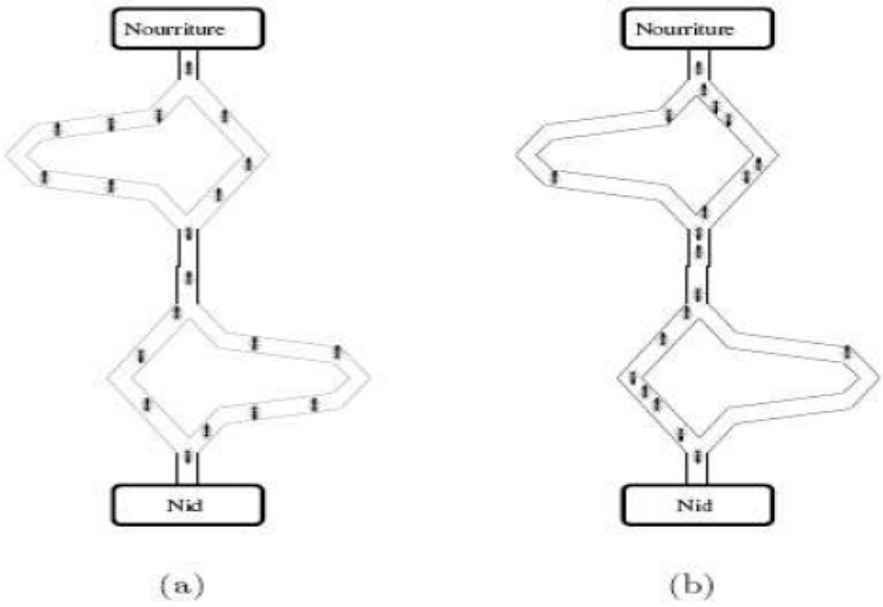

FIGURE 1.18 – Expérience du double point binaire :(a) au début de l'expérience, (b) à la fin de l'expérience.

1.6.1.2 Principe

Le premier algorithme qui s'inspire de cette analogie a été proposé en 1996 par Colorni, Dorigo et Maniezzo . Le but initial de cet algorithme était de résoudre le problème du voyageur de commerce. Si l'on considère un problème de voyageur de commerce à N villes, chaque fourmi k parcourt le graphe et construit un trajet de longueur $n = |N|$ [23].
Pour chaque fourmi, le trajet d'une ville i à une ville j dépend de :

♣ La liste des villes déjà visitées, qui définit les mouvements possibles à J_i^k chaque pas, quand la fourmi k est sur la ville i .

♣ L'inverse de la distance entre les villes .$\eta_{ij} = 1/d_{ij}$ appelée visibilité. Cette information est utilisée pour diriger les fourmis vers des villes proches et ainsi, éviter de trop longs déplacements .

♣ La quantité de phéromone déposée sur l'arête reliant deux villes τ_{ij} , appelée intensité de la piste. Cette quantité définit l'attractivité d'une piste, et elle est modifiée après le passage d'une fourmi. C'est la pseudo-mémoire du système.

La règle de déplacement est la suivante :

$$p_{ij}^k(t) = \begin{cases} \frac{(\tau_{ij}(t))^\alpha (\eta_{ij})^\beta}{\sum_{l \in J_i^k} (\tau_{il}(t))^\alpha (\eta_{il})^\beta} & si \;\; j \;\; \in \;\; J_i^k \\ 0 & si \;\; j \;\; \notin \;\; J_i^k \end{cases} \tag{1.6}$$

où α et β sont deux paramètres contrôlant l'importance relative de l'intensité et de la visibilité. Après un tour complet, chaque fourmi dépose une quantité de phéromone $\Delta\tau_{ij}^k(t)$ sur l'ensemble de son parcours. Cette quantité dépend de la qualité de la solution trouvée, et elle est définie par :

$$\Delta\tau_{ij}^k(t) \begin{cases} \frac{Q}{L^k(t)} & si \;\; (i,j) \;\; \in \;\; T^k(t) \\ 0 & si \;\; (i,j) \;\; \notin \;\; T^k(t) \end{cases} \tag{1.7}$$

où $T^k(t)$ est le trajet effectué par la fourmi k à l'itération t ;$L^k(t)$ est la longueur de $T^k(t)$ et Q est un paramètre fixé .

Enfin, il est nécessaire d'introduire un processus d'évaporation des phéromones.En effet, pour éviter de se faire piéger dans des solutions sous optimales, il est nécessaire qu'une fourmi "oublie" les mauvaises solutions.
La règle de mise à jour est donc.

$$\tau_{ij}(t+1) = (1-\rho)\tau_{ij}(t) + \Delta\tau_{ij}(t) \tag{1.8}$$

où $\Delta\tau_{ij}(t) = \sum_{k=1}^{m} \Delta\tau_{ij}^{k}(t)$ et m est le nombre de fourmis.
Depuis le développement de la première méthode basée sur l'analogie de colonies de fourmis, elle a été étendue à la résolution de plusieurs problèmes d'optimisation, discrets et continus.
Plusieurs variantes de colonies de fourmis sont proposées dans la littérature.
✓ **Algorithme générale de colonies de fourmis est présenté comme suit [34] :**

Début
1 initialiser : créer une population initiale de fourmis
2 répéter
2.1 Pour chaque fourmi **faire**
2.1.1 construire une solution basée sur la procédure de construction ,par les traces de phéromones
2.1.2 actualiser les valeurs des traces de phéromones pour améliorer la qualité des solutions trouver
Finpour
3 jusqu'a le critère d'arrêt soit satisfait
Fin

1.6.2 Colonies de fourmis pour les problèmes multiobjectifs

1.6.2.1 Taxonomie des algorithmes MOACO

Pour résoudre un problème d'optimisation multiobjectif (PMO) avec ACO, plusieurs points sont à définir pour lesquels différents choix sont possibles. Une taxonomie de plusieurs algorithmes MOACO. Cette taxonomie classe les algorithmes MOACO par le nombre de structures de phéromone et le nombre d'informations heuristiques [4].
Nous considérons, dans ce travail, que ces algorithmes diffèrent essentiellement dans les points suivants :

1.6.2.1.1 Structures de phéromone

Les traces de phéromone représentent le moyen qu'utilise une fourmi pour attirer les autres fourmis de sa colonie vers les aires correspondantes. Pour le cas uni-objectif, ces traces reflètent la désirabilité de visiter une zone de l'espace de recherche suivant la fonction à optimiser. En traitant un PMO, où on a plusieurs objectifs à optimiser simultanément, deux choix s'offrent :

1. On peut utiliser une seule structure de phéromone Dans ce cas, les traces de phéromone déposées par une fourmi sont définies relativement à une agrégation des objectifs à optimiser.
2. Une deuxième possibilité consiste à considérer plusieurs structures de phéromone Dans ce cas, on peut utiliser plusieurs colonies de fourmis, chacune utilise sa propre phéromone.

1.6.2.1.2 Définition du facteur phéromone

A chaque étape de la construction de solution d'une fourmi, un candidat est choisi relativement à une probabilité de transition qui dépend de deux facteurs : un facteur phéromone et un facteur heuristique. La définition du facteur phéromone dépend de la définition de la structure de phéromone, comme discuté dans le premier point.

1. Lorsqu'une seule structure phéromone est utilisée, le facteur phéromone est défini relativement à cette structure.
2. Lorsque plusieurs structures phéromone sont utilisées, on peut soit utiliser pour chaque objectif à optimiser la structure phéromone correspondante, soit utiliser toutes les structures phéromone. Dans ce dernier cas, généralement une agrégation des structures phéromone est utilisée : une somme pondérée ou un produit pondéré

1.6.2.1.3 Définition du facteur phéromone

Ce facteur heuristique doit donner une idée sur l'apport du candidat en question à la solution en cours de construction. Deux stratégies différentes ont été considérées :

1. une première stratégie est de considérer une agrégation des différents objectifs dans une seule information heuristique et d'utiliser cette information comme facteur heuristique
2. une deuxième stratégie consiste à associer à chaque objectif une structure heuristique comme dans la définition du facteur phéromone, on peut soit utiliser séparément chaque fonction heuristique pour l'objectif correspondant soit agréger les différentes structures heuristiques

1.6.2.1.4 Solutions à récompenser

Lors de la mise à jour de phéromone, on doit décider sur quelles solutions construites déposer la phéromone :

1. Une première possibilité, appelée l'approche Pareto, consiste à récompenser les solutions non-dominées de l'ensemble Pareto du cycle courant. Pour cette dernière possibilité on peut encore décider si on va récompenser toutes les solutions de l'ensemble Pareto ou bien seulement les nouvelles solutions non-dominées qui entrent dans l'ensemble au cycle courant
2. Une deuxième possibilité, appelée l'approche non Pareto, est de récompenser chacune des solutions qui trouve la meilleure valeur pour chaque critère du cycle courant

Récemment plusieurs travaux ont proposé des algorithmes basés sur l'optimisation par colonies de fourmis pour résoudre des PMO, appelés dans la littérature algorithmes MOACO.
Nous présentons dans cette section les algorithmes MOACO les plus connus qui ont été proposés dans la littérature[4] :

1.6.2.2 Multiple Objective Ant-Q

L'algorithme "Multiple Objective Ant-Q" (MOAQ) a été proposé par Mariano et Morales [26] dans pour résoudre le problème de distribution de l'eau dans les réseaux d'irrigation. Cet algorithme est basé sur un algorithme d'apprentissage renforcé distribué appelé Ant-Q
L'idée de base de MOAQ est de réaliser un algorithme d'optimisation avec une famille d'agents (fourmis) pour chaque objectif. Chaque famille k essaye d'optimiser un objectif en considérant les solutions trouvées pour les autres objectifs et leur fonction correspondante HE^k. De cette manière, tous les agents des différentes familles travaillent dans le même environnement en proposant des actions et une valeur de récompense r qui dépend de comment leurs actions ont participé à trouver des solutions de compromis parmi les autres agents.
MOAQ présente d'autres caractéristiques spécifiques. D'abord, lors de la construction de solution, la fourmi de la $j^{ème}$ famille utilise la solution trouvée par la $j^{ème}$ fourmi de la famille $i-1$ .De plus, lorsqu'une solution trouvée n'est pas faisable, l'algorithme applique une pénalité à ses composants sur les valeurs de Q. Finalement, tout au long du processus, les solutions non-dominées sont enregistrées dans un ensemble externe.

1.6.2.3 Multiple Objective-Ant

L'algorithme Bicriterion−Ant a été proposé par Iredi et al [19]. spécialement pour résoudre le problème de tournée de véhicules bi-critère. Pour ce faire, il utilise deux structures de traces de phéromone différentes,τ et τ' , une pour chaque critère considéré.
A chaque génération, chacune des m fourmis de la colonie génère une solution au problème. Durant l'étape de construction, la fourmi choisit le prochain sommet j à visiter relativement à la probabilité suivante :

$$p(ij) = \frac{\tau_{ij}^{\lambda\alpha} * {\tau'}_{ij}^{(1-\lambda)\alpha} * \eta_{ij}^{\lambda\alpha} * {\eta'}_{ij}^{(1-\lambda)\alpha}}{\sum_{u\in\Omega} \tau_{iu}^{(\lambda\alpha)} * {\tau'}_{iu}^{((1-\lambda)\alpha)} * \eta_{iu}^{\lambda\alpha} * {\eta'}_{iu}^{(1-\lambda)\alpha}} \tag{1.9}$$

où η et η' sont les valeurs heuristiques associées à l'arête a_{ij} relativement au premier et second objectif, respectivement Ω est le voisinage faisable de la fourmi, et λ est calculé pour chaque fourmi h,$f \in 1..m$ comme suit :

$$\lambda_h = \frac{h-1}{m-1} \tag{1.10}$$

Une fois que toutes les fourmis ont construit leurs solutions, les traces de phéromone sont évaporées par la règle habituelle. Ensuite, chaque fourmi qui a généré une solution dans le front Pareto au cycle courant est autorisée à mettre à jour les deux structures phéromone τ et τ' en déposant une quantité égale à $\frac{1}{l}$ ou l est le nombre de fourmis qui sont en train de mettre à jour les traces de phéromone. Les solutions non-dominées générées tout au long de l'exécution de l'algorithme sont mises dans un ensemble externe.

1.6.2.4 L'algorithme bicriterion MC

Iredi et al [19] ont proposé dans le même travail un autre algorithme MOACO, "BicriterionMC", très similaire au précédent BicriterionAnt. La différence principale est que chaque fourmi met à jour une seule structure de phéromone dans le nouvel algorithme. Les auteurs introduisent une définition générale pour l'algorithme basée sur l'utilisation de p structures de traces de phéromone et p par la suite mettent $p = 2$ pour résoudre des problèmes bi-critères. Pour ce faire, ils considèrent les deux méthodes différentes pour la mise à jour des traces de phéromone.

1. **Mise à jour par origine :**une fourmi dépose des traces de phéromone seulement dans sa colonie. Cette méthode force les deux colonies à chercher dans des régions différentes du front Pareto. L'algorithme utilisant cette première méthode est appelé "UnsortBicriterion".
2. **Mise à jour par région :**la séquence des solutions sur le front Pareto est divisée en p parties de taille égale. Les fourmis qui ont trouvé des solutions $i^{ème}$ dans la partie dépose de la phéromone pour la colonie i ;$i \in [1, p]$ Le but est de guider explicitement les fourmis des colonies de chercher dans des régions différentes du front Pareto, chacune dans une région. L'algorithme utilisant cette méthode est appelé "BicriterionMC".

Pinto et Baràn proposent un algorithme similaire à celui de iredi et al [19]. Le concept d'hétérogénéité est utilisé afin de modifier les deux algorithmes mono objectif Max-Min Ant System et le Ant Colony System Cette approche prend trois informations heuristiques η_1,η_2,η_3 pour optimiser simultanément trois fonctions objectifs du problème de routage multicaste Une variable λ_i est proposée pour chaque information heuristique η_i, la probabilité de choisir un nœud j tandis qu'une fourmi visite le nœud i est donnée par la formule suivante [38] :

$$p(ij) = \frac{\tau_{ij}^{\alpha} * ({\tau 1}_{ij}^{\lambda_1} {\tau 2}_{ij}^{\lambda_2} {\tau 3}_{ij}^{\lambda_3})^{\beta}}{\sum_{u\in\Omega} \tau_{iu}^{(\alpha)} ({\tau 1}_{iu}^{\lambda_1} {\tau 2}_{iu}^{\lambda_2} {\tau 3}_{iu}^{\lambda_3})^{\beta}} \tag{1.11}$$

Les solutions de l'ensemble Pareto sont utilisées pour effectuer la mise à jour de la mémoire de phéromone τ .

1.6.2.5 Pareto ant colony optimization

"Pareto Ant Colony Optimization" (P-ACO) a été proposé par Doerner et al [12].Il a été initialement appliqué au problème de sélection de portefeuille multiobjectif. Il suit le schéma général de l'algorithme ACS, mais la mise à jour globale de phéromone est réalisée en utilisant deux fourmis, la meilleure et la deuxième-meilleure solution générée dans le cycle courant pour chaque objectif k .Dans P-ACO, plusieurs structures phéromone τ_k sont considérées, une pour chaque objectif. A chaque cycle de l'algorithme, chaque fourmi calcule un ensemble de poids $p = (p_1; ....; p_k)$ et l'utilise pour combiner les traces de phéromone et l'information heuristique. Les solutions non-dominées trouvées tout au long de l'exécution sont là aussi enregistrées dans un ensemble externe.

1.6.2.6 Multiple ant colony system pour le problème de tourées de véhicules avec fenêtres de temps

l'algorithme "Multiple Ant Colony System for Vehicle Routing Problem with Time Windows" (MACS-VRPTW) a été introduit par Gambardella et al [20, 14].Il utilise, comme P-ACO, le schéma de ACS.
MACS-VRPTW est organisé avec une hiérarchie des colonies de fourmis conçue pour optimiser successivement une fonction multiobjectif :la première colonie, ACSVEI, minimise le nombre de véhicules alors que la deuxième,ACS-TIME, optimise les solutions faisables trouvées par la première. Chaque colonie utilise une structure phéromone indépendante pour son objectif spécifique et elles collaborent en partageant la meilleure solution globale trouvée ψ^{gb}. Lorsque ACS-VEI est activée, elle essaie de trouver une solution faisable avec un véhicule en moins que ceux utilisés dans ψ^{gb}. Le but de ACS-TIME est d'optimiser le temps total de la tournée des solutions qui utilisent le même nombre de véhicules utilisé dans ψ^{gb}. A chaque fois une meilleure solution est trouvée l'algorithme tue les deux colonies, et le processus est réitéré : deux nouvelles colonies sont activées travaillant avec la nouvelle solution.

1.6.2.7 Multiple ant colony system

Multiple Ant Colony System (MACS) est une variante de MACS-VRPTW décrit dans la section précédente. Il utilise aussi ACS, mais contrairement à son prédécesseur, MACS utilise une seule structure phéromone τ, et plusieurs informations heuristiques, η_k, initialement deux, $\eta_0$et η_1. De cette manière, une fourmi se déplace d'un sommet i à un sommet j en appliquant la règle suivante :

$$j = \begin{cases} arg\max_{j\in\Omega}(\eta_{ij}.[\eta^0_{ij}]^{\lambda\beta}[\eta^1_{ij}]^{(1-\lambda)\beta}) & siq < q_0 \\ \hat{\imath} & sinon. \end{cases} \tag{1.12}$$

ou λ est calculée pour chaque fourmi h comme $\lambda = h/m$ avec m est le nombre de fourmis, et $\hat{\imath}$ est le sommet choisi relativement a la probabilité suivante :

$$p(ij) = \frac{\tau_{ij} * (\eta_{ij}.[\eta^0_{ij}]^{\lambda\beta}[\eta^1_{ij}]^{(1-\lambda)\beta})}{\sum_{u\in\Omega}(\tau_{iu}.[\eta^0_{iu}]^{\lambda\beta}[\eta^1_{iu}]^{(1-\lambda)\beta})} \tag{1.13}$$

A chaque fois une fourmi traverse une arête a_{ij}, elle réalise une mise à jour de phéromone locale comme suit :

$$\tau_{ij} = (1-\rho)\tau_{ij} + \rho.\tau_0 \tag{1.14}$$

Initialement, τ_0 est calculé d'un ensemble de solutions heuristiques en prenant leurs moyennes dans chaque fonction objectif, f_0,f_1 et en appliquant l'expression suivante :

$$\tau_0 = \frac{1}{\hat{f}_0 0.\hat{f}_1} \tag{1.15}$$

Cependant, la valeur deτ_0 ne reste pas fixe, comme habituellement dans ACS, mais elle est mise à jour durant l'exécution de l'algorithme, avec la formule précédente avec les nouvelles valeurs des fonctions objectifs des solutions non dominées trouvées
La mise à jour globale est réalisée avec chaque solution S de l'ensemble Pareto courant en appliquant la règle suivante :

$$\tau_{ij} = (1-\rho).\tau_{ij} + \frac{\rho}{f^0(S).f^1(S)} \tag{1.16}$$

Il existe beaucoup d'autre travaux pour traiter un problème multiobjectifs avec colonies de fourmis (surtout les problèmes bi-objectif).

Conclusion

Au niveau de ce chapitre, nous avons présenté en premier lieu les problèmes d'optimisation combinatoire. Ensuite nous avons introduit les algorithmes génétiques et leur principe de fonctionnement. Enfin, nous avons présenté l'optimisation par colonies de fourmis.
Ces deux outils seront utilisés dans ce mémoire comme outils principaux d'optimisation pour la résolution du PDPTW .

CHAPITRE 2

RAPPELS SUR CERTAINS PROBLÈMES DE TRANSPORT

De nos jours, le problème de transport occupe une place importante dans la vie économique de notre société. Les entreprises ont découvert qu'une meilleure planification des tournées permettrait d'épargner des coûts importants. Effectivement, les coûts de transport effectués à travers la chaîne logistique et durant le processus de vente représentent entre 10 à 25 % du coût total de production. Selon CASS (Centre Des Droits Des Entreprises) Information Systèmes, cela peut représenter jusqu'à 57% des coûts logistiques, surpassant ainsi les coûts de stockage 31% et d'entreposage 8%. Les problèmes étudiés en transports sont variés. Ce chapitre permettra de toucher les grandes tendances des dernières années pour ce qui est niveau des problèmes. Dans ce chapitre nous donnerons un apperçu sur le problème de transport ses variantes, et les différentes contraintes que peut avoir un problème de transport.

2.1 Le problème de transport et le problème de flot

2.1.1 Problème de flot de valeur maximale à coût minimal

Il arrive souvent que l'on ait à représenter un réseau de transport de marchandises ou un réseau électrique au moyen d'un graphe. Le problème de flot dans ces réseaux revient a étudier la circulation de matières continues ou discrètes sur les arcs du graphe et qui obéit au principe de conservation. (ie : chercher le flot maximal de coût minimal).

2.1.1.1 Notion de base

2.1.1.1.1 Réseau de transport

Le réseau de transport est un graphe fini, sans boucle comportant une entrée X_1(source) et une sortie X_P (puits), telles que : depuis X_1 il existe un chemin vers tout autre sommet X_k et de tout sommet X_k il existe un chemin vers X_p. Tout arc u est valué par un entier positif $C(u)$, nommé capacité de l'arc u, qui représente une capacité de transport associée à la liaison figurée par cet arc (Ex. tonnages disponibles sur des bateaux, des camions, ...).

2.1.1.1.2 Flux

Un flux est la quantité $\varphi(u)$ transportée sur chaque arc u.

2.1.1.1.3 Flot

Un flot ϕ est déterminé par la donnée du flux pour tout arc du réseau de transport.
On definit un flot comme une fonction :

$$f : E \longrightarrow \mathbb{R}.$$
$$e \longmapsto f(e).$$

La valeur d'un flot $V(\phi)$ est par définition, la somme des flux partant de la source $X_1(V(\phi))$ est aussi égale à la somme des flux des arcs arrivant sur le puits X_p.

2.1.1.2 Problème de flot de valeur maximale à coût minimal

2.1.1.2.1 Présentation

Connaissant les capacités des arcs d'un réseau de transport et les coûts unitaires de transport sur chaque arc, le problème du flot maximum consiste à trouver la quantité maximale de flot qui peut circuler de la source à la destination au moindre coût. L'algorithme le plus connu pour résoudre ce problème est celui de B. Roy [33].

2.1.1.2.2 Formulation

- R est un réseau de transport où s et p désignent respectivement la source et le puits.
- A chaque arc (i, j) sont associées deux valeurs positives [c_{ij}, p_{ij}] où c_{ij} est la capacité et p_{ij} est le coût unitaire associé à l'arc.
- Le coût d'un flot : est la somme des coûts sur tous les arcs du réseau.

$$\sum_{ij} \phi_{ij} * p_{ij}. \tag{2.1}$$

Problème à résoudre :

$$\begin{cases} \min \sum_{ij} \phi_{ij} * p_{ij}, & \\ \phi_{ij} \leq C_{ij} & \forall i \in \mathbb{E}, \forall j \in \mathbb{R} \\ \sum_{ij} \phi_{ij} = \sum_{ji} \phi_{ji}; & \forall (i,j) \in \mathbb{E}, (i \neq s, j \neq p) \\ \sum_{sj} \phi_{sj} = \sum_{jp} \phi_{jp} = V(\phi). & \end{cases}$$

2.1.2 Le problème de transport

On connait mal l'origine exacte du problème de transport. Cependant, il s'agit d'un des plus vieux problèmes combinatoires. Des mathématiciens s'y sont intéressés depuis le début du vingtième siècle cherchant à apporter une réponse à ce problème qui est assez simple [5][41] :
X_{ij} :la quantité transporté de l'origine i vers la destination j.
C_{ij} :le cout unitaire de transport de l'origine i vers la destination j.

Données : un ensemble de nœuds (un ensemble de m origines O_i et de n destinations D_j) et a_i la quantité disponible à l'origine i et b_j les besoins(quantité demandé) pour chaque destinations j.
Trouver : les quantités X_{ij} à transporter des diffèrent dépôts i vers les différentes destinations j.
Minimisant :la quantité envoyée de chaque source à chaque destination en minimisant les coûts de transport. Les coûts sont proportionnels aux quantités transportées.

Le modèle mathématique est en général présenté sous la forme suivante avec que les contraintes

sont les conditions qui obligent à satisfaire la demande et épuiser la disponibilité.

$$Min \sum_{i=1}^{m}\sum_{j=1}^{n} C_{ij}X_{ij}$$

$$\sum_{i=1}^{m} X_{ij} = b_j \quad j = 1...n, \tag{2.2}$$

$$\sum_{j=1}^{n} X_{ij} = a_i \quad i = 1...m, \tag{2.3}$$

$$X_{ij} \geq 0 \quad i = 1...m \; j = 1...n, \tag{2.4}$$

Tel que (2.2) représente les besoins de la destination j et (2.3) représente le disponibilité de la source i et (2.4) représente la non négativité des quantités
Il s'agit d'un programme linéaire avec $(m \cdot n)$ variables de décision,(n+m) contraintes fonctionnelles et $m \cdot n$ contraintes non négatives.

Remarque 1. C'est un problème de flot minimal sur un réseau particulier avec une quantité de flot fixée.

$$\sum_{i=1}^{m} a_i = \sum_{j=1}^{n} b_j. \tag{2.5}$$

✓ Condition nécessaire et suffisante :
Une condition nécessaire et suffisante pour l'existence d'une solution réalisable au problème transport est que :

$$\sum_{i=1}^{m} a_i \quad = \quad \sum_{j=1}^{n} b_j \qquad \forall (i,j) = \{1,...,n\}\{1,...,n\}. \tag{2.6}$$

✓ Problème de transport non équilibré [5] :
si

$$\sum_{i=1}^{m} a_i \quad \neq \quad \sum_{j=1}^{n} b_j \qquad \forall (i,j) = \{1,...,n\}\{1,...,n\}. \tag{2.7}$$

Ce problème du transport est connu comme un problème de transport déséquilibré. On a deux cas [5] :

$$\sum_{i=1}^{m} a_i \quad > \sum_{j=1}^{n} b_j \quad ou \quad \sum_{i=1}^{m} a_i \quad < \quad \sum_{j=1}^{n} b_j. \tag{2.8}$$

On pourra se ramener à l'énoncé précédent de la manière suivante [5] :

Si $\sum_{i=1}^{m} a_i \quad > \quad \sum_{j=1}^{n} b_j$ Il suffit d'introduire une destination fictive y_{n+1} de coût de transport égale à zéro entre x_i et y_{n+1} $(i = 1...m)$ dont la demande :

$$b_{n+1} = \sum_{i=1}^{m} a_i - \sum_{j=1}^{n} b_j$$

Si $\sum_{i=1}^{m} a_i \quad < \quad \sum_{j=1}^{n} b_j$ il suffit d'introduire une source fictive x_{n+1} de coût de transport égale à zéro entre y_i et x_{m+1} $(i = 1...m)$ dont la disponibilité :

$$a_{m+1} = \sum_{j=1}^{n} b_i - \sum_{i=1}^{n} a_i$$

✓ La résolution du problème de transport peut être résumée à l'aide de l'organigramme suivant :

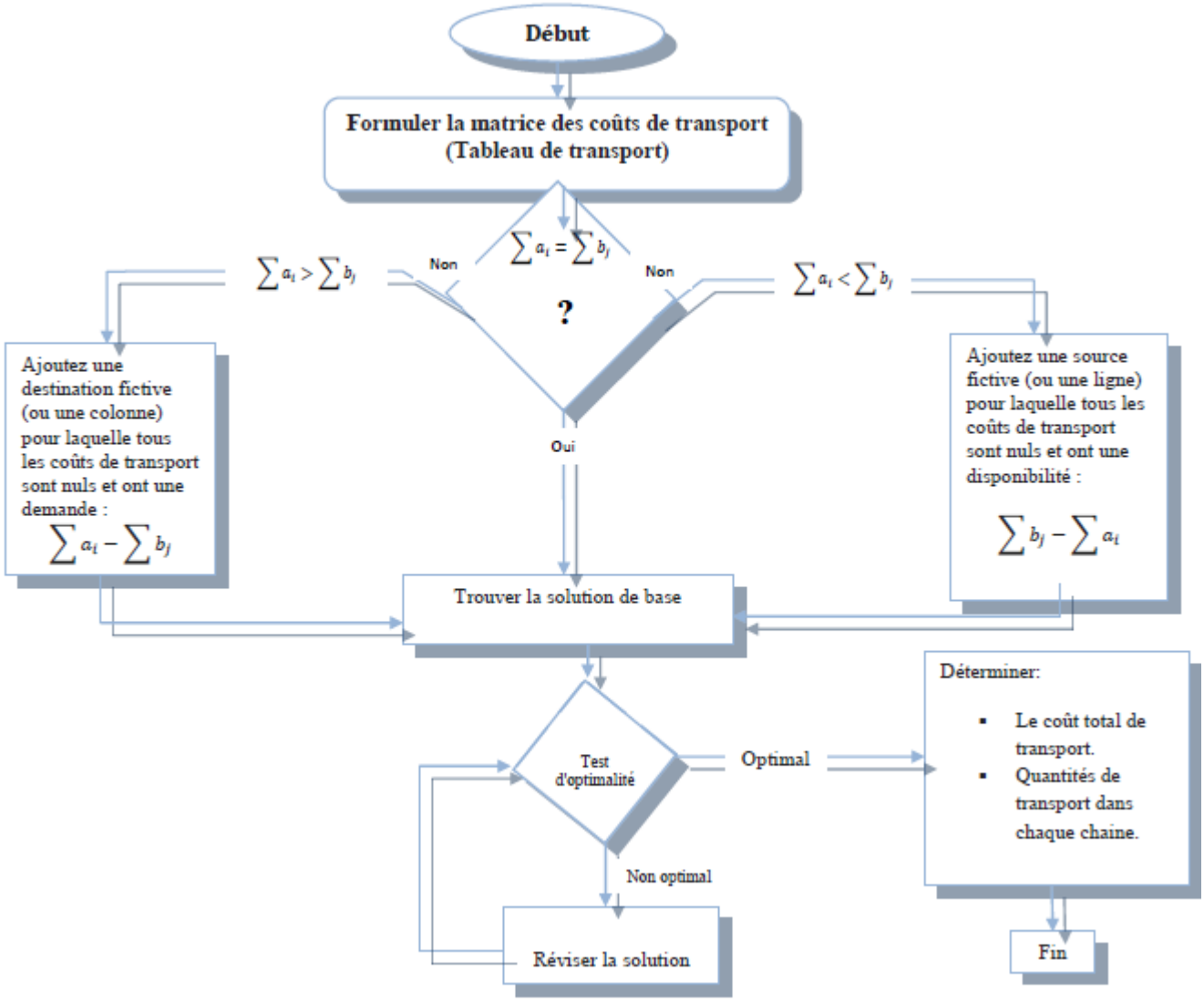

FIGURE 2.1 – Organigramme de la résolution de problème de transport.

Le problème de transport classique étant un problème basique et élémentaire, nous nous passerons des détails le concernant. Plusieurs références incontournables peuvent être consulées [33].

2.2 Le problème de tournées de véhicules (VRP)

Le VRP (The véhicule routing problem) constitue une généralisation du TSP à plusieurs voyageurs. On parle d'ensembles de véhicules :

Données : un ensemble de nœuds clients et d'arêtes (munies de couts) et une flotte illimitée de véhicules partant d'un unique dépôt.

Trouver : un ensemble de routes (tournées de véhicules) recouvrant tous les nœuds clients qui partent et reviennent au dépôt.

Minimisant :Le cout total de transport lié aux arcs empruntés par les véhicules et/ou les couts fixes associés à l'utilisation des véhicules[25]

Pour les N clients et les M véhicules, on définit :

$$x_{ij}^k = \begin{cases} 1 & \text{si le véhicule } k \text{ effectue le trajet } (i,j)\ . \\ 0 & \text{sinon.} \end{cases}$$

$$y_i^k = \begin{cases} 1 & \text{si le véhicule } k \text{ visite le client } i\ . \\ 0 & \text{sinon.} \end{cases}$$

Le modèle mathématique est le suivant :

$$minimiser\ z = \sum_{k\in\mathcal{K}} \sum_{i\in\mathcal{N}} \sum_{j\in\mathcal{N}} C_{ij} X_{ij}^k.$$

Sous les contraintes :

$$\sum_{i>1} y_i^k \leq \mid J \mid \times y_0^k \quad \forall k \in \mathcal{K}, \tag{2.9}$$

$$\sum_{k\in\mathcal{K}} yi^k = 1 \quad \forall i \neq 1 \in \mathcal{N}, \tag{2.10}$$

$$\sum_{i\in\mathcal{N}} x_{ij}^k = y_j^k \quad \forall j \in \mathcal{N}, \forall k \in \mathcal{K}, \tag{2.11}$$

$$\sum_{j\in\mathcal{N}} x_{ij}^k = y_i^k \quad \forall i \in \mathcal{N}, \quad \forall k \in \mathcal{K}, \tag{2.12}$$

$$\sum_{i,j\in\mathcal{S}} x_{ij}^k \leq \mid S \mid -1 \qquad \forall k = 1 \ldots \mathcal{M} \quad S \subset \mathcal{N} t.q \quad 2 \leq \mid \mathcal{S} \mid \leq n-2, \tag{2.13}$$

Tq :
J : Ensembles des clients. S : Ensembles des sommets. Chaque client doit être visité une fois, ce qui est assuré par la contrainte (2.9) . La contrainte (2.10) garantit que chaque noeud doit être visité une seulle fois par un et un seul véhicule. Les contraintes (2.11) et(2.12) sont le pendant pour le VRP de (2.2) et (2.3) (on arrive et on part de chez chaque client). Enfin, on retrouve les contraintes d'élimination des sous-tours en (2.13) .
En general, l'objectif à optimiser, pour les problèmes de tournées de véhicules peut être soit [8] :

- Minimiser les distances de parcours.
- Minimiser les temps de travail.
- Minimiser les coûts des tournées à planifier.
- Minimiser le nombre de véhicules à utiliser.
- Maximiser le gain à récolter.
- Minimiser le temps d'attente.
- Minimiser le retard maximal.

Les contraintes sont multiples et varient en fonction des objectifs, néanmoins,nous avons rencontré les principales contraintes suivantes [8] :

- Contraintes liées au véhicule (par exemple : la capacité du véhicule, localisation du véhicule, etc).
- Contraintes liées au client (par exemple : la priorité des clients lors d'un service, les fenêtres de temps du client, le score associé aux clients, etc.).
- Contraintes liées aux tournées (par exemple : la longueur maximale de la tournée en temps ou en distance).

Par exemple,voici certaines contraintes [13, 8] :

– **Les contraintes de capacité** sont dues aux capacités limitées des véhicules. Ces capacités peuvent être impossibles à dépasser, et la contrainte de capacité sera une contrainte rigide, comme elles peuvent être dépassées tout en ayant, dans ce cas, une pénalité à assumer

– **Les contraintes de temps** sont dues à la disponibilité limitée des clients. Ces contraintes sont appelées fenêtres de temps. Elles peuvent être impossibles à violer, et la contrainte de temps sera une contrainte rigide, comme elles peuvent être violées tout en ayant une pénalité à assumer(plus detailler dans la section qui vas suivre)

– **Fenêtre de temps :** c'est un intervalle de temps $[e_i, l_i]$ dont le quel le service doit commencer au plutôt à la date e_i, et au plus tard à la date l_i pour chaque site i, et si le véhicule arrive avant e_i, il doit attendre jusqu'à e_i pour commencer le service.
– **Les contraintes de précédences** sont présentes pour garantir qu'un client ne soit pas visité avant son fournisseur ou autrement dit le site de départ p_k (le k^{eme} site de cuillette) de chaque demande de transport k doit être visité avant son site de d'arrivé d_k (le k^{eme} site de livraison).

2.3 Capacited Vehicle Routing Problem (CVRP)

Le CVRP est une extension du VRP .Le critère le plus utilisé dans ce genre de problème est soit la minimisation de la distance totale parcourue, soit la minimisation du coût, par un nombre minimum de véhicules [13, 25]

Données :	Un ensemble de nœuds clients (ayant une demande) et d'arêtes (munies de coûts) et une flotte illimitée de véhicules de capacité uniforme Q partant d'un unique dépôt.
Trouver :	Des tournées de véhicules, satisfaisant chaque demande une et une seule fois et respectant les contraintes de capacité.
Minimisant :	Le coût total de transport liée aux arcs empruntés par les véhicules et/ou les coûts fixes associés à l'utilisation des véhicules[25]

Soient :

✓ $G = (N, A)$ un graphe où $N = \{0, ..., n\}$ est un ensemble de sommets avec le sommet 0 fixe comme dépôt et $A = \{(i, j) : i, j \in N \text{ et } i \neq j\}$ est l'ensemble des arcs.

$n =$	Nombre de sommets.
$m =$	Nombre de véhicules.
$D =$	Capacité d'un véhicule.
$T_k =$	Temps maximal de la tournée du véhicule k.
$d_i =$	Demande du sommet i.
$t_i^k =$	Temps nécessaire au véhicule k pour charger ou décharger au sommet i .
$t_{ij}^k =$	Temps nécessaire au véhicule k pour voyager du sommet i au sommet j.
$C_{ij} =$	Coût ou distance du voyage du sommet i au sommet j.

$$x_{ij}^k = \begin{cases} 1 & \text{Si le véhicule } k \text{ effectue le trajet } (i,j) \text{ .} \\ 0 & \text{Sinon.} \end{cases}$$

$$Minimiser \quad (\sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{m} C_{ij}x_{ij}^k)$$

$$\sum_{i=1}^{n}\sum_{k=1}^{m} x_{ij}^k = 1 \quad j = 2...n, \tag{2.14}$$

$$\sum_{j=1}^{n}\sum_{k=1}^{m} x_{ij}^{k} = 1 \quad i = 2...n, \tag{2.15}$$

$$\sum_{i=1}^{n}\sum_{k=1}^{m} x_{ip}^{k} - \sum_{j=1}^{n}\sum_{k=1}^{m} x_{pj}^{k} = 0 \quad k = 1...m \quad p = 1...m, \tag{2.16}$$

$$\sum_{i=1}^{n} t_{i}^{k} \sum_{j=1}^{n} x_{ij}^{k} + \sum_{i=1}^{n}\sum_{j=1}^{n} t_{ij}^{k} x_{ij}^{k} \leq T_{k} \quad k = 1...m, \tag{2.17}$$

$$\sum_{i=1}^{n} d_{i}(\sum_{j=1}^{n} x_{ij}^{k}) = D \quad k = 1...m, \tag{2.18}$$

$$\sum_{j=2}^{n} x_{1j}^{k} \leq 1 \quad k = 1...m, \tag{2.19}$$

$$\sum_{i=2}^{n} x_{i1}^{k} \leq 1 \quad k = 1...m, \tag{2.20}$$

Les équations (2.14) et (2.15) assurent que chaque nœud n'est servi qu'une seule fois par un et un seul véhicule.
L'équation (2.16) assure la continuité d'une tournée par un véhicule : le nœud visité doit impérativement être quitté.
L'équation (2.17) assure le respect de la contrainte de la durée totale d'une tournée.
L'équation (2.18) assure le respect de la contrainte de capacité du véhicule.
Les équations (2.19) et (2.20) assurent qu'un véhicule ne sort du dépôt et n'y revient qu'une seule fois.

2.4 Le VRP et ses variantes

Plusieurs dérivées du VRP sont apparues dues essentiellement aux activités des chercheurs qui travaillent de plus en plus sur les problèmes de transport et de distribution que rencontrent-les sociétés.
Nous allons faire dans ce qui suit un aperçu sur les variantes qui ont était développé pour arriver au PDPTW [21].

TSP	Travelling Salesman Problem	Le problème du voyageur de commerce consiste à trouver le chemin le plus court pour passer par l'ensemble des villes données une fois et une seule et revenir à la ville de départ. C'est donc un cas particulier du VRP sans contrainte de capacité et avec un seul véhicule.
VRP	vehicule routing problem	le problème de tourné de véhicules consiste à trouver un ensemble de routes (tournées de véhicules) recouvrant tous les nœuds clients qui partent et reviennent au dépôt dans le but de minimiser Le cout total de transport lié aux arcs empruntés par les véhicules et/ou les couts fixes associés à l'utilisation des véhicules
CVRP	Capacitated Vehicle Routing Problem	Un problème CVRP consiste à affecter chaque client à une tournée effectuée par un seul véhicule de capacité finie. Ce véhicule commence et termine sa tournée au dépôt.
VRPPD	Vehicle Routing Problem with Pickup and Delivery	Le problème de collecte et de livraison a les mêmes propriétés que le VRP. Il ajoute à celles-ci le fait que chaque client introduit deux positions géographiquement différentes : la première pour le ramassage du produit et la deuxième pour la livraison de ce dernier. Ceci va directement induire une contrainte de précédence à ajouter au problème du VRP classique sachant que, dans une tournée, chaque opération de livraison doit être précédée par l'opération de ramassage respective.
VRPTW	Vehicle Routing Problem with Time Windows	Le VRPTW est un des problèmes les plus étudiés. Dans un VRPTW, de nouvelles contraintes temporelles sont ajoutées : chaque client doit être servi dans un intervalle de temps durant lequel il est disponible pour être visité.
PDPTW	Pickup and Delivery Problem with Time Windows	Ce problème est une variante du VRPTW. Outre l'existence des contraintes de fenêtres de temps, ce problème possède un ensemble de clients et un ensemble de fournisseurs. A chacun de ces clients correspond un et un seul fournisseur. Les véhicules ne doivent alors passer par un client qu'après avoir visité son fournisseur (contraintes de précédences) .

Un problème $NP-$difficile est un problème dont l'existence d'un algorithme déterministe exact qui puisse le résoudre en un temps polynomial est peu probable. Vu que le VRP est $NP-$difficile alors ses extensions qui sont caractérisées par des contraintes additionnelles sont aussi des problèmes $NP-$difficiles.

2.5 The pick up and delivery problem with time windows

Le problème de PDPTW est une extension du problème de VRP (comme défini précédemment) le PDPTW peut être étudié sous deux formes statiques et dynamiques :
✓ Un problème est dit statique si toutes les données sont connues sur l'horizon de temps où l'on doit calculer les tournées.
✓ Dans le cas dynamique, le système doit être capable d'intégrer des imprévus alors que des tournées ont déjà été planifiées et que leur exploitation a démarré. Ces imprévus peuvent être la panne ou le retard d'un véhicule, l'apparition ou la disparition d'un ou de plusieurs nœuds (client / fournisseur)

Le PDPTW qui est le problème de collecte et distribution à fenêtres de temps (Pickup and Delivery Problem with Time Windows), se divise en deux : $1-$PDPTW (à un véhicule) et $m-$PDPTW (à plusieurs véhicules).

Conclusion

Au niveau de ce chapitre, nous avons présenté en premier lieu les différentes études réalisées dans le domaine de l'ordonnancement de transport à savoir le problème général de construction de tournées de véhicules (le VRP),le problème de flot,le CVRP. Passant de leur formulation mathématique et leur variantes .
Ensuite nous avons aperçu sur les variantes qui ont était développé pour arriver au PDPTW.Enfin nous avons définir le problème PDPTW.

CHAPITRE 3

RÉSOLUTION DU 1-PDPTW PAR LES ALGORITHMES GÉNÉTIQUES ET COLONIES DE FOURMIS

Dans ce chapitre nous allons présenter le m-PDPTW et le 1-PDPTW et leurs formulations mathématiques respectives. Nous détaillerons par le suite deux approches de résolution adaptées et implémentées. La première approche évolutionnaire est basée sur une nouvelle fonction objective et des opérateurs génétiques contrôlés. La deuxième approche est basée sur la méthode de colonies de fourmis, nous terminons par un exemple d'application pour les deux méthodes.

3.1 Le 1-PDPTW

Le problème 1-PDPTW concerne le cas traitant d'un seul véhicule. Comme tout problème d'ordonnancement, plusieurs travaux basés sur des méthodes exactes et sur des méthodes approchées s'y sont intéressés [29, 37, 35, 36, 28, 22].
L'application de ces méthodes s'est limitée, pour l'obtention de solutions optimales,à un nombre restreint de clients, le recours aux méthodes approchées a été le choix de plusieurs chercheurs.
On sait que la complexité algorithmique est exponentielle pour les problèmes d'ordonnancement et malgré la multitude des méthodes de résolution disponibles, il est souvent nécessaire de modifier ces dernières pour mieux les adapter au problème et pour améliorer ainsi les solutions obtenues.

3.1.1 Formulation mathématique

Nous présentons ici une formulation mathématique du 1-PDPTW, en supposant que :
- Il y a un seul fournisseur pour un seul client et un seul client pour un seul fournisseur.
- Un nœud n'est servi que par le véhicule une et une seule fois.
- Il y a un seul dépôt.
- Les contraintes de capacité doivent être respectées.
- Les contraintes de temps sont rigides concernant l'heure d'arrivée.
- Le véhicule commence le trajet du dépôt et y retourne à la fin.
- Le véhicule reste à l'arrêt à un nœud le temps nécessaire pour le traitement de la demande.
- Si le véhicule arrive au nœud i avant la date e_i de début de sa fenêtre, il le sert en comptant une pénalité égale au temps d'attente.

Les variables de données :

- $\mathcal{N}$: Ensemble des noeuds clients, fournisseurs et dépôt.
- N' : Ensemble des noeuds clients, fournisseurs.
- N^+ : Ensemble des noeuds fournisseurs.
- N_i^+ : Fournisseur du nœud i.
- N^- : Ensemble des noeuds clients.
- N_i^- : Client du nœud i.
- d_{ij} : Distance euclidienne entre le nœud i et le nœud j ,si $d_{ij} = \infty$ alors le chemin entre i et j n'existe pas (impasse, rue piétonne,...).
- t_{ij} : Temps mis par le véhicule pour aller du nœud i au nœud j.
- $[e_i, l_i]$: Fenêtre de temps du nœud i.
- s_i : Temps d'arrêt au nœud i.
- q_i : Quantité à traiter au nœud i, si $q_i > 0$, le nœud est un fournisseur. Si $q_i < 0$ le nœud est un client et si $q_i = 0$ alors le nœud a été servi.
- Q : Capacité du véhicule.
- $i = 0 \dots N$: indice des noeuds prédécesseurs.
- $j = 0 \dots N$: indice des noeuds successeurs.

Les variables de décisions :

- $x_{ij} = \begin{cases} 1 & \text{Si le véhicule voyage du nœud } i \text{ au nœud } j \text{ .} \\ 0 & \text{Sinon.} \end{cases}$
- A_i : Temps d'arrivée au nœud i.
- D_i : Temps de départ du nœud i, $D_i = A_i + s_i \; \forall \; i \in N$.
- y_i : Quantité présente dans le véhicule visitant le nœud i.

Le problème se formule comme suit :

$$Minimiser \quad (\sum_{i \in N} \sum_{j \in N} d_{ij} x_{ij}), \tag{3.1}$$

$$\sum_{j \in \mathcal{N}} x_{ij} = 1 \quad \forall i \in \mathcal{N}, \tag{3.2}$$

$$\sum_{i \in \mathcal{N}} x_{i0} = 1, \tag{3.3}$$

$$\sum_{j \in \mathcal{N}} x_{0j} = 1, \tag{3.4}$$

$$\sum_{i \in \mathcal{N}} x_{iu} - \sum_{j \in \mathcal{N}} x_{uj} = 0, u \in \mathcal{N}, \tag{3.5}$$

$$X_{ij} = 1 \implies y_j = y_i + q_i, \quad i, j \in \mathcal{N}, \tag{3.6}$$

$$y_0 = 0, \tag{3.7}$$

$$Q \geq y_i \geq 0 \forall i \in \mathcal{N}, \tag{3.8}$$

$$X_{ij} = 1 \Longrightarrow D_i + t_{ij} \leq D_j, \quad i, j \in \mathcal{N}, \tag{3.9}$$

$$D_w \leq D_v, \quad i \in N, w \in N_i^+, v \in N_i^-, \tag{3.10}$$

$$D_0 = 0, \tag{3.11}$$

$$x_{ij} = \{0, 1\}, \quad i = 0 \dots \mathcal{N}, \quad j = 0 \dots \mathcal{N}. \tag{3.12}$$

Les équations (3.2) assure que chaque sommet n'est visité qu'une seule fois par le véhicule.
Les équations (3.3) et (3.4) assurent que le véhicule ne sort du dépôt et n'y revient qu'une seule fois.
L'équation (3.5) assure la continuité d'une tournée par un véhicule : le sommet visité doit impérativement être quitté.
Les équations (3.6), (3.7) et (3.8) assurent le non dépassement de la capacité de transport du véhicule.
Les équations (3.9), (3.10)et (3.11) assurent le respect des contraintes de précédences.

La fonction à optimiser et les contraintes à respecter

Les fonctions à optimiser et les contraintes à respecter diffèrent d'un problème à un autre. Dans la suite, nous présentons les fonctions à optimiser et les contraintes adaptées pour chacune des approches de résolution du 1-PDPTW.

Le 1-PDPTW multiobjectif

Dans le cas multiobjectif les fonctions objectif du 1-PDPTW sont données de la manière suivantes :

$$Minimiser f = \begin{cases} f_1 = \sum_{i \in N} \sum_{j \in N} d_{ij} X_{ij}, \\ f_2 = \sum_{i \in N} \sum_{j \in N} \max(0, e_i - A_i), \\ f_3 = \sum_{i \in N} \sum_{j \in N} \max(0, D_i - l_i). \end{cases} \tag{3.13}$$

f_1 est la fonction objectif pour minimiser la distance parcouru.
f_2 est la fonction objectif pour minimiser le temps d'attente total.
f_3 est la fonction objectif pour minimiser le retard total.
Avec le respect des contrainte précédentes : (3.2),(3.3),(3.4),(3.5) (3.6), (3.7), (3.8),(3.9), (3.10)et (3.11)

Exemple 1. On prend comme donnée 4 noeuds avec la matrice des distances et des temps comme suit :

Distance /km	0	1	2	3	4
0	0	42.7	45.6	38	50.9
1	41.7	0	3.8	7.4	9.7
2	41.6	3.8	0	10.8	5.9
3	37.3	7.4	10.8	0	21.7
4	46.8	10.6	5.8	21.7	0

Temps/min	0	1	2	3	4
0	0	73	70	63	79
1	64	0	12	20	27
2	56	13	0	30	14
3	54	20	30	0	120
4	65	27	15	42	0

Les fenêtres de temps, le temps de service, ainsi que la quantité a traités à chaque noeuds et que l'on a associer à notre modèle sont donnée comme suit :

propriétés i	e_i	l_i	s_i	q_i
1	100	250	08	20
2	110	120	14	-20
3	280	380	20	20
4	300	400	06	-20

On pose la solution t par la suite suivante :

$$0 \rightarrow 1 \rightarrow 2 \rightarrow 3 \rightarrow 4 \rightarrow 0$$

Le calcule de la valeur des 3 fonctions objectifs est comme suit :

$f_1(t) = \ Distance_{0,1} + Distance_{1,2} + Distance_{2,3} + Distance_{3,4} + Distance_{4,0}$

Pour la fonction f_2 et f_3 :

$A_1 = D_0 + temps_{0,1} = 0 + 73 = 73 \Rightarrow Attente = max(0, e_1 - A_1) = max(0, 100 - 73) = 27;$

$D_1 = e_1 + s_1 = 100 + 8 = 108 \Rightarrow Retard = max(0, D_1 - l_1) = max(0, 108 - 250) = 0;$

$A_2 = D_1 + temps_{1,2} = 108 + 12 = 120 \Rightarrow Attente = Attente + max(0, e_2 - A_2) = 27 + max(0, 110 - 120) = 27;$

$D_2 = A_2 + s_2 = 120 + 14 = 134 \Rightarrow Retard = Retard + max(0, D_2 - l_2) = 0 + max(0, 134 - 120) = 14;$

$A_3 = D_2 + temps_{2,3} = 134 + 30 = 164 \Rightarrow Attente = Attente + max(0, e_3 - A_3) = 27 + max(0, 280 - 164) = 143;$

$D_3 = e_3 + s_3 = 280 + 20 = 300 \Rightarrow Reatrd = Retard + max(0, D_3 - l_3) = 14 + max(0, 300 - 380) = 14;$

$A_4 = D_3 + temps_{3,4} = 300 + 120 = 420 \Rightarrow Attente = Attente + max(0, e_4 - D_4) = 143 + max(0, 300 - 420) = 143;$

$D_4 = A_4 + s_4 = 420 + 6 = 426 \Rightarrow Retard = Retard + max(0, D_4 - l_4) = 14 + max(0, 426 - 400) = 14 + 26 = 40;$

Donc $f_2(t) = 143$ et $f_3(t) = 40$

3.2 Le m-PDPTW

L'objectif est de servir toutes les demandes des clients, en minimisant le coût total de transport. Ce coût est relatif au nombre de véhicules utilisés et à la distance parcourue par chaque véhicule.[13]

3.2.1 Formulation mathématique

- Un nœud (fournisseur ou client) n'est servi que par un seul véhicule.
- Il y a un seul dépôt.
- Les contraintes de capacité doivent être respectées.
- Les contraintes de temps sont rigides concernant l'heure d'arrivée.
- Chaque véhicule commence le trajet du dépôt et retourne à la fin.
- Un véhicule reste à l'arrêt à un nœud le temps nécessaire pour le traitement de la demande.
- Si un véhicule arrive au nœud i avant la date e_i de début de sa fenêtre, il doit attendre jusqu'à cet instant e_i.

Les variables de données

- $\mathcal{N}$: Ensemble des noeuds clients, fournisseurs et dépôt.
- N' : Ensemble des noeuds clients, fournisseurs.
- N^+ : Ensemble des noeuds fournisseurs.
- N^- : Ensemble des noeuds clients.
- K : Nombre de véhicules.
- d_{ij}^k : Distance euclidienne entre le nœud i et le nœud j parcourue par la véhicule k, si $d_{ij}^k = \infty$ alors le chemin entre i et j n'existe pas (impasse, rue piétonne,...).
- t_{ij}^k : Temps mis par le véhicule k pour aller du nœud i au nœud j.
- $[e_i, l_i]$: Fenêtre de temps du nœud i.
- s_i : Temps d'arrêt au nœud i.
- q_i : Quantité à traiter au nœud i, si $q_i > 0$, le nœud est un fournisseur. Si $q_i < 0$ le nœud est un client et si $q_i = 0$ alors le nœud a été servi.
- Q_k : Capacité du véhicule k.
- $i = 0 \dots N$: Indice des noeuds prédécesseurs.
- $j = 0 \dots N$: Indice des noeuds successeurs.
- $k = 1 \dots K$: Indice des véhicules.

Les variables de décisions

- $x_{ij}^k = \begin{cases} 1 & \text{Si le véhicule } k \text{ effectue le trajet } (i,j). \\ 0 & \text{Sinon.} \end{cases}$
- A_i : Temps d'arrivée au nœud i.
- D_i : Temps de départ du nœud i.
- y_{ik} : Quantité présente dans le véhicule k visitant le nœud i.
- C_k : Coût de transport par unité de distance associé au véhicule k.

La formulation du m-PDPTW est comme suit :

$$Minimiser \quad (\sum_{k\in K}\sum_{i\in N}\sum_{j\in N} C_k d_{ij}^k x_{ij}^k) \tag{3.14}$$

$$\sum_{i=1}^{N}\sum_{k=1}^{K} X_{ij}^k = 1, \quad j = 2...\mathcal{N}, \tag{3.15}$$

$$\sum_{j=1}^{N}\sum_{k=1}^{K} X_{ij}^k = 1, \quad i = 2...\mathcal{N}, \tag{3.16}$$

$$\sum_{i\in N} X_{i0}^k = 1, \quad \forall k \in K, \tag{3.17}$$

$$\sum_{j\in N} X_{0j}^k = 1, \quad \forall k \in K, \tag{3.18}$$

$$\sum_{i\in N} X_{iu}^k - \sum_{j\in N} X_{iu}^k = 0, \forall k \in K, \forall u \in \mathcal{N}, \tag{3.19}$$

$$X_{ij}^k = 1 \implies y_j^k = y_i^k + q_i \quad \forall i,j \in N \quad \forall k \in K, \tag{3.20}$$

$$Y_0^k = 0, \qquad \forall k \in K, \tag{3.21}$$

$$Q_k \geq y_j^k \geq 0, \quad \forall j \in N, \quad \forall k \in K, \tag{3.22}$$

$$D_w \leq D_v \quad , \forall v \in N^+ \quad , \forall v \in N^-, \tag{3.23}$$

$$D_0 = 0, \tag{3.24}$$

$$X_{ij}^k = 1 \implies D_i + t_{ij}^k \leq D_j \quad \forall i,j \in N \quad \forall k \in K, \tag{3.25}$$

Les équations (3.15) et (3.16) assurent que chaque nœud n'est servi qu'une seule fois par un et un seul véhicule.
Les équations (3.17) et (3.18) assurent qu'un véhicule ne sort du dépôt et n'y revient qu'une seule fois.
L'équation (3.19) assure la continuité d'une tournée par un véhicule : le nœud visité doit impérativement être quitté.
Les équations (3.20), (3.21) et (3.22) assurent le non dépassement de la capacité de transport d'un véhicule.
Les équations (3.23), (3.24)et (3.25) assurent le respect des précédences. Un client ne doit pas être visité avant son fournisseur.
Dans le cas multiobjectif la fonction de minimisation est donnée comme suit :

$$Minimiser f = \begin{cases} k, \\ \sum_{i\in N}\sum_{j\in N} \max(0, D_i - l_i), \\ \sum_{k\in K}\sum_{i\in N}\sum_{j\in N} C_k d_{ij}^k x_{ij}^k. \end{cases} \tag{3.26}$$

En respectant de même, les contraintes de précédence, de capacité des véhicules ainsi que la disponibilité des véhicules et le respect de la continuité des tournées. Ces dernières sont assurées par les équations (3.15), (3.16), (3.17), (3.18), (3.19), (3.20), (3.21), (3.22), (3.23), (3.24) et (3.25).

3.3 Résolution du problème 1-PDPTW multiobjectifs

Pour résoudre un problème de 1-PDPTW il existe plusieurs approches évolutionnistes. On s'intéresse à la première approche évolutionnaire qui est basée sur une nouvelle fonction objectif et des opérateurs génétiques contrôlés[21].

3.3.1 Première approche : algorithmes génétiques

3.3.1.1 Formulation mathématique adaptée

La fonction à optimiser : adaptée dans cette $1^{ère}$ approche évolutionniste pour la résolution du 1-PDPTW est la suivante :

$$Minimiser f = \begin{cases} \alpha_1 \sum_{i\in N} \sum_{j\in N} d_{ij} X_{ij} + \\ \alpha_2 \sum_{i\in N} \sum_{j\in N} \max(0, e_i - A_i) + \\ \alpha_3 \sum_{i\in N} \sum_{j\in N} \max(0, D_i - l_i) \end{cases} \tag{3.27}$$

Où a_1, a_2 et a_3 sont des coefficients de pondération et de mise à l'échelle.
Les contraintes de cette fonction sont assurées par les équations (3.2), (3.3), (3.4), (3.5), (3.6), (3.7), (3.8), (3.9), (3.10) et (3.11).

3.3.1.2 Le codage

Il existe plusieurs approches de codage des solutions pour les problèmes d'ordonnancement des systèmes de transport. Nous distinguons le codage par liste de permutation, le codage par liste de rang et le codage par matrice de permutation.
Notre problème consiste à ordonnancer le passage d'un véhicule de service de capacité limitée Q chez les clients et leurs fournisseurs. Ces derniers que nous appelons " noeuds " ou " sommet " ont chacun une position géographique (X, Y), une fenêtre de temps de service $[e_i,l_i]$, un temps de service si et une quantité à traiter q_i .
On sait intéresser principalement au codage par liste de permutation. Ce type de codage consiste à classer les noeuds dans l'ordre de passage du véhicule. Afin d'illustrer cette méthode nous allons considérer un ordonnancement qui consiste à séquencer les différents noeuds selon l'ordre suivant : nœud 2, nœud 4, nœud 1 et nœud 3.

Exemple 2. Une solution peut être codée comme suit avec une liste de permutation :

Position	$1^{ére}$	$2^{éme}$	$3^{éme}$	$4^{éme}$
nœud(i)	2	4	1	3

3.3.1.3 Génération de la population initiale des solutions initiales

Afin d'améliorer la qualité des solutions de la population initiale, nous avons opté pour la construction de celle-ci à l'aide d'heuristiques élaborées dans le but de minimiser la distance totale parcourue, la somme des temps d'attente et la somme des retards en tenant compte des différentes caractéristiques du problème.

A. Minimisation de la distance totale parcourue (heuristique 1)
Pour minimiser la distance totale parcourue par les individus de la population initiale, une heuristique a été élaborée. Le principe de cette heuristique est de fixer le nœud de départ (dépôt) puis de le faire suivre par celui qui lui est le plus proche et de finir par le retour au dépôt. Cette procédure est répétée jusqu'à l'obtention du nombre désiré d'individus à placer dans la population initiale.

Début
1 $S = \{\}$ (Initialiser la séquence à une séquence vide) .
2 Choisir le dépôt comme nœud de départ i_0 ; $S = S \cup \{i_0\}$.
3 **Répéter**
3.1 Trier les noeuds restants selon la distance euclidienne les séparant du dernier nœud séquencé.
3.2 Placer le nœud choisi à la suite de la séquence courante.
4 **Jusqu'à** noeuds restants = {}
5 Finir par le dépôt comme nœud d'arrivée .
Fin

Exemple 3. Soient les 4 noeuds A (dépôt), B, C et D avec les coordonnées suivantes : $A(5,5)$; $B(7,7)$; $C(10,10)$; $D(4,4)$.

En appliquant l'heuristique 1 l'individu construit est illustré dans le tableau suivant :

A	D	B	C	A

B. Minimisation de la somme des temps d'attente (heuristique 2)
Afin de faire en sorte que la génération des individus de la population initiale soit guidée, nous avons élaboré une heuristique qui utilise les caractéristiques temporelles du problème en vue de minimiser la somme des temps d'attente. Le principe de la deuxième heuristique est de partir du dépôt et d'y retourner en passant par le reste des noeuds triés dans l'ordre décroissant de la somme de la date de fin de fenêtre de temps et du temps de service de chacun simultanément ($l_i + s_i$).

Exemple 4. Soient les 3 noeuds B, C et D avec leur dates de fin de fenêtre de temps respectives 30, 40, 50 et leurs temps de service respectifs 10, 20, 25. Soit le nœud A le dépôt.

En appliquant l'heuristique 2 l'individu construit est présenté dans le tableau suivant :

A	D	C	B	A

C. Minimisation de la somme des retards (heuristique 3)
Afin de minimiser la somme des retards et de faire en sorte que la génération des individus de la population initiale soit logique, nous avons élaboré une heuristique qui utilise les caractéristiques temporelles du problème. Le principe de la troisième heuristique est de partir du dépôt et d'y retourner en passant par le reste des noeuds triés dans l'ordre croissant de la date de fin de fenêtre de temps.

Exemple 5. Soient les 3 noeuds B, C et D avec leurs dates de fin de fenêtre de temps respectives 30, 40, 50. Soit le nœud A le dépôt.

En appliquant l'heuristique 3 l'individu construit est présenté dans le tableau suivant :

A	B	C	D	A

3.3.1.4 Conception des solutions réalisables et correction des individus non fiables

Pour satisfaire les contraintes (3.6), (3.7), (3.8) de capacité du véhicule et (3.9) et (3.10) de précédences entre les nœuds, nous avons élaboré un moyen de correction des solutions non fiables. Ce moyen consiste à repérer les gènes mal placés et à rectifier leur position. Ceci nous permet d'avoir une population initiale totalement composée d'individus réalisables, et de corriger les individus non réalisables après l'étape de mutation. Ce moyen est composé de deux procédures de correction : la première concernant les contraintes de précédences et la deuxième les contraintes de capacité. En appliquant la première suivie de la deuxième nous obtenons des individus fiables.

A. Procédure de vérification et correction des individus vis-à-vis des contraintes de précédence

Le principe de cette procédure est de parcourir le chromosome gène par gène et de vérifier quand le nœud est un client, si son fournisseur respectif a été visité. Si ce n'est pas le cas, nous insérons ce fournisseur juste devant le client. Quand la correction est faite, la procédure se poursuit jusqu'à la fin du chromosome.

Début

1. i , j , a , k : entiers .
2. **Pour** i de 1 à $n-2$.
   (a) **Si** (le nœud $I[i]$ est un client (successeur de $I[i]=0$)).
      i. **Pour** j de i à $n-2$.
         A. **Si** le fournisseur de $I[i]$ est trouvé (prédécesseur de $I[i]=I[j]$).
            1. soit $a=I[j]$.
            2. Pour k de j à i .
               a. $I[k]=I[k-1]$.
               b. $k=k-1$.
               Fin Pour
            3. $I[i]=a$.
            Fin Si
         B. $j=j+1$.
         Fin Pour
      ii. $i=i+1$.
   Fin Si
Fin Pour
Fin

Exemple 6. Soient les couples de (fournisseur, client) suivants :

$(4, 2)$; $(6, 3)$; $(10, 1)$; $(9, 7)$ et $(5, 8)$. Le dépôt est le nœud 0.

Considérons l'individu :

0	4	6	2	9	1	3	5	7	10	8	0

L'algorithme permet de localiser le gène (10) mal placé et de l'insérer avant son client (1) pour donner l'individu viable suivant :

0	4	6	2	9	10	1	3	5	7	8	0

B. Procédure de vérification et correction des individus vis-à-vis des contraintes de capacité

Après avoir appliqué la procédure de correction de précédence, on applique cette procédure pour assurer la vérification et la correction vis à vis des contraintes de capacité.
Son principe est de parcourir le chromosome gène par gène et de calculer la quantité transportée. Si celle-ci excède la capacité du véhicule, la procédure revient au fournisseur précédent qui n'est pas suivi par son client puis elle le fait immédiatement suivre par ce dernier. Quand la correction est faite, la procédure se poursuit jusqu'à la fin du chromosome.

Début

1. $quant$, Q : réel .
2. i, a, j, k, f, $test$: entiers .
3. $quant = 0$.
4. Q = quantité maximale du véhicule.
5. **Pour** i de 1 à $n-2$.
   (a) $quant = quant$ + quantité du nœud $I[i]$.
   (b) **Si** $(quant > Q)$.
      i. **Pour** f de $i-1$ à 1 .
         A. **Si** (le nœud $I[f]$ est un fournisseur (prédécesseur de $I[f] = 0$)) .
            1. $test = 0$.
            2. Pour k de $f+1$ à i.
               (a) Si(le client de $I[f]$ est trouvé (successeur de $I[f] = I[k]$)).
                  i . $test = 1$.
                  Fin Si
               (b) $k = k+1$
               Fin Pour
            3. Si $(test = 0)$.
               (a) Pour j de $i+1$ à $n-1$.
                  i. **Si** (le client de $I[f]$ est trouvé (successeur de $I[f] = I[j]$))
                  A. $a = I[j]$.
                  B. **Pour** k de j à $f+1$.
                  1. $I[k] = I[k-1]$.
                  Fin Pour
                  C. $k = k-1$.
                  D. $I[f+1] = a$.
                  Fin Si
               Fin Pour
               (b) $j = j+1$.
               Fin Si
            Fin Si
         B. $f = f-1$.
         Fin Pour
      Fin Si
   (c) $i = i+1$;
Fin Pour
Fin

Exemple 7. Soient les couples de (fournisseur, client) suivants : $(4,2)$; $(6,3)$; $(10,1)$; $(9,7)$ et $(5,8)$.

Le dépôt est le nœud 0 ; $Q = 60$; $q[1] = -20$; $q[2] = -20$; $q[3] = -20$; $q[4] = 20$; $q[5] = 20$; $q[6] = 20$; $q[7] = -20$; $q[8] = -20$; $q[9] = 20$; $q[10] = 20$.

Considérons l'individu :

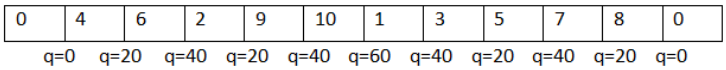

FIGURE 3.1 – Individu viable vis-à-vis des contraintes de capacité.

3.3.1.5 Sélection

La méthode de sélection élaborée est basée sur le principe de la roulette exposé au chapitre précédent. Dans cette méthode, les individus non dominés sont gardés afin de permettre l'obtention de différentes solutions Pareto optimales possibles qui constitueront un support de décision et garantiront aussi la diversité de l'évolution des individus.

3.3.1.6 Croisement

Afin d'assurer la viabilité des solutions et de décroître le temps total de calcul, nous avons conçu un opérateur de croisement qui ne nécessite pas de procédures de correction. Considérons p la position de croisement, l'algorithme de l'opérateur de croisement est le suivant :

Début
1 I_1, I_2 les individus parents et I_3, I_4 les individus enfants .
2 n : nombre des gènes : nombre des fournisseurs, clients et le ($dépôt^*2$) (départ et arrivée) .
3 Choisir au hasard une position de croisement p .
4 Fixer le dépôt comme nœud de départ et d'arrivée .
5 **Répéter** du second gène au gène de la position p le premier et le dernier gènes sont le dépôt .
  5.1 **Si** [(le gène de I_1 est un fournisseur $\cap \nexists$ dans I_3) $\cap$ ($Q \geq y_i + q_i$)]$\cup$[(le gène de I_1 est un client $\cap$ le fournisseur respectif $\exists$ dans I_3)] .
    5.1.1 Copier le gène de I_1 dans I_3 .
    5.1.2 Garder la position f_1 du dernier gène placé dans I_3 .
  5.2 **Si** [(le gène de I_2 est un fournisseur $\cap \nexists$ dans I_4) $\cap$ ($Q \geq y_i + q_i$)]$\cup$[(le gène de I_2 est un client $\cap$ le fournisseur respectif $\exists$ dans I_4)] .
    5.2.1 Copier le gène de I_2 dans I_4 .
    5.2.2 Garder la position f_2 du dernier gène placé dans I_4 .
    5.2.3 Aller au gène suivant .
6 **Tant que** ($f_1 < n$) .
7 **Répéter** du second gène de I_2 au gène $n-1$ ($-1 \rightarrow$ exclure le dépôt arrivée) .
  7.1 **Si**[(le gène de I_2 est un fournisseur $\cap \nexists$ dans I_3) $\cap$ ($Q \geq y_i + q_i$)]$\cup$[(le gène de I_2 est un client $\cap$ le fournisseur respectif $\exists$ dans I_3)] .
    7.1.1 Copier le gène de I_2 dans I_3 .
    7.1.2 Passer au nœud suivant de I_2 .
8 **Tant que** ($f_2 < n$) .
9 **Répéter** du second gène de I_1 au gène $n-1$ ($-1 \rightarrow$ exclure le dépôt arrivée) .
  9.1 **Si**[(le gène de I_1 est un fournisseur $\cap \nexists$ dans I_4) $\cap$ ($Q \geq y_i + q_i$)]$\cup$[(le gène de I_1 est un client $\cap$ le fournisseur respectif $\exists$ dans I_4)] .
    9.1.1 Copier le gène de I_1 dans I_4 .
    9.1.2 Passer au nœud suivant de I_1 .
Fin

Exemple 8. On a que $p = 5$;

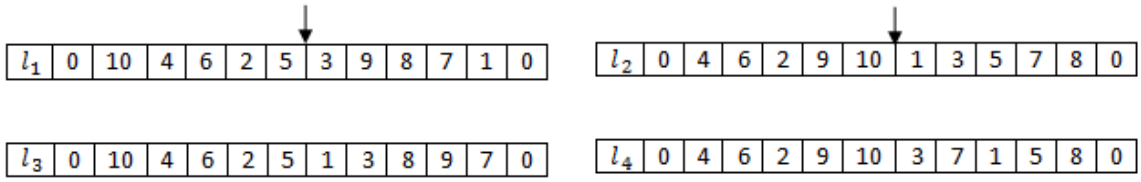

FIGURE 3.2 – Croisement.

Dans cet opérateur nous choisissons au hasard une position de croisement p. Après nous fixons le dépôt comme nœud de départ et d'arrivée et nous parcourons la solution (chromosome) gène par gène du dépôt de départ vers la position de croisement p. Si le sommet est un fournisseur respectant les contraintes de capacité nous le copions dans l'enfant correspondant. Si le sommet est un client et son fournisseur a été servi nous le copions aussi dans l'enfant correspondant.
Pour compléter le premier enfant du croisement, nous parcourons le deuxième parent gène par gène. Si le sommet est un fournisseur pas encore servi dans le premier enfant et son insertion dans ce dernier permet de vérifier la contrainte de capacité, alors nous le copions dans cet enfant. Si le sommet est un client dont le fournisseur est visité dans le premier enfant alors nous le copions dans ce dernier. Sinon nous répétons cette procédure jusqu'à compléter le premier enfant. Puis nous appliquons cette procédure pour générer le deuxième enfant en utilisant ses deux parents.

3.3.1.7 Mutation

Pour ce problème, plusieurs opérateurs de mutation ont été développées, nous nous sommes intéressés principalement au premier opérateur existant dans notre étude. C'est un opérateur contrôlé basé sur la condition d'optimalité locale. Son but est de minimiser la distance totale parcourue. Son principe est de faire la somme des distances parcourues de trois noeuds successifs en parcourant l'individu, puis de permuter les gènes centraux des deux groupes ayant les plus grandes sommes de distances parcourues.

Début
  Pour i de 1 à $n-2$;
    1 Faire la somme des distances entre le gène i et ses deux voisins ;
    2 Permuter les gènes centraux des deux groupes ayant les plus grandes sommes de distances ;
  Fin Pour
Fin

A l'aide de cet opérateur nous augmentons les chances d'amélioration de l'individu après une opération de mutation.

3.3.1.8 Méthode d'évaluation multicritère de la $1^{ère}$ approche évolutionniste

La stratégie d'évaluation de la première approche est basée sur l'utilisation de la méthode de la Pareto Optimalité.
Dans cette application, puisque nous minimisons F_1, on dit qu'une solution sol_1 domine une autre sol_2 si et seulement si $f_{11} \leq f_{12}$, $f_{21} \leq f_{22}$ et $f_{31} \leq f_{32}$. Et on dit aussi que deux solutions sont non dominées l'une par rapport à l'autre si est seulement si elles vérifient une des

conditions du systèmes suivant :

$$\begin{cases} f_{11} > f_{12}, f_{21} \leq f_{22}, f_{31} \leq f_{32}. \\ f_{11} \leq f_{12}, f_{21} > f_{22}, f_{31} \leq f_{32}. \\ f_{11} \leq f_{12}, f_{21} \leq f_{22}, f_{31} > f_{32}. \\ f_{11} > f_{12}, f_{21} > f_{22}, f_{31} \leq f_{32}. \\ f_{11} > f_{12}, f_{21} \leq f_{22}, f_{31} > f_{32}. \\ f_{11} \leq f_{12}, f_{21} > f_{22}, f_{31} > f_{32} \end{cases} \tag{3.28}$$

Avec :

$$f_1 = \alpha_1 \sum_{i \in \mathcal{N}} \sum_{j \in \mathcal{N}} d_{ij} X_{ij}, \tag{3.29}$$

$$f_2 = \alpha_2 \sum_{i \in \mathcal{N}} \sum_{j \in \mathcal{N}} \max(0, e_i - A_i), \tag{3.30}$$

$$f_3 = \alpha_3 \sum_{i \in \mathcal{N}} \sum_{j \in \mathcal{N}} \max(0, l_i - D_i), \tag{3.31}$$

f_{11}, f_{21} et f_{31}les valeurs respectives de f_1 , f_2 et f_3 de la solution sol_1.
f_{12} , f_{22} et f_{32}les valeurs respectives de f_1 , f_2 et f_3 de la solution sol_2.

3.3.1.9 Structure de l'algorithme évolutionniste de la $1^{ère}$ approche

L'algorithme suivant représente la structure évolutionniste qu'on a implémentée pour la première approche de résolution du 1-PDPTW qui a été inspirée par la référence [21]

Début
1 Créer, corriger et évaluer, la population initiale (de taille constante N).
2 Croiser la population initiales afin d'avoir plus d'individus
3 **Tant que** le critère d'arrêt n'est pas satisfait **faire**
  3.1 Lancer la roulette.
  3.2 Selon la probabilité obtenue, remplir le reste de la population inter -médiaire avec des enfants qui sont le fruit de croisements, de mutations ou de copie .
Fin Tant que
Copier les solutions non dominées dans la population suivante.
Fin

3.3.2 Deuxième approches : colonies de fourmis

3.3.2.1 Les propriétés utilisées dans l'adaptation proposée

Dans notre travail nous avons modifié l'algorithme de colonies de fourmis pour l'adapter aux defferentes fonctions objectives et aux contraintes de notre problème qui est le PDPTW a un véhicule.
Avec les changements suivants :
Les objectifs a minimiser sont :

- la distance du trajet (f_1).
- Les temps d'attentes (f_2).
- Les retard (f_3).

L'optimisation par colonies de fourmis s'inspire du comportement des fourmis lorsque celles ci sont à la recherche de nourriture. Les fourmis ont la particularité d'employer pour communiquer des substances volatiles appelées phéromones. Elles sont attirées par ces substances, qu'elles perçoivent grâce à des récepteurs situés dans leurs antennes. Les fourmis utilisent les pistes de phéromone pour marquer leur trajet.
Les traces de phéromone représentent le moyen qu'utilise une fourmi pour attirer les autres fourmis de sa colonie vers les aires correspondantes.[32]

3.3.2.1.1 Structures de phéromone.

Dans notre modèle on a utilisé une seule structure de phéromone comme proposé dans [11]. Dans ce cas, les traces de phéromone déposées par une fourmi sont définies relativement à une agrégation des objectifs à optimiser.
A chaque étape de la construction de solution d'une fourmi, un candidat est choisi relativement à une probabilité de transition qui dépend de deux facteurs : un facteur phéromone et un facteur heuristique.

3.3.2.2 Formulation mathématique adaptée

La métaheuristique a été inspirée par les études sur les comportement des fourmis réelles.À l'origine, l'optimisation par colonie de fourmis a été conçue pour résoudre le (pvc) problème du Voyageur de Commerce dont l'objectif de cette méthode est de trouver une tournée la plus courte pour un ensemble de villes donnée.
Avant de décrire en détail les étapes de notre algorithme, nous allons donné les notations que l'on vas utiliser dans la suite.

3.3.2.2.1 Constantes

- f : Nombre de fourmis.
- $distance_{ij}$: Distance entre le nœud i et j.
- $temps_{ij}$: Temps mis par le véhicule pour aller de i vers j.
- Q : Constantes.
- ρ : Taux de persistance des traces de phéromone (coefficient qui définit la vitesse d'évaporation des phéromones sur les nœud entre l'instant t et $t+1$).
- e_i : Début de la fenêtre de temps du nœud i.
- l_i : Fin de la fenêtre de temps du nœud i.
- s_i : Temps de service du nœud i.
- q_i : Quantité à traiter au nœud i.
- n : Ensemble des fournisseur.
- m : Ensemble des clients.

3.3.2.2.2 Variables

- τ_{ij} : Quantité de phéromones sur le chemin du nœud i au nœud j.

$$\tau_{ij}(t+1) = (1-\rho)\tau_{ij}(t) + \sum_{k=1}^{n} \Delta\tau_{ij}^{k}(t) \tag{3.32}$$

- $\Delta\tau_{ij}^{k}$: Ajout de phéromone par la $k^{éme}$ fourmi du nœud i au nœud j.

$$\Delta\tau_{ij}^{k}(t) = \begin{cases} \frac{Q}{l^{k}(t)} & Si(i,j) \in j^{k}(i). \\ 0 & Si(i,j) \notin j^{k}(i). \end{cases} \tag{3.33}$$

- l^{k} : Cout total de la solution trouvée par la fourmi k.
- J_{i}^{k} : Représente les nœud condidat qui vérifier la contrainte de capacité ainsi que celle de la précédence.
- $A[i]$: Représente l'arrivés au nœud i.
- $D[i]$: Représente le départ du nœud i.
- α : Importance relative du taux de phéromone.
- β : Importance relative du temps de trajet, le temps d'attente et le retard.
- P_{ij}^{k} : La probabilité de déplacement du nœud i au nœud j par la $k^{éme}$ fourmi.

$$P_{ij}^{k}(t) = \begin{cases} \frac{(\tau_{ij}(t))^{\alpha}((\frac{1}{temps[i][j]+Attente[i][j]+Retard[i][j]})^{\beta})}{\sum_{l\in J_{i}^{k}}((\tau_{il}(t))^{\alpha})(\frac{1}{temps[i][l]+Attente[i][l]+Retard[i][l]})^{\beta})} & Si \;\; j \in J_{i}^{k}. \\ 0 & Si \;\; j \notin J_{i}^{k}. \end{cases} \tag{3.34}$$

3.3.2.3 Fonctionnement de l'algorithme :

1. Initialisation
   Initialiser $t = 0$
   NB :entier représentant le nombre d'itération qu'on vas faire
   Initialiser l'intensité de la trace à $\tau_{ij}^0 = \tau_0 = \text{constante} \neq 0$
2. Commande
   (a) Pour H allant de 1 à NB faire
   Toutes les K fourmis commencent leurs trajet au nœud 0 qui représente le dépôt ; Après elles vont chaqu'une vers un nœud fournisseurs ainsi avec le dépôt et le nœud fournisseurs commence la liste qui modélise leurs mémoires Pour chaque fourmi k
      i. Pour j allant de 1 à n faire
         A. i représente le nœud choisi précédemment.
         B. Si j vérifie les deux contraintes de précédences ainsi que celle de la capacité.
         C. Choisir le prochain nœud j tel que $j \in J_i^k$.
         D. Ou j est choisi avec la probabilité.

$$P_{ij}^k(t) = \begin{cases} \frac{(\tau_{ij}(t))^\alpha((\frac{1}{temps[i][j]+Attente[i][j]+Retard[i][j]}^\beta))}{\sum_{l\in J_i^k}((\tau_{il}(t))^\alpha)(\frac{1}{temps[i][l]+Attente[i][l]+Retard[i][l]}^\beta)} & Sij \in J_i^k. \\ 0 & Sij \notin J_i^k. \end{cases}$$

         E. le j choisi est le j qui vérifie

$$j = Argmax(\tau_{ij}(t))^\alpha(\frac{1}{temps[i][j] + Attente[i][j] + Retard[i][j]}^\beta)$$

      ii. Enregistrer toutes les solutions trouver dans une liste tabou
         A. Pour k allant de 1 a K faire
            – Calculer $L_k(t)$
            – Calculer $\Delta\ \tau_{ij}^k(t)$
            – Mettre a jour le taux de phéromone :

$$\tau_{ij}(t+1) = (1-\rho) * \tau_{ij}(t) + \Delta\tau_{ij}(t).$$

            – Avec que :

$$\Delta\tau_{ij}(t) = \sum_{k=1}^{k=K} (\Delta\tau_{ij}^k(t))$$

   (b) $t = t + 1$;
   (c) H=H+1 ;
3. Prendre les solutions enregistrer dans la la liste tabou
4. Faire une sélection des solutions non dominée et les afficher

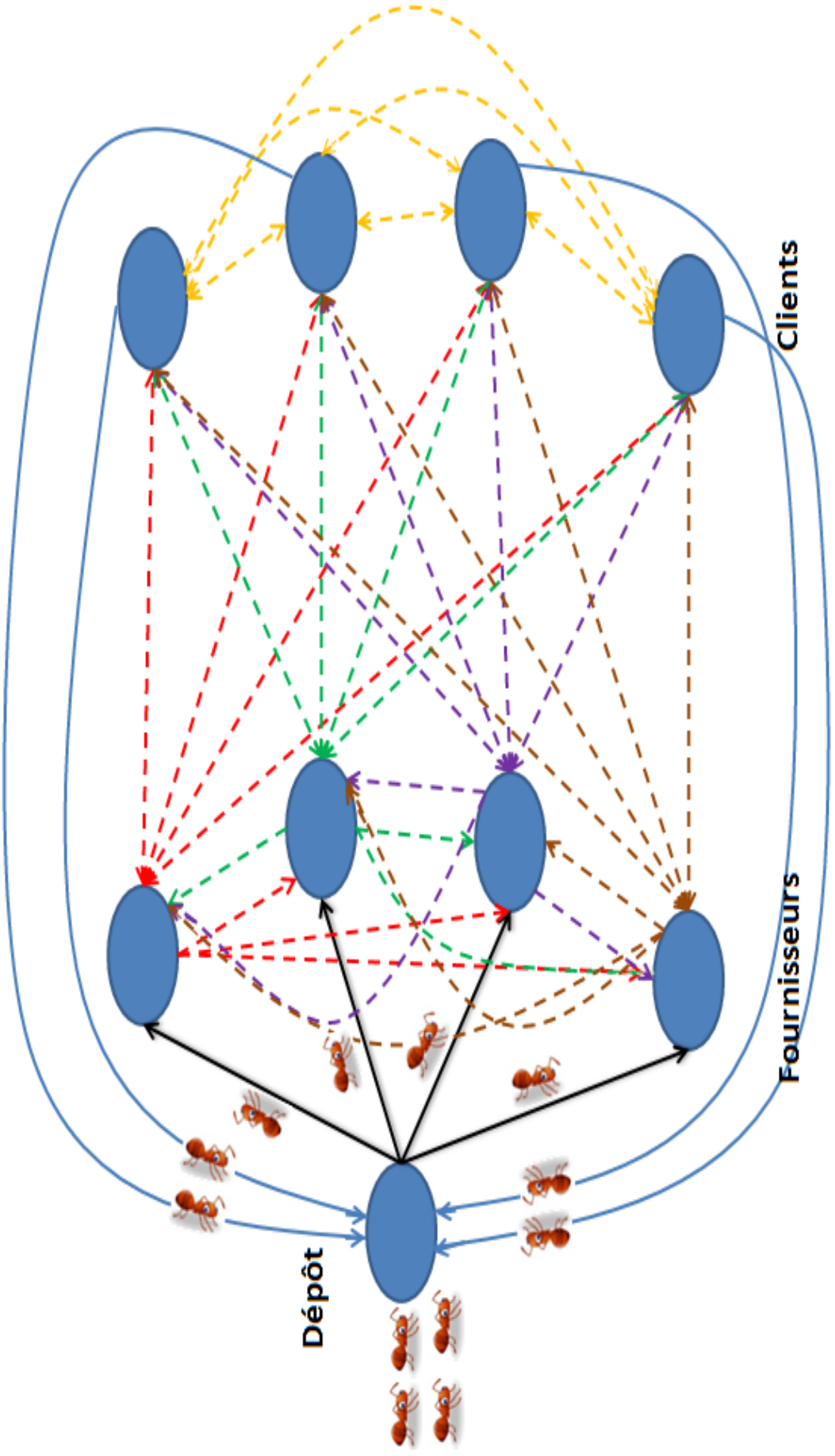

FIGURE 3.3 – Exemple de colonie de fourmis

3.4 Cas pratique

Dans notre cas pratique nous avons pris des villes de la wilaya de Bejaia comme fournisseurs et clients et Bejaia ville comme le dépôt (nœud 0) avec les distances et les temps associés (que nous avons récupéré dans Google map).
Les villes choisies sont les suivantes :
Bejaia(0), Sidi-Ayad(1), Sidi-Aich(2), Timezrit(3), Leflay(4), Chemini(5), Tifra (6), Hemmam-Sillale(7), Fenaia(8), Seddouk(9), M'cisna(10), El-kseur(11), Ibourassen-Oued-Ghir (12), Toudja(13), Tala-Hamza(14).
Les fournisseurs et leurs client respective sont :

Fournisseurs	1	2	3	6	10	12	11
Clients	4	7	5	8	9	13	14

Les distance entres les différentes villes sont données dans le tableau suivant :

Distance /km	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
0	0	42.7	45.6	38	50.9	55.8	41.7	47.1	33.6	54.6	53.4	23	9.6	22.2	9.3
1	41.7	0	3.8	7.4	9.7	18.7	17.8	14.2	17.4	14	12.5	19.1	34.7	43	36.3
2	41.6	3.8	0	10.8	5.9	14.9	14	10.5	13.6	13.7	12.4	19	34.5	42.8	39.3
3	37.3	7.4	10.8	0	21.7	30.7	16.6	19.9	12.2	29.5	22.5	14.7	30.2	38.5	31.6
4	46.8	10.6	5.8	21.7	0	11.7	15.7	12.1	18.9	13.2	16.4	24.2	39.7	48	46.8
5	57	20.8	16	31.9	11.7	0	22.5	18.9	29.1	18.6	21.9	34.4	49.9	58.2	57
6	41.6	18.8	14	16.5	15.7	22.5	0	3.6	13.7	29.7	25.9	19.1	34.6	43.1	41.7
7	45	15.2	10.4	19.9	12.1	22.5	3.6	0	21.9	23.4	22.3	25.1	37.9	46.2	45
8	33.6	16.3	13.2	11.9	18.5	27.6	13.4	21.9	0	26.3	22.3	11.5	26.6	34.9	33.7
9	54.6	18.5	13.7	29.5	13.2	18.6	27.1	23.5	26.7	0	6.15	32	45.5	55.8	54.6
10	53.4	12.5	12.4	21.4	16.4	21.9	30.3	26.7	29.9	6.5	0	35.3	47.5	55.8	54.6
11	23.3	19.2	18.9	14.8	24.2	33.2	19.1	25.1	11.5	32	35.3	0	16.2	24.6	21
12	11.7	33.8	33.4	29.3	38.7	47.7	33.6	39	25.5	46.5	49.8	14.9	0	14.1	11.8
13	23.1	43.3	43	38.9	48.3	57.3	43.1	48.6	35	56.1	54.9	24.4	16	0	23.1
14	9.3	36.2	41.9	31.6	47.2	56.2	42.1	47.6	34	55	51.1	23.4	9	23	0

Temps/min	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
0	0	73	70	63	79	102	74	79	64	93	114	47	30	57	22
1	64	0	12	20	27	48	41	34	38	36	34	37	56	72	61
2	56	13	0	30	14	35	28	21	26	27	34	29	48	64	58
3	54	20	30	0	41	62	33	39	27	53	47	27	46	62	50
4	65	27	15	42	0	31	34	27	36	28	39	39	57	48	67
5	82	47	35	61	31	0	51	44	55	37	48	57	75	91	85
6	57	40	29	34	33	50	0	8	28	51	61	31	49	64	59
7	62	33	21	39	26	50	7	0	31	46	53	36	54	70	64
8	46	37	25	28	35	57	26	31	0	48	53	19	39	34.9	48
9	75	37	25	51	28	39	50	43	45	0	12	46	64	81	74
10	82	32	32	47	37	48	60	53	54	12	0	55	65	81	74
11	30	39	32	30	41	62	33	36	20	52	63	0	22	38	32
12	19	55	46	46	54	74	48	52	36	64	75	19	0	27	20
13	38	71	61	62	69	89	64	70	54	79	93	36	31	0	40
14	20	63	57	53	65	75	60	66	49	75	87	32	16	40	0

Les fenêtres de temps, le temps de service, ainsi que la quantité a traités à chaque noeuds et que l'on a associer à notre modèle sont donnée comme suit :

propriétés i	e_i	l_i	s_i	q_i
1	100	250	08	20
2	120	200	14	20
3	280	380	20	20
4	300	400	06	-20
5	450	500	15	-20
6	800	1120	06	20
7	600	800	04	-20
8	100	840	15	-20
9	800	1200	15	-20
10	600	1000	12	20
11	600	1200	14	20
12	600	1200	23	20
13	800	1200	55	-20
14	600	1200	20	-20

3.4.1 Résolution avec l'algorithme génétique

Nous allons énumérer dans le tableau suivant quelque solutions récupéré par l'implémentation des donnée ,étant donnée que le nombre de solutions générer est grand :

num-solu	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	f_1	f_2	f_3
1 :	0	11	14	12	13	6	8	3	1	2	4	5	10	9	7	0	261	621	4443
2 :	0	12	13	11	14	10	9	6	8	2	7	3	5	1	4	0	319	720	5948
3 :	0	2	1	3	4	5	7	6	8	10	9	11	12	13	14	0	249	507	9
4 :	0	11	14	12	13	6	8	10	9	2	7	3	5	1	4	0	304	621	4960
5 :	0	12	13	11	14	10	9	6	8	3	1	2	4	5	7	0	272	720	5711
6 :	0	11	14	12	13	6	8	2	1	3	4	5	7	10	9	0	283	621	4561
7 :	0	2	1	3	4	5	7	11	14	12	13	6	8	10	9	0	318	346	161
8 :	0	12	13	11	14	10	9	6	2	1	4	7	8	3	5	0	314	720	5705
9 :	0	2	1	3	4	5	7	6	12	13	11	14	10	9	8	0	324	507	569
10	0	2	1	3	4	5	7	11	14	12	10	9	6	8	13	0	319	318	118
11	0	11	1	3	4	5	2	7	6	8	10	9	12	13	14	0	247	553	2214
12	0	2	1	3	4	5	7	11	6	8	10	12	9	13	14	0	371	431	103
13	0	11	1	3	4	5	2	7	14	12	13	10	9	6	8	0	313	553	2873
14	0	2	1	3	4	12	5	7	6	8	10	9	11	13	14	0	319	384	221
15	0	11	1	3	4	2	5	7	6	8	10	9	12	13	14	0	249	553	2202
16	0	2	1	3	4	5	7	11	6	8	10	12	9	13	14	0	371	431	103
17	0	11	1	3	4	5	2	7	14	12	10	9	6	8	13	0	318	553	2661
18	0	2	1	3	4	12	5	7	6	8	10	9	11	13	14	0	319	384	221
19	0	11	1	3	4	2	5	12	13	14	10	9	6	8	7	0	344	553	3258
20	0	2	1	3	4	5	7	6	8	11	10	9	12	13	14	0	271	507	22
21	0	2	1	3	4	5	7	11	6	8	10	9	12	13	14	0	287	431	9
22	0	2	1	3	4	5	7	11	14	12	10	9	6	8	13	0	319	318	118
23	0	2	1	3	4	5	7	11	6	8	14	12	10	9	13	0	344	431	58
24	0	2	1	3	4	5	7	11	12	6	8	10	9	14	13	0	327	371	9
25	0	2	1	3	4	5	7	11	12	6	8	10	14	9	13	0	408	371	43
26	0	2	1	3	4	5	7	11	14	12	13	10	9	6	8	0	314	346	246
27	0	2	1	3	4	5	7	6	8	10	11	14	12	9	13	0	339	507	98
28	0	2	1	3	4	5	7	11	14	12	6	8	10	13	9	0	400	325	9
29	0	2	1	3	4	5	7	11	14	12	6	8	10	9	13	0	319	325	9
30	0	11	1	3	4	5	2	7	6	8	14	12	10	9	13	0	304	553	2368
31	0	2	1	3	4	5	12	7	6	8	10	9	11	13	14	0	314	401	9
32	0	11	1	3	4	5	2	7	14	12	6	8	10	13	9	0	399	553	2758

Aprés avoir généré 45 générations, les solutions que nous avons récupérer sont les suivantes :

0	2	1	3	4	5	7	6	8	10	9	11	12	13	14	0	249	507	9
0	2	1	3	4	5	7	11	14	12	13	6	8	10	9	0	318	346	161
0	2	1	3	4	5	7	11	14	12	10	9	6	8	13	0	319	318	118
0	11	1	3	4	5	2	7	6	8	10	9	12	13	14	0	247	553	2214
0	2	1	3	4	5	7	11	6	8	10	9	12	13	14	0	287	431	9
0	2	1	3	4	5	7	11	14	12	13	10	9	6	8	0	314	346	246
0	2	1	3	4	5	7	11	14	12	6	8	10	9	13	0	319	325	9
0	2	1	3	4	5	12	7	6	8	10	9	11	13	14	0	314	401	9

3.4.2 Résolution avec colonies de fourmis

On a initialiser $t = 0$ et $\tau_{ij} = 2$ et après avoir essayer plusieurs combinaison pour alpha et beta on est arrivées à prendre $alpha = 0.2$ et $beta = 0.8$ et le taux d'évaporation a 0.5
Pour la première itération :
les tournées trouvées avec la loi de déplacement en respectant la contraintes de capacités et de précédences sont comme suit :

0	1	2	3	4	5	7	11	12	14	6	8	13	10	9	0
0	2	1	3	4	5	7	11	12	14	6	8	13	10	9	0
0	3	1	4	5	10	11	12	14	13	6	9	8	2	7	0
0	6	1	10	9	11	12	14	13	8	4	3	5	2	7	0
0	10	1	11	12	14	13	6	2	9	8	7	4	3	5	0
0	12	1	10	2	7	9	11	14	13	6	8	4	3	5	0
0	11	1	10	2	7	14	12	3	9	6	13	8	5	4	0

Après avoir finis les tournées, on calcule $L_k(t)$ et Δ_{ij}^k On mets a jours le taux de phéromone :
la matrice des nouveaux taux de phéromones est donné dans la figure suivante :

$$\begin{pmatrix}
0.000 & 1.000 & 1.000 & 1.000 & 1.000 & 1.000 & 2.476 & 1.000 & 1.000 & 1.000 & 2.973 & 3.095 & 3.036 & 1.000 & 1.000 \\
1.000 & 0.000 & 1.000 & 1.000 & 1.000 & 1.000 & 1.000 & 1.000 & 1.000 & 1.000 & 6.608 & 2.973 & 1.000 & 1.000 & 1.000 \\
1.000 & 1.000 & 0.000 & 1.000 & 1.000 & 1.000 & 1.000 & 6.608 & 1.000 & 2.973 & 1.000 & 1.000 & 1.000 & 1.000 & 1.000 \\
1.000 & 1.000 & 1.000 & 0.000 & 1.000 & 6.486 & 1.000 & 1.000 & 1.000 & 3.095 & 1.000 & 1.000 & 1.000 & 1.000 & 1.000 \\
3.095 & 1.000 & 1.000 & 6.486 & 0.000 & 1.000 & 1.000 & 1.000 & 1.000 & 1.000 & 1.000 & 1.000 & 1.000 & 1.000 & 1.000 \\
5.010 & 1.000 & 2.476 & 1.000 & 3.095 & 0.000 & 1.000 & 1.000 & 1.000 & 1.000 & 1.000 & 1.000 & 1.000 & 1.000 & 1.000 \\
1.000 & 2.476 & 2.973 & 1.000 & 1.000 & 1.000 & 0.000 & 1.000 & 3.036 & 1.000 & 1.000 & 1.000 & 1.000 & 3.095 & 1.000 \\
2.476 & 1.000 & 1.000 & 1.000 & 2.973 & 1.000 & 1.000 & 0.000 & 1.000 & 3.036 & 1.000 & 1.000 & 1.000 & 1.000 & 3.095 \\
1.000 & 1.000 & 1.000 & 1.000 & 4.513 & 3.095 & 1.000 & 2.973 & 0.000 & 1.000 & 1.000 & 1.000 & 1.000 & 1.000 & 1.000 \\
1.000 & 1.000 & 1.000 & 1.000 & 1.000 & 1.000 & 3.095 & 1.000 & 2.973 & 0.000 & 1.000 & 4.513 & 1.000 & 1.000 & 1.000 \\
1.000 & 2.973 & 5.131 & 1.000 & 1.000 & 1.000 & 1.000 & 1.000 & 1.000 & 2.476 & 0.000 & 1.000 & 1.000 & 1.000 & 1.000 \\
1.000 & 3.095 & 1.000 & 1.000 & 1.000 & 1.000 & 1.000 & 1.000 & 1.000 & 1.000 & 1.000 & 0.000 & 4.450 & 1.000 & 3.036 \\
1.000 & 3.036 & 1.000 & 3.095 & 1.000 & 1.000 & 1.000 & 1.000 & 1.000 & 1.000 & 1.000 & 1.000 & 0.000 & 1.000 & 4.450 \\
1.000 & 1.000 & 1.000 & 1.000 & 1.000 & 1.000 & 5.010 & 1.000 & 4.571 & 1.000 & 1.000 & 1.000 & 1.000 & 0.000 & 1.000 \\
1.000 & 1.000 & 1.000 & 1.000 & 1.000 & 1.000 & 1.000 & 1.000 & 1.000 & 1.000 & 1.000 & 1.000 & 3.095 & 6.486 & 0.000
\end{pmatrix}$$

Après avoir actualiser le taux de phéromone, les fourmis refond leurs parcours en prenant en considération le nouveau taux de phéromone donc les nouvelles solutions sont de la forme suivantes :

0	1	10	11	2	7	14	12	3	9	6	13	8	4	5	0
0	2	7	11	12	14	13	6	10	9	8	3	5	1	4	0
0	3	5	10	11	12	14	13	6	9	8	1	4	2	7	0
0	6	8	11	12	14	13	10	9	3	5	2	7	1	4	0
0	10	2	7	11	12	14	13	6	9	8	3	5	1	4	0
0	12	3	11	14	13	6	10	2	9	8	7	5	1	4	0
0	11	12	14	13	6	10	9	8	3	5	2	7	1	4	0

On refait la même procédure et on recalcule $L_k(t)$ et Δ_{ij}^k et on met a jours le taux de phéromone
Les nouvelles solutions trouver dans la nouvelle itération sont :

0	1	10	11	2	7	14	12	3	9	6	13	8	4	5	0
0	2	7	11	12	14	13	6	10	9	8	3	5	1	4	0
0	3	5	10	11	12	14	13	6	9	8	1	4	2	7	0
0	6	8	11	12	14	13	10	9	3	5	2	7	1	4	0
0	10	2	7	11	12	14	13	6	9	8	3	5	1	4	0
0	12	3	11	14	13	6	10	2	9	8	7	5	1	4	0
0	11	12	14	13	6	10	9	8	3	5	2	7	1	4	0

Les nouveaux taux de phéromones sont :

$$\begin{pmatrix}
0.000 & 0.500 & 0.500 & 0.500 & 0.500 & 0.500 & 2.652 & 0.500 & 0.500 & 0.500 & 3.464 & 3.156 & 3.293 & 0.500 & 0.500 \\
0.500 & 0. & 0.500 & 0.500 & 7.276 & 0.500 & 0.500 & 0.500 & 0.500 & 0.500 & 3.304 & 1.486 & 0.500 & 0.500 & 0.500 \\
0.500 & 0.500 & 0.000 & 0.500 & 0.500 & 0.500 & 0.500 & 8.305 & 0.500 & 3.261715 & 0.500 & 0.500 & 0.500 & 0.500 & 0.500 \\
0.500 & 0.500 & 0.500 & 0.000 & 0.500 & 8.244 & 0.500 & 0.500 & 0.500 & 1.547 & 0.500 & 2.274 & 0.500 & 0.500 & 0.500 \\
8.323 & 0.500 & 0.500 & 3.243 & 0.000 & 0.500 & 0.500 & 0.500 & 0.500 & 0.500 & 0.500 & 0.500 & 0.500 & 0.500 & 0.500 \\
2.505 & 4.252786 & 4.261 & 0.500 & 1.547 & 0.000 & 0.500 & 0.500 & 0.500 & 0.500 & 0.500 & 0.500 & 0.500 & 0.500 & 0.500 \\
0.500 & 1.238 & 1.486 & 0.500 & 0.500 & 0.500 & 0.000 & 0.500 & 2.932 & 2.477848 & 3.883 & 0.500 & 0.500 & 1.547 & 0.500 \\
1.238334 & 3.523438 & 0.500 & 0.500 & 1.486 & 2.274 & 0.500 & 0.000 & 0.500 & 1.518 & 0.500 & 2.477 & 0.500 & 0.500 & 1.547 \\
0.500 & 0.500 & 0.500 & 4.086 & 2.256 & 1.547 & 0.500 & 3.261 & 0.000 & 0.500 & 0.500 & 1.914 & 0.500 & 0.500 & 0.500 \\
0.500 & 0.500 & 0.500 & 1.914 & 0.500 & 0.500 & 1.547 & 0.500 & 6.848 & 0.000 & 0.500 & 2.256 & 0.500 & 0.500 & 0.500 \\
0.500 & 1.486 & 6.318 & 0.500 & 0.500 & 0.500 & 0.500 & 0.500 & 0.500 & 4.261 & 0.000 & 0.500 & 0.500 & 0.500 & 0.500 \\
0.500 & 1.547 & 0.500 & 0.500 & 0.500 & 0.500 & 0.500 & 0.500 & 0.500 & 0.500 & 0.500 & 0.000 & 7.226 & 0.500 & 3.293 \\
0.500 & 1.518 & 0.500 & 3.322 & 0.500 & 0.500 & 0.500 & 0.500 & 0.500 & 0.500 & 0.500 & 0.500 & 0.000 & 0.500 & 7.226 \\
0.500 & 0.500 & 0.500 & 0.500 & 0.500 & 0.500 & 7.866 & 0.500 & 2.285 & 0.500 & 1.914 & 0.500 & 0.500 & 0.000 & 0.500 \\
0.500 & 0.500 & 0.500 & 0.500 & 0.500 & 0.500 & 0.500 & 0.500 & 0.500 & 0.500 & 0.500 & 0.500 & 1.547 & 10.019 & 0.000
\end{pmatrix}$$

Après un nombre d'itération prédéfinis on a aboutis au solutions efficaces suivantes :

0	1	2	3	4	5	7	11	12	14	6	8	13	10	9	0	369	307	52
0	11	12	14	13	6	10	9	8	3	5	2	7	1	4	0	317	614	5284
0	1	4	3	5	10	11	12	14	13	6	9	8	2	7	0	381	299	1434
0	11	12	14	13	6	10	9	8	3	5	1	4	2	7	0	310	614	5300
0	1	4	2	3	5	7	11	12	14	13	6	10	9	8	0	330	307	385
0	3	5	1	4	2	7	11	12	14	13	6	10	9	8	0	327	323	1050

3.4.3 Classement des solutions efficaces par la méthode PROMÉTHÉE

En combinant les solutions trouvées par les deux approches nous allons classer ces solutions avec la méthode d'aide à la décision PROMETHEE 2 [9].
On commence par introduire le vecteur comprenant les poids des critères et le second vecteur comprennent les qualifications d'une action aux yeux de chaque critère.
Voir la figure 3.4 pour les paramètres de la méthode :

	Scénario1	Distance	Attente	Retard
	Unité	km	minute	minute
	Cluster/Groupe	◆	◆	◆
	Préférences			
	Min/Max	min	min	min
	Poids	0,30	0,25	0,45
	Fn. de préférence	Linéaire	Linéaire	Linéaire
	Seuils	absolu	absolu	absolu
	- Q: Indifférence	50,00	80,00	60,00
	- P: Préférence	80,00	150,00	120,00
	- S: Gaussien	n/d	n/d	n/d

FIGURE 3.4 – Les choix des paramètres pour les 3 critères sous Visual PROMETHEE Academic

Pour modéliser les préférences du décideur sur les 3 critères considérés. Nous associons à chacun d'entre eux un critère généralisé de Type linéaire. La matrice de performance est donnée dans la figure 3.5 :

	Evaluations			
☑	action1	369,00	307,00	52,00
☑	action2	317,00	614,00	5284,00
☑	action3	381,00	299,00	1434,00
☑	action4	310,00	614,00	5300,00
☑	action5	330,00	307,00	385,00
☑	action6	249,00	507,00	9,00
☑	action7	318,00	346,00	161,00
☑	action8	319,00	318,00	118,00
☑	action9	247,00	553,00	2214,00
☑	action10	287,00	431,00	9,00
☑	action11	314,00	346,00	246,00
☑	action12	319,00	325,00	9,00
☑	action13	327,00	323,00	1050,00
☑	action14	314,00	410,00	9,00

FIGURE 3.5 – les évaluations des differentes solutions

Le Classement des solutions est donné dans la figure 3.7 :

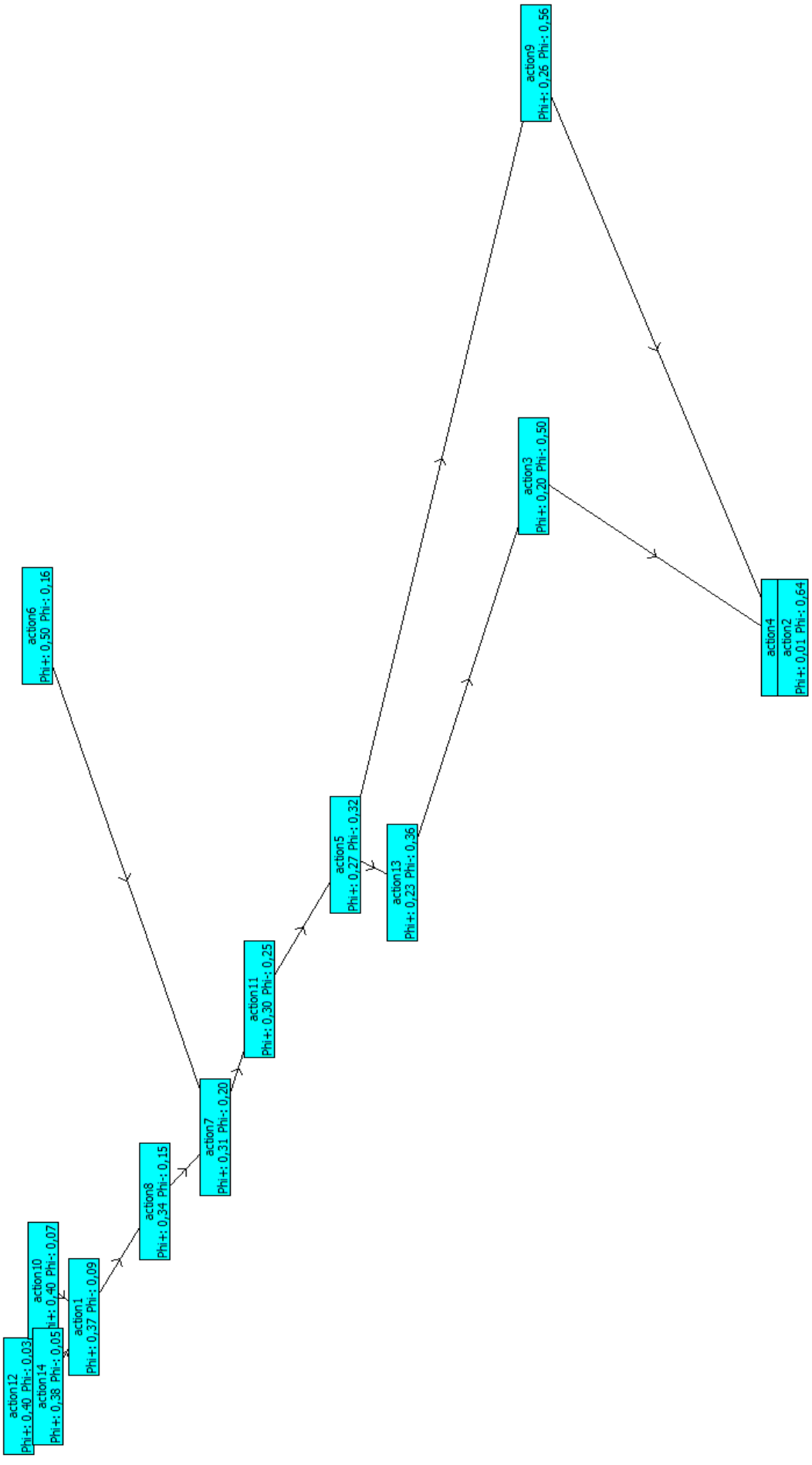

FIGURE 3.6 – Le graphe de classement des solutions d'après PROMÉTHÉE

La solution efficace qui correspond le mieux aux préférences considérées est la suivante :

0 2 1 3 4 5 7 11 14 12 6 8 10 9 13 0

La succession de villes que doit suivre le véhicule est présenter dans la figure suivante :

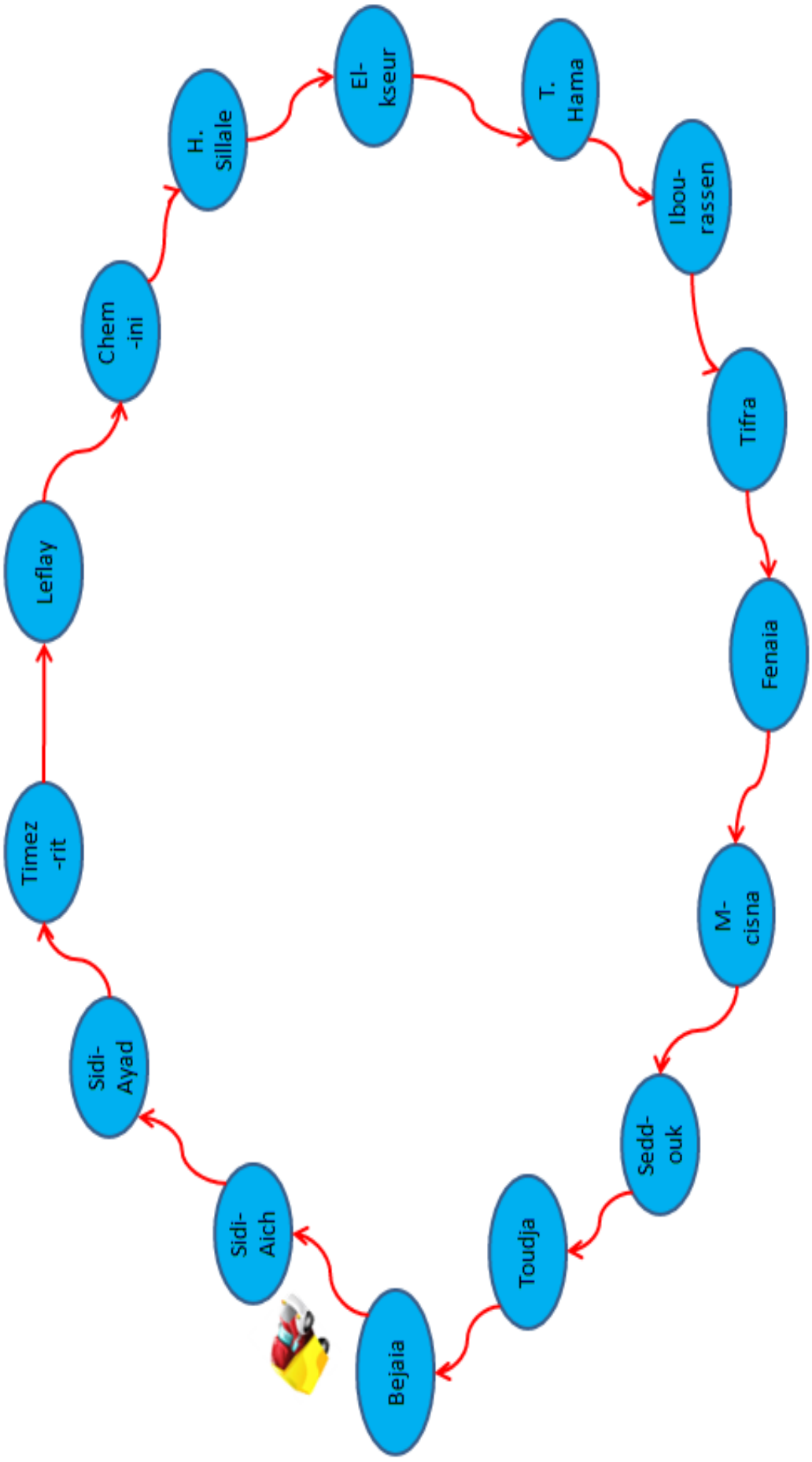

FIGURE 3.7 – Le trajet du véhicule associer à la solution efficace

Conclusion

Au niveau de ce chapitre, nous avons présenté en premier lieu le PDPTW et sa formulation mathématique. Ensuite nous avons introduit les travaux de Kammarti [21]et nous avons adaptés la méthode de colonies de fourmis au probléme $1-$PDPTW ; Enfin, nous avons programmé les deux méthodes et présenter un exemple d'application pour finir par regrouper les solutions des deux approches et les classer par la méthode d'aide à la décision PROMÉTHÉE.

CONCLUSION GÉNÉRALE

Les problèmes de tournées de véhicules font partie intégrante de la vie quotidienne des décideurs et des planificateurs. Le travail que nous présentons, traite un problème de ramassage et de livraison avec fenêtres de temps et à un seul véhicule 1-PDPTW.

Ce problème, fréquemment rencontré dans notre vie courante (ramassage scolaire, transport de fonds, livraison alimentaire, ramassage de déchets) consiste à confectionner une route pour le véhicule qui lui permette de servir tous les clients tout en minimisant la distance totale parcourue, le temps total d'attente et le retard total et en respectant la capacité de ce dernier et les fenêtres de temps des clients.

Dans ce travail, nous avons abordé quelques méthodes d'optimisation dans le cadre de résolution d'un problème multi-objectifs, ces problèmes sont connus par leurs nature particulièrement difficile, ce qui conduit à l'utilisation des méthodes approchées telles que les méta-heuristiques pour les résoudre.

D'abord, nous avons introduit dans le premier chapitre, les problèmes d'optimisation combinatoire en général ainsi que les différentes méthodes de leur résolution. Et nous avons rappelé les principes et les caractéristiques des algorithmes génétiques et de la colonies de fourmis, qui sont les outils de base utilisés dans nos recherches.

Ensuite, nous avons présenté les différents problèmes de d'élaboration de tournées de véhicules. Dont l'objectif est de servir un ensemble de clients en minimisant un ou plusieurs critères liés au coût de livraison des biens et dont fait partie le 1-PDPTW qui a fait l'objet de nos travaux.

Au niveau du troisième chapitre, notre étude a porté sur les problèmes du PDPTW et leur formulation mathématique. Dans cette partie, nous avons présenté la résolution du 1-PDPTW avec la $1^{ère}$ approche évolutionnist et notre approche de colonie de fourmi adaptés.

Enfin, dans une perspective d'amélioration des performances de nos approches, nous nous proposons :

* De développer nos approches pour traiter le cas m-PDPTW afin de pouvoir tester les outils originaux développés, leur comportement et leurs performances vis-à-vis d'un problème faisant appel à plusieurs véhicules.
* Adapter l'algorithme colonie de fourmis proposé pour des variables continues.
* Intégrer d'autres algorithmes, tels que la recherche tabou pour la résolution du 1-PDPTW.
* Développer une approche qui permet de résoudre le m-PDPTW en étudiant le cas où les

véhicules utilisés peuvent échanger la marchandise entre eux, afin de diminuer le temps de circulation et augmenter la satisfaction des clients en leur procurant leurs biens dans les brefs délais.

Résumé

Le VRP est un problème classique qui consiste à construire des routes visitant un ensemble de clients en minimisant le coût de transport, en satisfaisant les demandes de ces derniers et en respectant les différentes capacités des véhicules. Nous nous intéressons à une importante variante du VRP qui est le PDPTW.

Dans ce mémoire nous avons traité un problème de collecte et de distribution avec fenêtres de temps à un seul véhicule 1-PDPTW. Dans ce problème, le véhicule doit servir un ensemble de demandes de biens à transporter à partir d'un ensemble de fournisseurs vers des clients, tout en respectant des contraintes de capacité et des contraintes de temps. Nous avons résolu ce problème en utilisant un algorithme évolutionniste avec des opérateurs génétiques spécifiques (la $1^{ère}$ heuristiques) et notre approche de colonie de fourmis adaptés pour fournir un ensemble de solutions viables. Ces approches de résolution minimisent le compromis entre la distance totale parcourue, le temps total d'attente et le retard total.

Mots clés : VRP, fenêtre de temps, heuristiques, algorithmes évolutionnistes, colonie de fourmi.

Abstract

The VRP is a classical problem that consists on building routes visiting a set of customers minimizing the transportation cost, while satisfying their demands and the different vehicles capacities. We are interested in an important variant of the VRP which is the PDPTW.

In the present work we have dealt with the problem of collecting and delivery with time windows for a single vehicle 1-PDPTW. In this problem, a vehicle has to serve a set of customers from a set of suppliers, while satisfying their demands, under some given capacity requirements. This problem has been solved using an evolutionary algorithm with a special genetic operator in one hand, and by an adapted ant colony approach to provide a set of viable solutions on the other hand.
By these approaches we minimize the trade-off between the total distance trip, the total waiting time and the total delay.

Keywords : VRP, time window, heuristics, evolutionary algorithms, ant colony.

BIBLIOGRAPHIE

[1] M. Gendreau F. Guertin J. Y. Potvin É. Taillard, P. Badeau. A tabu search heuristic for the vehicle routing problem with soft time windows. pages 170–186, 1996.

[2] A. E. Smith A. Konaka, D. W. Coitb. Multi-objective optimization using genetic algorithms : A tutorial. Technical report, Information Sciences and Technology, Penn State Berks, USA Department of Industrial and Systems Engineering, Rutgers University Department of Industrial and Systems Engineering, Auburn University, January 2006.

[3] F. G. Radet A. Souquet. *Algorithmes Genetiques.*

[4] I. Alaya. Optimisation multiobjectif par colonies de fourmis cas des problèmes de sac à dos. Thèse de doctorat, Université de la Manouba Ecole Nationale Des Sciences De L'informatique Et Université Claude Bernard Lyon1, 5 Mai 2009.

[5] A. ASASE. *The Transportation Problem.* In partial fulfilment of the requirements for the degree of master of science, Departement Of Mathematics Faculty Of Physical Science And Technology Kumasi, October 2011.

[6] C. C. White B. S. Stewart. Multiobjective a*. *Journal Of Association For Computing Machinery(JACM)*, 38(4) :775, 1991.

[7] A. Bendahmane. *Le Recuit Simulé.* Université des Sciences et de la Technologie d'Oran - Mohamed Boudiaf, 25 Octobre 2011.

[8] M. brahim. *Modèles De Résolution Approchée Et Efficace Pour Les Problèmes Des Réseaux De Transport Et De Télécommunication.* Thèse de doctorat, Université De Picardie Jules Verne D'amiens,École Doctorale : Sciences, Technologie et Santé (ED 547), 15 Juin 2015.

[9] J. P. Brans and B. Mareshal. *PROMETHEE - GAIA une méthodologie d'aide a la décision en présences de criteres mulitiples.* Ellipces, 2002.

[10] Y. Cooren. *Perfectionnement D'un Algorithme Adaptatif d'Optimisation Par Essaim Particulaire : Application En Génie Médical Et En Électronique.* Thèse de doctorat, L'université 2223 Paris 12 Val De Marne, Mars 2019.

[11] P. Czyzak and A. Jaszkiewicz. Pareto simulated annealing a metaheuristic technique for multiple-objective combinatorial optimisation. *Journal of Multi-Criteria Decision Analysis*, 7 :34–47, 1998.

[12] K. Doerner, W. J. Gutjahr, R. F. Hartl, C. Strauss, and C. Stummer. *Pareto Ant Colony Optimization : A Metaheuristic Approach to Multiobjective Portfolio Selection*, 2004. Annals of Operations Research.

[13] I. H. Dridi. *Optimisation Heuristique Pour La Résolution Du M-PDPTW Statique Et Dynamique.* Doctorat délivré conjointement par l'École centrale de lille et l'École nationale

d'ingénieurs de tunis, École Centrale De Lille Université De Tunis El Manar École Nationale D'ingénieurs De Tunis, 15 décembre 2010.

[14] L. Gambardella, E. Taillard, and M. Dorigo. Ant colonies for the quadratic assignment problem. *Journal of the Operational Research Society*, 50 :167–176, 1999.

[15] A. Gherboudj. Méthodes de résolution de problèmes difficiles académiques. Thèse de doctorat, Université de Constantine2, 2013.

[16] h. Hacene. Etude comparative des heuristiques d'optimisation oombinatoire. Memoire de master, Universite de Biskra, 2015.

[17] H. Hamichi. Hybridations d'algorithmes méthaheuristiques en optimisation globale et leurs applications. Thèse de doctorat en cotutelle, Ecole Mohammadia D'ingénieurs,Université Mohammed V Agdal, 23 Juin 2013.

[18] M. Assaad HAMIDA. *Introduction Aux Methodes De Controle Intelligent.* Faculté des Nouvelles Technologies de l'Information et de la Communications, 2014/2015.

[19] S. Iredi, D. Merkle, and M. Middendorf. Bi-criterion optimization with multi colony ant algorithms. pages 359–372 of : First International Conference on Evolutionary Multi–criterion Optimization, 2001.

[20] A. Jaszkiewicz. Genetic local search for multiple objective combinatorial optimization. *European Journal of Operational Research*, 137((1)) :50–71, 2002.

[21] R. Kammarti. *Approches Evolutionnistes Pour La Resolution Du 1-PDPTW Statique ET Dynamique.* These présentée en vue d'obtenir le grade de docteur, Ecole Centrale De Lille Universite Des Sciences Et Technologies De Lille, 13 décembre 2006.

[22] A. Landrieu, Y. Mati, and Z. Binder. A tabu search heuristic for the single vehicule pickup and delivery problem with time windows. *Journal of Intelligent Manufacturing*, 12 :497–508, 2001.

[23] V. Maniezzo M. Dorigo and A. Colorni. Ant system : Optimization by a colony of cooperating agents. *IEEE Transactions On Systems, Man, And Cybernatics-Part B Cybernetics*, 26(1) :29–41, February 1996.

[24] S. MAHDI. Optimisation multiobjectif par un nouveau schéma de coopération méta/exacte. Thèse de magister, Université Mentouri de Constantine.

[25] A. Malapert. Optimisation de tournées de véhicules pour l'exploitation de réseau telecom. Technical report, Université Paris 6 Pierre et Marie Curie UFR d'Informatique, 2006.

[26] C. E. Mariano and E. Morales. A multiple objective ant-q algorithm for the design of water distribution irrigation networks. *Tech. Rept. HC-9904*, June 1999.

[27] M. Palpant. Recherche exacte et approchee en optimisation combinatoire : schemas d'integration et applications. Thèse de doctorat, Universite d'Avignon, 2005.

[28] H. N. Psaraflis. Analysis of an $o(n^2)$ heuristic for the single vehicule many-to-many euclidean dial-a-ride problem. *Transportation science*, 17 :133–145, 1983.

[29] H. N. Psaraflis. An exact algorithm for the single vehicule many-to-many immediate request dial-a-ride problem with time windows. *Transportation science*, 17 :351–357, 1983.

[30] H. Moskowitz R. Carraway, T. L. Morin. Generalized dynamic programming for multicriteria optimization. *European Journal Of Operational Research,*, 44 :95–104, 1990.

[31] C. Reeves. *Modern Heuristic Techniques for Combinatorial Problems. Advances topics in computer science.* 1995.

[32] A. SAHA. Résolution des problèmes multi objectifs à base de colonies de fourmi. Thèse de magister en informatique, Université de Batna.

[33] M. S.Bazaraa and J. J.Jarvis. *Linear Programming And Network Flows.* School Of Industrial And Systems Engineering Georgia Institute of Technology ,Atlanta,Georgia, 1943.

[34] M. Sevaux. *Métaheuristiques : Stratégies Pour l'Optimisation De La Production De Biens Et De Services.* Habilitation À diriger des recherches, Université de Valenciennes et du Hainaut-Cambrésis, Juillet 2004.

[35] T. R. Sexton and L. D. Bodin. Optimisation single vehicule many-to-many operations with desired delivery times : I. scheduling. *Transportation Science*, 19 :378–410, 1985.

[36] T. R. Sexton and L. D. Bodin. Optimisation single vehicule many-to-many operations with desired delivery times : Ii. routing. *Transportation Science*, 19 :411–435, 1985.

[37] M. M. Solomon. Algorithms for the vehicule routing and sheduling problem with times window contraints. *Operations Research*, 41 :469–488, 1987.

[38] M. A. Tahraoui. Colonie de fourmis pour le problÉme du transport multimodal. Thèse de magister, Université Mira Abderrahmene de Béjaia, Janvier 2009.

[39] E. Talbi. Métaheuristique pour l'optimisation combinatoire multi-objectif :etat de l'art.

[40] F. Troudi. Résolution du problème de l'emploi du temps : Proposition d'un algorithme Évolutionnaire multi objectif. Thèse de magister en informatique, Université Mentouri – Constantine, 2005-2006.

[41] A. V. Zykina and O. N. Kaneva. Algorithm for solving stochastic transportation problem. 2017. Omsk State Technical University prospekt Mira 11, 644050 Omsk, Russia.