RÉPUBLIQUE ALGÉRIENNE DÉMOCRATIQUE ET POPULAIRE

Ministère de l'Enseignement Supérieure et de la Recherche Scientifique

Université Mira Abderrahmene de Béjaia

COLONIE DE FOURMIS POUR LE PROBLÉME DU TRANSPORT MULTIMODAL

PAR

TAHRAOUI Mohammed Amine

PRÉSENTÉ À LA FACULTÉ DES SCIENCES ET SCIENCES DE L'INGÉNIEUR

DÉPARTEMENT D'INFORMATIQUE, ÉCOLE DOCTORALE D'INFORMATIQUE

ReSyD

(Réseaux et Systèmes Distribués)

POUR L'OBTENTION DU GRADE DE MAGISTÈRE EN INFORMATIQUE

À

L'Université Mira Abderrahmene

Béjaia, 06000

Accepté sur proposition du jury

Président :	Moussa Kerkar, Professeur à Université de Béjaïa, Algérie
Rapporteur :	Zineb Habbas, MCF-HDR, IUT de Metz, France
Examinateurs :	Boukeram Abdellah, MCF à l'Université de Sétif, Algérie
	Moussaoui Abdelouhab, MCF à l'Université de Sétif, Algérie
Invité :	Aloui Abdelouhab, Chargé de cours, Université de Béjaïa, Algérie

UNIVERSITÉ MIRA ABDERRAHMENE

École doctorale d'informatique, ReSyD.

Date: **Janvier 2009**

Nom : **Tahraoui Mohammed Amine.**

Titre : **Optimisation par Colonie de Fourmis pour le probléme du Déplacement Multimodal.**

Département : **Informatique.**

Grade : **Magistère en informatique.**

A ma mère

A mon père

A mon grand-père

A toute ma famille

Table des Matières

Liste des tableaux

Table des figures

Remerciements

Je loue Dieu de la grâce qu'il m'a faite en me donnant la santé, la patience et la détermination sans les quelles je n'aurais jamais pu porter mon projet à son terme.

Je voudrais tout d'abord exprimer mes remerciements et ma gratitude au Mme Habbas pour m'avoir encadré durant ce projet, son aide et sa confiance m'ont grandement aidé à mener à bien mon travail, ainsi que Mr Aloui pour ses conseils et critiques.

Je remercie les membres de jury qui ont accepté de juger ce travail. J'adresse mes très sincères remerciements à Mr Kerkar, Mr Boukeram et à Mr Moussaoui de me faire l'honneur de s'intéresser à ce travail et d'avoir accepté de faire partie de jury.

Je remercie tous mes collègues de l'école doctorale et notamment Abd Elkader pour avoir lu et corrigé mon mémoire.

Enfin, et surtout, je remercie vivement ma famille qui m'a beaucoup soutenu et encouragé, en particulier mes parents, mes soeurs qui ont été toujours derrière moi pour m'encourager et me soutenir. Merci encore.

Introduction

L'OPTIMISATION des réseaux de transport, constitue un des grands axes d'application de la recherche opérationnelle. Au cours de ces dernières années, la croissance de la taille des réseaux et les multiples évolutions technologiques en transport a fait émerger une nouvelle politique de déplacement : le transport multimodal. Dans la littérature, le terme "multimodal" est utilisé pour désigner un système qui offre la possibilité d'utiliser plus de deux modes de transport pour déplacer des passagers ou des marchandises, d'un point à un autre. D'autre part, plusieurs dates de départ sont planifiées pour chaque mode au niveau de chaque station de transport. De nos jours, les systèmes multimodaux d'information se basent essentiellement sur des données dynamiques notamment en fonction de la date de départ et du mode de transport choisis au niveau des stations. Ces caractéristiques induisent de nouveaux défis, de nouvelles contraintes auxquels il faut répondre, à savoir comment planifier, comment optimiser, comment réguler le réseau de transport et quoi faire en cas de perturbation, ...

L'évaluation d'un réseau de transport est réalisée en déterminant les caractéristiques des meilleurs itinéraires à emprunter pour réaliser un déplacement d'un point source à un point destination, selon un certain nombre de considérations. Ce qui revient aussi à résoudre une variante particulière du problème de Plus Court Chemin (PCC), qui est l'un des problèmes les plus classiques d'optimisation combinatoire. Dans ce contexte, le plus court chemin multimodal se traduit par la recherche de la meilleure combinaison des dates et des modes de transport qu'un usager doit emprunter pour réaliser son déplacement en minimisant des critères particuliers tels que le temps et le coût.

Les approches de résolution exactes du plus court chemin proposées dans la littérature sont des solutions efficaces pour le cas mono-objectif du problème de taille réduite. Cependant, elles s'avèrent moins efficaces lorsqu'elles sont adoptées pour un réseau de taille importante et pour la résolution de la variante multicritère de recherche d'itinéraire. En conséquence, l'utilisation d'une méthode exacte pour le problème de plus court chemin multi-objectif dans un réseau de transport multimodal, a priori, est impossible. Pour pallier à cet inconvénient, il est nécessaire voire même indispensable d'avoir recours à des approches de résolution dites *métaheuristiques*. Généralement, ces outils sont reconnus et appréciés pour leur capacité à fournir des

solutions de très bonne qualité avec des temps de calculs raisonnables. Les algorithmes de colonies de fourmis forment une classe des métaheuristiques récemment proposée pour les problèmes d'optimisation difficiles. C'est cette métaheuristique qui en l'occurrence a été choisie dans le cadre de ce mémoire.

L'optimisation par colonie de fourmis (OCF) introduite par Dorigo et al [16] est une métaheuristique où le comportement des fourmis artificielles est totalement inspiré du comportement des fourmis réelles avec une adaptation pour le problème considéré. Le principe de base de l'OCF est de modéliser le problème à résoudre sous forme d'un graphe, et d'utiliser des fourmis artificielles pour trouver le plus court chemin. l'apprentissage dans un algorithme de colonie de fourmis est réalisé à l'aide de la trace de phéromone qui constitue un élément central dans cette métaheuristique.

Peu d'efforts ont été faits jusqu'à maintenant pour la résolution du PCC dans un réseau de transport multimodal par les métaheuristiques. C'est en partie ce qui motive l'orientation du présent travail de recherche vers l'adaptation d'un algorithme d'OCF dans le but de résoudre de manière efficace le problème de plus court chemin multimodal. La définition d'une approche d'optimisation variée selon la variante du problème traité. Dans notre contribution, nous avons utilisé une démarche progressive de développement et de résolution (du plus facile au plus difficile). Plusieurs versions d'algorithmes OCF ont été proposées et validées. L'objectif à long terme de ces travaux est de proposer une approche capable :

- d'aborder des systèmes de transport de taille importante.
- d'intégrer un nombre important de critères.
- de fournir un ensemble de solutions plutôt qu'une seule solution.
- d'ajouter d'autres hypothèses (panne d'un mode de transport, rupture d'un chemin d'une manière stochastique, etc.)

Le présent document est organisé de la manière suivante :

- **Chapitre 1(Optimisation par colonie de fourmis et problème de transport multimodal)** : Ce chapitre est dédié à un état de l'art. Il a pour but de présenter une synthèse des deux sujets abordés dans ce mémoire : L'Optimisation par Colonie de Fourmis(OCF) et le problème de transport multimodal. Avant d'introduire la méthode OCF, il dresse un panorama sur les métaheuristiques.

- **Chapitre 2(OCF mono-objectif appliquée au réseau de transport multimodal)** : Ce chapitre présente deux versions OCF mono-objectif pour résoudre le problème de plus court chemin multimodal (OCF1-PCC-RTM et OCF2-PCC-RTM), la deuxième est la version MAX-MIN Ant-System de la

première. Ces deux versions sont étendues pour résoudre le problème du plus court chemin dans le cas d'optimisation bi-objectif multimodal en cherchant à minimiser le temps du trajet et les temps d'attentes.

- **Chapitre 3(OCF multi-objectif appliqué au réseau de transport multimodal)** : Ce chapitre poursuit le travail développé au chapitre 2, en proposant un algorithme OCF générique afin de résoudre la variante multiobjectif du problème de plus court chemin, basée sur l'approche Pareto.

- **Chapitre 4 (Bi-colonie pour l'optimisation d'un déplacement multimodal dans un réseau perturbé**) : Ce chapitre définit une nouvelle approche OCF fournissant un support pour l'aide au déplacement multimodal en mode dégradé de fonctionnement du réseau.

- Enfin, ce mémoire se termine par une conclusion générale qui propose quelques perspectives de recherche.

Chapitre 1

Optimisation par colonie de fourmis et problème de transport multimodal

1.1 Introduction

La notion de multimodalité dans les réseaux de transport collectif fait l'objet de plusieurs thématiques de recherche liées à l'aménagement des réseaux. Plusieurs travaux ont été effectués dans l'optimisation du réseaux de transport durant les cinquante dernières années. Le problème de plus court chemin est l'un des problèmes les plus étudiés dans ce domaine. Son but consiste à trouver le meilleur chemin entre deux points quelconques dans un réseau donné, ceci revient à la détermination de l'itinéraire entre les points sources et destinations en minimisant des critères particuliers tels que le temps, le coût,..., etc. Ce genre de problème fait partie de la classe des problèmes combinatoires NP-difficiles. Les algorithmes connus pour les résoudre nécessitent un temps exponentiel en fonction du nombre de critères considérés et la taille du réseau traité. L'arrivée d'une nouvelle classe de méthodes, nommée métaheuristique permet de pailler cet inconvénient sans toutefois garantir l'optimalité.

Ce chapitre a pour but de présenter une vue sur la littérature relative aux deux sujets principaux de ce mémoire, soit l'Optimisation par Colonie de Fourmis (OCF), et le problème de transport multimodal.

La Section 1 porte sur les métaheuristiques. Un rappel est d'abord fait sur les principaux concepts en optimisation combinatoire et les méthodes utilisées pour la résolution des problèmes NP-Difficiles. Parmi ces méthodes se trouvent la métaheuristique définie ci-dessous.

L'optimisation par colonie de fourmis est abordée en détails avec ses variantes de base en section 2. En section 3, il sera détaillé le système de transport et les variantes du problème de plus court chemin, en présentant les différents modèles proposés pour

la représentation graphique du problème considéré. Ainsi, une implémentation objet est retenue.

1.2 Métaheuristiques pour l'optimisation difficile

1.2.1 Optimisation difficile

L'optimisation combinatoire occupe une place très importante en recherche opérationnelle, en mathématiques discrètes et en informatique. Son importance se justifie d'une part par la grande difficulté des problèmes d'optimisation et d'autre par de nombreuses applications pratiques pouvant être formulées sous la forme d'un problème d'optimisation combinatoire. Bien que les problèmes d'optimisation combinatoire soient souvent faciles à définir, ils sont généralement difficiles à résoudre. En effet, la plupart de ces problèmes appartiennent à la classe des problèmes NP-difficiles et ne possèdent donc pas à ce jour de solution algorithmique efficace valable pour toutes les données [33].

L'optimisation combinatoire trouve des applications dans des domaines aussi variés que la gestion, l'ingénierie, la conception, la production, la télécommunication, le transport, l'énergie, la médecine et l'informatique elle-même.

1.2.2 Problème d'optimisation

Le monde réel offre un ensemble très divers de problèmes d'optimisation :
- Problème combinatoire discret ou à variables continues.
- Problèmes à un ou plusieurs objectifs.
- Problèmes statiques ou dynamiques.
- Problème dans l'incertain.

Cette liste n'est évidemment pas exhaustive, et un problème peut être à la fois multi-objectif et dynamique. Dans ce qui suit, on se limite à l'optimisation mono-objectif. Le cas muli-objectif sera introduit dans le troisième chapitre.

Un problème d'optimisation consiste à trouver, parmi un ensemble de solutions potentielles, une solution optimale au regard d'un critère donné. De manière plus formelle, un Problème d'Optimisation (PO) peut être représenté par un modèle $PO = \{S, X, D, C, F\}$ où :
- S : l'espace de recherche défini par un ensemble de solutions potentielles.
- $X = (x_1, x_2, ..., x_r)$: un vecteur de variables de décision.
- $D = (d_1, d_2, ..., d_r)$: un vecteur de valeurs définissant les domaines des variables de décision.
- C : un ensemble de contraintes sur X, c'est-à-dire, un ensemble de relations limitant les valeurs qui peuvent être simultanément assignées aux variables de

décision.

– F : une fonction objectif qui associeé à chaque solution s une valeur $F(s)$.

La variable générique x_i prend des valeurs en domaine d_i. Une solution faisable est une attribution complète des valeurs aux variables qui satisfait l'ensemble de contraintes C. Résoudre un tel problème (plus précisément une instance du problème) consiste à trouver une solution $s^* \in S$ optimisant la valeur de la fonction coût F. Une telle solution s^* s'appelle une *solution optimale* ou un *optimum global*. Nous avons donc la définition suivante :

Définition 1. *Une solution $s \in S$ est un minimum global si $F(s) \leq F(s')$ pour tout $s' \in S$.*

1.2.3 Méthode de résolution

De nombreuses méthodes de résolution ont été développées pour résoudre ces problèmes d'optimisation. Ces méthodes peuvent être classées sommairement en deux grandes catégories : les méthodes exactes (complètes) qui garantissent la complétude de la résolution et les méthodes approchées (incomplètes) qui perdent la complétude pour gagner en efficacité.

1.2.3.1 Méthode exacte

Le principe essentiel d'une méthode exacte consiste généralement à énumérer exhaustivement l'ensemble des solutions de l'espace de recherche, où au moins avoir l'assurance de n'écarter aucune solution ayant le potentiel d'être meilleure que la solution optimale trouvée par l'algorithme. Il existe de nombreuses méthodes exactes (déterministes) permettant de résoudre certains problèmes dans un temps fini, on peut citer comme exemple : la programmation linéaire, quadratique ou dynamique, la méthode de séparation- évaluation (branch and bound), la méthode du gradient, ...

1.2.3.2 Méthode approchée

Les méthodes de résolution approchées appelées *métaheuristiques* sont généralement utilisées dans la résolution des problèmes combinatoire où les méthodes exactes échouent, c'est-à-dire, qu'aucun algorithme exact ne s'exécutant en temps polynomial n'est connu pour ces problèmes, les métaheuristiques offrent un compromis intéressant. Elles ne garantissent pas l'optimalité des solutions, mais elles s'exécutent dans un temps considérablement plus court que les méthodes dites "exactes". Ces dernières sont très souvent inapplicables, parce qu'elles sont trop coûteuses en termes du temps de calcul et d'espace mémoire. L'informatique contribue grandement au

développement des métaheuristiques en leur donnant une force et une capacité de calcul de plus en plus grande [43]. Le terme métaheuristique a été introduit par Glover en 1986, pour différencier la recherche avec tabous des autres heuristiques, ce terme est dérivé de la composition de deux mots grecs :

1. *heuristique* qui vient du verbe heuriskein qui signifie 'trouver'
2. *meta* qui est un suffixe veut dire 'au-delà', un passage à un niveau supérieur pour étudier ou manipuler des informations de niveau inférieur.

Les propriétés fondamentales des métaheuristiques sont les suivantes [5] :

- Le but visé par les métaheuristiques est d'explorer efficacement l'espace de recherche afin de déterminer des solutions (presque) optimales.
- Les techniques qui constituent les algorithmes de type métaheuristique vont de la simple procédure de recherche locale à des processus d'apprentissage complexes.
- Les concepts de base des métaheuristiques peuvent être décrits de manière abstraite, sans faire appel à un problème spécifique.
- Les métaheuristiques peuvent faire appel à des heuristiques qui tiennent compte de la spécificité du problème traité, mais ces heuristiques sont contrôlées par une stratégie de niveau supérieur.
- Les métaheuristiques peuvent faire usage de l'expérience accumulée durant la recherche de l'optimum, pour mieux guider la suite du processus de recherche.

La littérature consacrée aux métaheuristiques est très riche en termes d'approches de résolution et ses applications aux problèmes combinatoire. Sans être détaillée ni exhaustive, la suite de cette section sera consacrée à la présentation sommaire des métaheuristiques les plus utilisées : les méthodes de recherche locale et les algorithmes évolutionnaires. Par la suite, nous expliquons en détail les algorithmes de colonies de fourmis.

1. Recherche Locale

L'idée naturelle de cette classe est d'améliorer de façon itérative une solution initiale en explorant un voisinage de celle-ci dans l'espace de recherche. La méthode de descente, le recuit simulé et la recherche tabou sont les méthodes de recherche locale les plus connues et les plus utilisées.

Descente : Les méthodes de descente sont très anciennes et doivent leurs succès à leur rapidité et leur simplicité. Le principe de base consiste à utiliser une structure de voisinage (N) pour explorer l'espace de recherche. A chaque pas, ces méthodes progressent vers une solution voisine de meilleure qualité. Le processus est itéré jusqu'à ce que tous les voisins candidats sont moins bons que la solution courante. On distingue différents types de descente en fonction de la règle de déplacement définie dans la structure de voisinage. Cet algorithme de descente (appelé aussi algorithme

classique d'amélioration itérative) converge rapidement mais, ne conduit pas, en général, au optimum global, mais seulement à un optimum local. Nous donnons maintenant le pseudo-code d'un algorithme type de descente pour un problème de minimisation :

Algorithm 1.1 Descente

1: **Début**
2: choisir une solution initiale $s \in S$.
3: Déterminer une solution s' qui minimise F dans $N(s)$.
4: **si** $F(s') < F(s)$ **alors**
5: $s' \leftarrow s$ et retourner à 3
6: **sinon**
7: Arrêter.
8: **finsi**
9: **Fin.**

Recuit Simulé : Les algorithmes de recuit simulé, introduits dans le domaine d'optimisation en 1983 par Kirkpatrick et al [39], sont des méthodes inspirées des principes de la physique et des refroidissements des métaux. L'idée fondamentale consiste à autoriser le déplacement d'une solution s vers une solution $s' \in N(s)$ telle que $F(s') > F(s)$. La probabilité de faire une telle démarche est calculée en fonction de la température T qui est ajustée au fur et à mesure que le temps avance. Le pseudo code du recuit simulé est représenté dans l'algorithme 1.2.

Algorithm 1.2 Recuit Simulé

1: **Début**
2: Sélectionner une solution initiale $s \in S$.
3: Sélectionner une température initiale $T > 0$.
4: Sélectionner au hasard $s' \in N(s)$.
5: $\sigma \leftarrow F(s') - F(s)$
6: **si** $\sigma < 0$ **alors**
7: $s \leftarrow s'$
8: **sinon**
9: $x \leftarrow hasard([0, 1])$
10: **si** $x < \exp^{-\sigma/T}$ **alors**
11: $s \leftarrow s'$
12: **finsi**
13: **finsi**
14: Mettre à jour T
15: Aller à l'étape 4 si la condition d'arrêt n'est pas vérifiée.
16: **Fin.**

En général, on choisit une température initiale suffisamment élevée qui donne une plus grande liberté pour l'exploration de l'espace de recherche. Puis, petit à petit, la température décroît jusqu'à atteindre une valeur proche de 0, ce qui signifie que la méthode n'acceptera plus de détériorer la solution courante [5].

Le recuit simulé a été appliqué avec succès à un grand nombre de problèmes d'optimisation NP-difficiles tels que, le problème d'affectation quadratique [12], et les problèmes d'ordonnancement de type job-shop [52].

Recherche Tabou : La recherche tabou est introduite par Glover [27]. Son fonctionnement est plus proche de la méthode de descente que du recuit simulé. Tabou considère un ensemble de points voisins du point courant, puis le meilleur voisin, dans cet ensemble, est choisi pour être le point courant suivant, même s'il est de qualité inférieure au point courant précédent. Cependant, cette stratégie a tendance à revenir sur des points déjà visités, c'est pourquoi la méthode tabou utilise une mémoire à court terme : *la liste tabou*. Cette liste conserve une information sur les dernières itérations de l'algorithme qui permet de se prémunir contre les cycles courts [2]. Les opérations interdites sont alors déclarées "tabou", d'où le nom de la méthode, ce qui a pour effet de restreindre le voisinage aux solutions les plus susceptibles de diriger l'exploration vers des régions inexplorées. La recherche se poursuit jusqu'à ce que un nombre donné d'itérations soit effectué, ou on peut fixer un temps limite après lequel la recherche doit s'arrêter. En résumé, le pseudo-code de cette méthode est représenté dans l'algorithme 1.3 :

Algorithm 1.3 Recherche Tabou

1: **Début**
2: Sélectionner une solution initiale $s \in S$.
3: $Tabou \leftarrow \varnothing$,$s^* \leftarrow s$.
4: Déterminer une solution s' qui minimise F dans $N(s)$.
5: **si** $F(s') < F(s)$ **alors**
6: $\quad s^* \leftarrow s'$.
7: **finsi**
8: $s' \leftarrow s$
9: Mettre à jour la liste $Tabou$.
10: Aller à l'étape 4 si la condition d'arrêt n'est pas vérifiée.
11: **Fin.**

La méthode tabou a donné de très bons résultats pour de nombreux problèmes difficiles, en plus, la méthode comporte moins de paramètres de réglage que le recuit simulé, ce qui la rend plus simple d'emploi. Pour une description complète de la méthode, le lecteur pourra utilement se reporter au papier de Glover et Laguna [28].

2. Les algorithmes évolutionnaires

Les Algorithmes Evolutionnaires (AE) sont des métaheuristiques à base de population qui s'inspirent des mécanismes de l'évolution naturelle. Une première adaptation des AE en optimisation combinatoire grâce à un algorithme génétique, a été introduite par Holland en 1975 [34].

Pour un problème d'optimisation donné, les points de l'espace de recherche sont appelés *individus* ou *chromosome*, ils sont constitués d'un ensemble de variables

appelés gènes, une population de solutions est donc un ensemble de chromosomes. Le pseudo-code standard de cette méthode est décrit à l'algorithme 1.4. Dans cet algorithme, On commence par générer une population initiale d'individus. Chaque individu de cette population est évalué selon une fonction objectif F, qui mesure son efficacité dans l'espace de recherche (on appelle aussi cette valeur l'adaptation). Ensuite, la population de solutions est évoluée progressivement, par générations successives en y appliquant des opérateurs de sélection, de croissement et de mutation. L'objectif de ces opérateurs est d'améliorer globalement les performances des individus de la population courante. La sélection permet de favoriser les meilleurs individus de la population, le croisement et la mutation ont pour but d'assurer une intensification et une diversification efficace de l'espace de recherche.

Algorithm 1.4 Algorithme génétique

1: **Début**
2: Générer aléatoirement une population initiale.
3: **répéter**
4: Evaluer chaque individu de la population selon F.
5: Sélectionner les parents dans la population.
6: Produire les enfants des parents sélectionnés par croisement.
7: Muter les individus.
8: **jusqu'à** satisfaction du critère d'arrêt.
9: Retourner la meilleure solution trouvée.
10: **Fin**.

Les algorithmes évolutionnaires sont capables de s'adapter à n'importe quel problème d'optimisation. ils sont largement utilisés pour un très grand nombre de problèmes d'optimisation NP- difficiles. On peut citer, par exemple, l'ordonnancement de la production, les systèmes d'aide à la décision, optimisation multi-critères et divers problèmes d'optimisation du domaine de transport.

3. Quelques autres métaheuristiques

Nous avons énuméré les grandes classes des métaheuriqtiques pour l'optimisation combinatoire. Evidement, d'autres méthodes de métaheuristiques moins répandues, existent dans la littérature. Nous pouvons citer notamment la méthode GRASP [23], l'optimisation par particule d'essaim [38], les systèmes immunitaires artificiels [21] et d'autres approches basées sur la logique floue. Actuellement, une attention particulière est portée sur une nouvelle classe de méthodes approchées qui implique une combinaison de plusieurs métaheuristiques, il s'agit d'une métaheuristique hybride.

1.3 Optimisation par Colonie de Fourmis

L'intelligence d'essaim est une approche relativement nouvelle, elle s'inspire des comportements sociaux des insectes et d'autres animaux. En particulier, les fourmis ont inspiré un certain nombre de méthodes et de techniques appliquées à l'optimisation, la plus réussi est la technique d'optimisation connue sous le nom Optimisation par Colonie de Fourmis (OCF).

1.3.1 Inspiration Biologique

Les éthologistes étudiant le comportement des insectes sociaux ont observé que la coopération au sein des colonies est auto-organisée, elle semble résulter uniquement des interactions entre les individus sans être supervisé d'une quelconque manière. Bien que ces interactions soient simples (par exemple, une fourmi se contente de suivre la trace odorante laissée par une autre), elles permettent à la collectivité de résoudre des problèmes difficiles, telle la recherche du chemin le plus court entre le nid et une source de nourriture, parmi d'innombrables voies possibles.

En marchant du nid à la source de nourriture et vice-versa (ce qui, dans un premier temps, se fait essentiellement d'une façon aléatoire), les fourmis déposent au passage sur le sol une substance odorante appelée *phéromone*, ce qui a pour effet de créer une piste chimique. Les fourmis peuvent sentir ces phéromones qu'ont un rôle de marqueur de chemin : quand les fourmis choisissent leur chemin, elles ont tendance à choisir la piste qui porte la plus forte concentration de phéromone. Cela leur permet de retrouver le chemin vers leur nid lors du retour. Il a été démontré expérimentalement que ce comportement permet l'émergence des chemins les plus courts entre le nid et la nourriture, à condition que les pistes de phéromones soient utilisées par une colonie entière de fourmis. Autrement dit, quand plusieurs chemins existent entre le nid et la nourriture, une colonie peut exploiter les phéromones déposés par des fourmis individuelles pour découvrir le chemin le plus court entre le nid et la nourriture (voir la figure 1.1 [17]).

Il semble donc que le comportement des fourmis réelles est un type de mécanisme d'optimisation distribué, à travers lequel chaque individu ne peut fournir qu'une très petite contribution. Il est intéressant de remarquer que, bien qu'une seule fourmi soit capable de construire une solution (i.e. de trouver un chemin du nid à la nourriture), c'est seulement le comportement du colonie qui crée le chemin le plus court. En un sens, l'optimisation est une propriété émergente de la colonie de fourmis [36].

1.3.2 Technique d'optimisation

Le comportement de fourmis réelles était la source principale d'inspiration pour la résolution de nombreux problèmes d'optimisation combinatoires sous le nom d'une

nouvelle métaheuristique "optimisation par les colonies de fourmis ou OCF". Dans l'OCF, un certain nombre de fourmis artificielles établissent des solutions au problème d'optimisation considéré et échangent l'information sur la qualité de ses solutions par l'intermédiaire d'un arrangement de communication qui est réminiscent de celui adopté par les fourmis réelles [13].

Différents algorithmes d'OCF sont proposés dans la littérature. L'algorithme d'optimisation original est connu sur le nom *Ant System* a été proposé par Dorigo et al [19]. Dés lors, un certain nombre d'algorithmes d'OCF ont été présentés [26, 18, 49, 10].

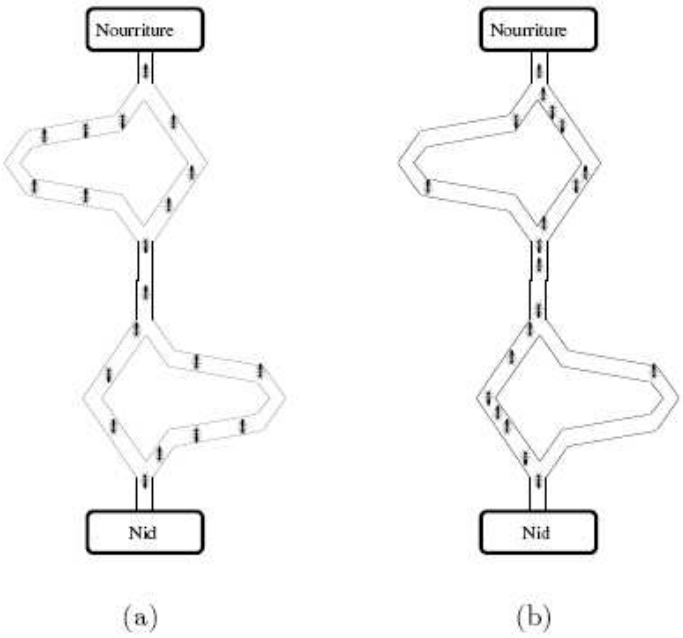

FIG. 1.1 – Expérience du double pont binaire :(a) au début de l'expérience, (b) à la fin de l'expérience.

1.3.3 La Métaheuristique OCF pour l'optimisation combinatoire

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

- 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.

Dans les algorithmes OCF, une population finie (colonie) d'agent (fourmis artificielles) résolvent collectivement le problème combinatoire en utilisant la représentation sous forme de graphe $G(V, E)$ pour trouver des solutions de bonne qualité. tandis qu'un agent construit sa propre solution, elle modifie la perception du problème par les autres agents en fonction des caractéristiques du problème et de ses propres performances. Les fourmis artificielles se déplacent d'un sommet à un autre en établissant incrémentalement une solution partielle. En plus, les fourmis déposent une certaine quantité de phéromone sur les sommets ou sur les bords qu'elles traversent. La quantité de phéromone déposée dépend de la qualité de la solution trouvée. Les fourmis suivantes emploient l'information de phéromone comme un guide vers des régions prometteuses de l'espace de recherche.

La métaheuristique OCF est montrée dans l'algorithme 1.5. Après l'initialisation, l'OCF effectue un certain nombre d'itérations : à chaque itération, un certain nombre de solutions sont construites par les fourmis, ces solutions sont alors améliorées par une phase de recherche locale (cette étape est facultative), et finalement la mise à jour des traces de phéromones est effectuée. Ce qui suit est une description plus détaillée de ces trois phases.

Algorithm 1.5 OCF

```
1: Début
2: Initialisation().
3: tantque (état d'arrêt non rencontrer) faire
4:    Construction Solutions().
5:    Recherche Locale().
6:    Mise a jour phéromone().
7: fin tantque
8: Fin.
```

1. Construction des Solutions

Un ensemble de fourmis artificielles construit des solutions (un ensemble fini de composants). La construction des solutions commence par une solution partielle $s_p = \varnothing$, à chaque étape de construction, la solution partielle s_p est prolongée en ajoutant un composant faisable de solution de l'ensemble $N(s_p)$ (les composants voisins ou les successeurs de s_p). Le choix d'un composant est guidé par un mécanisme stochastique, en fonction de la quantité de phéromone liée à chacun des éléments de l'ensemble $N(s_p)$. La règle pour le choix stochastique des composants de solution change à travers différentes variantes d'OCF, mais, elle est toujours inspirée par le modèle du comportement des fourmis réelles.

2. La Recherche Locale

La métaheuristique de colonie de fourmis est souvent plus efficaces quand elles est hybridée avec des algorithmes de recherche locale. Ceux-ci optimisent les solutions trouvées par les fourmis, avant que celles-ci ne soient utilisées pour la mise à jour des pistes de phéromone. Du point de vue de la recherche locale, utiliser des algorithmes de colonies de fourmis pour engendrer une solution initiale est un majeur avantage.

3. Mise à jour des phéromones

Les informations réunies par les fourmis depuis le début du processus de recherche, sont codées sous la forme de traces de phéromone, elles représentent une mémoire collective de communication à long terme qui influence le comportement des fourmis [36]. La mise à jour de la trace de phéromone est réalisée dans le but de mieux diriger la recherche. Cette opération s'effectue en deux étapes. Dans une première étape, toutes les traces de phéromone sont diminuées, pour simuler l'évaporation en multipliant chaque composant phéromonal par un ratio de persistance ρ tel que $0 < \rho < 1$. Dans un deuxième temps, elle consiste à augmenter les niveaux de phéromone liés à un ensemble choisi de bonnes solutions.

1.3.4 Principaux variantes d'OCF

Plusieurs variantes de colonies de fourmis sont proposées dans la littérature. Ici, nous présentons l'algorithme original *Ant System*, et les deux variantes les plus réussies : *Ant Colony System* et *MAX-MIN Ant System*.

1.3.4.1 Ant System

L'Ant System (AS) a été présenté en 1991 par Marco Dorigo dans sa thèse de doctorat [16]. Il est le premier algorithme de fourmis à apparaître et le prototype de plusieurs autres méthodes. L'AS a été créé au départ pour la résolution du problème de voyageurs de commerce ("Travelling Salesman Problem", TSP). Ce problème consiste à trouver le plus court chemin hamiltonien dans un graphe complètement connecté. Il s'agit sans doute du problème d'optimisation NP-Difficile le plus utilisé comme test pour cette nouvelle méthode d'optimisation.

Le fonctionnement général de AS (Algorithme 1.6) consiste à construire des chemins pour un ensemble fini de fourmis ($k = 1, \ldots, m$), de manière probabiliste en se basant sur la mémoire collective (la trace de phéromone) pour un nombre de cycles donnés. Un cycle, aussi appelé itération, est complété chaque fois que toutes les fourmis ont terminé la construction d'une solution (toutes les fourmis ont parcouru les n sommets qui composent le graphe).

Algorithm 1.6 Ant System

Début
$\tau_{rs} \leftarrow \tau_0$, $\eta_{rs} \leftarrow 1/d_{rs}$ $(\forall rs)$
$t \leftarrow 1$
pour $k = 1, \ldots, m$ **faire**
  Choisir une ville i au hasard et affecter $tabou_k \leftarrow i$
  tantque ($|tabou_k|$ ≠nombre villes) **faire**
    Choisir une ville $j \notin tabou_k$, selon la formule 1.4
    $tabou_k \leftarrow (tabou_k, j)$
  fin tantque
  Calculer la quantité $\Delta\tau_{ij}^k$ conformément à l'équation 1.3
fin pour
Mettre à jour τ_{rs} selon la formule 1.1
$t \leftarrow t + 1$
Vider toutes les listes tabou.
si $(t \leq tmax)$ ou (pas de stagnation) **alors**
  aller à l'étape 4
sinon
  Arrêter
finsi
Fin.

Au départ, tous les éléments de τ_{ij} (La trace de phéromone sur les diffèrent arcs (i, j) sont initialisées à une petite valeur constante positive τ_0. Ces variables de phéromones sont mises à jour de manière globale, à la fin de chaque cycle t $(1 \leq t \leq tmax)$ et selon la formule 1.1, où ρ est un paramètre d'évaporation.

$$\tau_{ij} = (1 - \rho) \times \tau_{ij} + \Delta\tau_{ij} \qquad (1.1)$$

On calcule d'abord $\Delta\tau_{ij}$ pour chaque arête (i, j), selon la formule 1.2, où $\Delta\tau_{ij}^k$ est la quantité de phéromone laissée par une fourmi k sur l'arc (i, j). Cette quantité se calcule selon la formule 1.3, où Q est une constante, L_k est la longueur du chemin trouvé par la fourmi k. Ainsi, la quantité de phéromone laissée par une fourmi sur une arête est fonction de la qualité de la solution finale qu'elle a produite. Toutes les fourmis participent à la mise à jour de la trace et elles n'ajoutent aucun phéromone sur les arêtes qu'elles n'ont pas empruntées pour construire leur solution.

$$\Delta\tau_{ij} = \sum_{k=1}^{m} \Delta\tau_{ij}^k \qquad (1.2)$$

$$\Delta\tau_{ij}^k = \begin{cases} Q/L_k & si\ la\ fourmi\ k\ emprunte\ l'arc\ (i-j) \\ 0 & Sinon \end{cases} \qquad (1.3)$$

Lorsqu'une fourmi k doit faire le choix de la prochaine ville à visiter, elle calcule la probabilité de transition vers chaque ville potentielle $s \in N(s_p)$, notée P_{rs}, où r est la dernière ville visitée par la fourmi k. Cette probabilité est en fonction de la visibilité η_{ij} et la quantité de phéromone τ_{ij} déposée sur l'arc reliant ces deux villes.

Pour le problème TSP, la valeur de η_{ij} est donnée par $1/d_{rs}$ (d_{rs} est la distance entre la ville r et s). Cette information statique est utilisée pour guider le choix des fourmis vers des villes proches, afin d'éviter les villes trop lointaines. La règle de déplacement appelée règle aléatoire de transition est donnée par la formule suivante :

$$P_{rs} = \begin{cases} \frac{[\tau_{rs}]^{\alpha}\ [\eta_{rs}]^{\beta}}{\sum\limits_{w \notin tabou} [\tau_{rw}]^{\alpha}\ [\eta_{rw}]^{\beta}} j \notin tabou_k \\ 0 \qquad\qquad Sinon \end{cases} \tag{1.4}$$

- α, β : Deux paramètres qui contrôlent l'importance relative entre l'intensité de la trace de phéromone et la visibilité.
- $tabou_k$: Une mémoire tabou est associée à chaque fourmi k, afin d'éviter de re-sélectionner une ville déjà visitée.

Ce processus est réitéré jusqu'à aller à un nombre maximum de cycles $tmax$ ou en cas d'une stagnation de la recherche (toutes les fourmis suivent le même chemin et construisent la même solution à plusieurs reprises), il dénote une situation dans laquelle on doit arrêter la rechercher. l'algorithme donne en retour le meilleur chemin mémorisé.

1.3.4.2 Ant Colony System

L'Ant Colony System (ACS) a été introduit par Dorigo et Grambardella [18] pour améliorer les performances du premier algorithme sur des problèmes de grandes tailles, ACS est fondé sur une adaptation du AS : [20]

1. ACS introduit une règle de transition dépendant d'un paramètre $q_0 (0 \leq q_0 \leq 1)$, qui définit une balance diversification/intensification. Une fourmi k sur une ville i choisira une ville j selon la formule suivante :

$$j = \begin{cases} argmax_{w \notin tabou_k}[(\tau_{iw})\ (\eta_{iw})^{\beta}] \quad q \leq q_0 \\ J \qquad\qquad Sinon \end{cases} \tag{1.5}$$

   Ou q est une variable aléatoire uniformément distribuée sur $[0, 1]$ et J une ville sélectionnée aléatoirement selon la règle de transition de AS avec $\alpha = 1$. En fonction du paramètre q_0, il y a donc deux comportements possibles : Si $q > q_0$ le choix se fait de la même façon que pour l'algorithme AS, et le système tend à effectuer une diversification. Si $q \leq q_0$, le système tend au contraire vers une intensification.

2. La gestion des pistes est séparée en deux niveaux : une mise à jour locale et une mise à jour globale. Chaque fourmi dépose une piste de phéromone lors de la mise à jour locale selon la règle :

$$\tau_{ij}(t+1) = (1-\rho) \cdot \tau_{ij}(t) + \rho \cdot \tau_{ij}(t) \tag{1.6}$$

   Où τ_0 est la valeur initiale de la piste. A chaque passage, les arêtes visitées voient leur quantité de phéromone diminuée, ce qui favorise la diversification

par la prise en compte des trajets non explorés. A chaque itération, la mise à jour globale s'effectue comme ceci :

$$\tau_{ij}(t+1) = (1-\rho) \cdot \tau_{ij}(t) + \rho \cdot \tau_{ij}(t) \qquad (1.7)$$

Où les arêtes (i, j) appartiennent au meilleur tour de longueur L^* et où $\tau_{ij} = 1/L^*$. Ici, seule la meilleure piste est donc mise à jour (meilleure solution par itération ou meilleure solution globale), ce qui participe à une intensification par sélection de la meilleure solution.

3. Le système utilise une liste de candidats qui stocke pour chaque ville les v plus proches voisines, classées par distances croissantes. Une fourmi ne prendra en compte une arête vers une ville en dehors de la liste que si celle-ci a déjà été explorée. Concrètement, si toutes les villes ont déjà été visitées dans la liste de candidats, le choix se fera en fonction de la règle 1.5, sinon c'est la plus proche des villes non visitées qui sera choisie.

Une autre variante de ACS est l'hybridation entre le ACS et une méthode de recherche locale de type 3-opt [18]. Ici, la recherche locale est lancée pour améliorer les solutions trouvées par les fourmis (et donc les ramener à l'optimum local le plus proche).

1.3.4.3 Max-Min Ant System

Stùtzie et Hoos [49] introduisent le *Max-Min Ant System* dans lequel la force de la trace de phéromone est restreinte à un intervalle donné. Deux bornes inférieure (τ_{min}) et supérieure (τ_{max}) pour la trace sont établies en fonction de la taille de l'instance à résoudre. Le but de cette limitation est d'éviter la convergence trop rapide qui est observé lorsque la mise à jour est faite uniquement par la meilleure fourmi du cycle (best iteration). Cette méthode est considérée par ses auteurs comme meilleure que l'AS et ACS pour le problème TSP, l'algorithme a été amélioré avec succès par une phase de recherche locale pour chaque fourmi.

1.3.5 Applications aux problèmes NP-Difficile

Les algorithmes de colonies de fourmis sont abondamment étudiés depuis quelques années et il serait trop long de faire ici une liste exhaustive de toutes les applications et variantes qui ont été produites, OCF a été examiné sur probablement plus de cent problèmes NP-Difficile différents [17], dans les deux principaux champs d'application (problèmes NP-Difficile et problèmes dynamique), certains algorithmes ont cependant donné de très bon résultats. On peut notamment retenir des performances particulièrement intéressantes dans le cas de l'affectation quadratique, problèmes de planification, l'ordonnancement séquentiel et l'ordonnancement de voitures.

Cependant, une attention croissante a été donnée aux problèmes bien plus provocant tels que : l'optimisation multi-objectif, les modifications dynamiques des

données, et la nature stochastique de la fonction objectif et des contraintes de résolution. D'autres développements se concentrent sur la prolongation de l'applicabilité des algorithmes d'OCF de discret aux problèmes continus et à l'étude des réalisations parallèles des algorithmes OCF.

1.4 Problème de plus court chemin et systèmes de transport Multimodal

II a été souligné dans la section précédente que le monde réel offre un ensemble très divers de problèmes d'optimisation. Dans ce mémoire, nous nous intéressons plus particulièrement au problème de plus court chemin (PCC) dans un Réseau de Transport Multimodal (RTM). En premier lieu, la littérature consacrée à ce problème est exposée : La notion de multimodalité, les différentes classes de problèmes rencontrés dans le domaine de transport, La classification présentée se base sur les caractéristiques du problème de plus court chemin traité et du modèle réseau utilisé. Cette classification est suivie par une étude de complexité du problème traité. Dans un second temps, les différentes modélisations des réseaux de transport multimodal sont particulièrement détaillées, ainsi, on propose une implémentation objet pour le modèle retenu.

1.4.1 Multimodalité

Ces dernières décennies, la compétition de plus en plus forte entre les différentes compagnies de transport et l'augmentation de la demande de transport a fait émerger un nouveau concept : le Transport Multimodal, la Multimodalité consiste à assurer un transport d'un point origine à un point destination en empruntant successivement deux ou plusieurs modes de transport (bus, voiture, métro), chacun avec son horaire de départ, d'arrivée et du coût associé. La liaison entre ces modes est traduite par la présence de pôles d'échange ou de nœuds de correspondance, au niveau desquels il y a des échanges de voyageurs entre deux ou plusieurs modes de transport (figure 1.2). Celui-ci vise à rééquilibrer l'utilisation du réseau et des modes de transport exploités. Il associe de façon complémentaire la sécurité et l'efficacité d'un mode (par exemple, le train) à la souplesse d'un autre mode (par exemple, le bus). Il présente également des avantages importants pour la collectivité en constituant une bonne réponse aux problèmes de congestion, d'environnement et d'insécurité routière. Dans ce contexte, l'optimisation des déplacements se traduit par la recherche de la meilleure combinaison des modes qu'un usager doit emprunter pour réaliser un déplacement entre deux points, source et destination spécifiés.

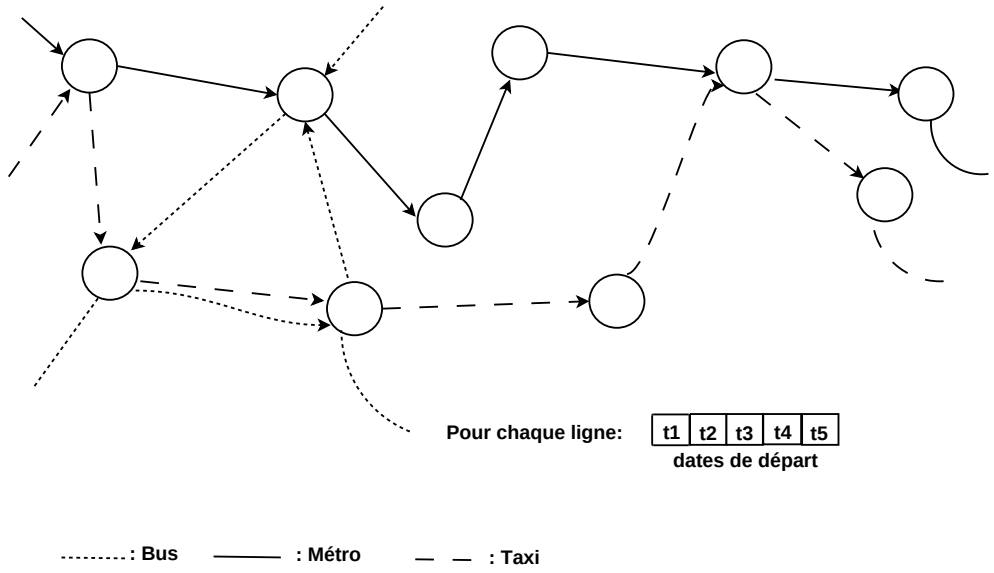

FIG. 1.2 – Exemple de reséaux multimodal.

1.4.2 Réseaux de transport et ses variantes

Les systèmes de transport sont caractérisés par le nombre de modes de transport qu'ils incluent et le type de données associées [7].

1. Classification selon le nombre de modes de transport

On distingue deux types de systèmes de transport : les systèmes unimodaux et les systèmes multimodaux. En effet, si le système comporte un seul mode de transport, il est dit unimodal. Dans le cas contraire, il est multimodal.

2. Classification selon le type des données

On parle des données statiques, dynamiques ou stochastiques.

- *Statiques :* la structure du réseau (stations, lignes) ainsi que ses paramètres (temps, dates de départ, coûts,..., etc.) sont fixes.
- *Dynamiques :* Les temps et les coûts des trajets sont fonction des dates de départ et des modes de transport choisis au niveau des stations. En plus, le caractère dynamique peut être représenter par la modification de la structure du réseau (stations, lignes, modes de transport) au cours du temps. L'aspect dynamique augmente fortement la complexité du problème.
- *Stochastiques :* si les paramètres (temps, coûts,..., etc.) et la structure du réseau (rupture d'un chemin) sont représentés par des fonctions probabilistes.

Notons que dans le cas dynamique il faut tenir compte l'aspect FIFO ou non du réseau. Le caractère FIFO est dû au fait que l'objectif d'arriver le plus tôt possible

à une station destination ne se réalise pas forcément en partant le plus tôt possible d'une station de départ.

1.4.3 Le problème de plus court chemin et ses variantes

L'évaluation d'un réseau de transport est réalisée en déterminant les caractéristiques des meilleurs itinéraires à emprunter pour réaliser un déplacement entre des points de départ et des points destinations spécifiés (problème de plus court chemin). Deux classes de caractéristiques principales permettent de classer les travaux identifiés dans la littérature sur ce problème combinatoire [7]. Elles sont relatives aux points de départ et d'arrivée du déplacement à optimiser et aux objectifs visés.

1. Classification selon les points de départ et d'arrivée

La recherche du plus court chemin diffère selon le nombre de points de départ (source) et de points d'arrivée (destination) sélectionnés. Elle peut être effectuée :

- *one-to-one shortest path :* d'un point source unique à un point destination unique lui aussi.
- *one-to-all shortest path :* d'un point à tout autre point du réseau.
- *all-to-one shortest path :* d'un ensemble de points sources à un point destination spécifique.
- *all-to-all shortest path* : entre toutes les paires de nœuds dans le réseau.

2. Classification selon les objectifs de résolution

chercher le chemin optimum revient à chercher l'itinéraire qui minimise ou maximise une fonction coût spécifique, on parle alors de chemin le plus rapide, le moins cher ou le plus court selon que l'objectif est la minimisation du temps, de coût ou de la distance, dans le cas de la maximisation (circuit touristique par exemple), on parle de chemin le plus long, le plus cher ou le plus lent, selon le nombre de critères choisis, on parle de *mono-objectif* si on considère un seul objectif, *bi-objectif* dans le cas de deux objectifs ou *multi-objectif* dans le cas où le nombre d'objectifs considéré est supérieur à deux.

A partir de ces classifications, on peut dire que l'objectif de nos travaux de recherche est de proposer une approche capable de résoudre le problème de plus court chemin (one-to-one) Multi-objectif dans un système de transport Multimodal (Dynamique, Stochastique et Non FIFO).

1.4.4 Etude de Complexité

Dans la littérature, une attention particulière est porté sur les problèmes de plus court chemin statique mono-objectif en utilisant les algorithmes exacts [4, 14, 22]. Cependant, même un algorithme exact peut nécessiter un temps de calcul très important dans le cas des graphes denses. C'est le cas par exemple des réseaux multimodaux qui contient beaucoup de nœuds de correspondance. Pour un réseau unimodal dynamique et Non FIFO, la variante mono-objectif est déjà NP-Difficile [44]. Ceci suffit amplement à justifier que la problématique étudiée dans ce mémoire appartienne à la classe NP-difficile.

En pratique, les problèmes de plus court chemin réels sont rarement mono-objectifs, il y a généralement plusieurs critères contradictoires à satisfaire simultanément. Ces problèmes d'optimisation ont la particularité d'être beaucoup plus difficiles à traiter que leur équivalent mono-objectif. La difficulté principale est qu'il n'existe pas une seule solution optimale, mais une population de solutions.

En conséquence, l'application considérée ici correspond à une variante plus complexe du problème de plus court chemin. L'utilisation d'une méthode exacte est impossible, Ce qui motive a priori l'utilisation des approches de résolution approchée. A partir de notre recherche bibliographique, nous constatons que ces outils d'optimisation ont été peu utilisés dans le domaine de transport multimodal. Dans ce cadre, Un travail plus récent sur le problème de plus court chemin intermodal est celui de boussejra et al [8]. C'est le seul travail le plus proche de notre. Les contributions développées dans ce papier portent sur deux algorithmes qui utilisent les mêmes opérateurs génétiques (croissement, mutation) et ne se distinguent que par la phase d'évaluation des individus, un processus de diversification est proposé pour assurer la variété des solutions. D'autres travaux sont consacrés au traitement en temps réel des requêtes et la gestion de perturbations [54], [53], [11].

1.4.5 Modélisation des réseaux de transport multimodal

Comme nous avons déjà vu dans la section 1.3.3, L'utilisation d'une modélisation graphique plus adaptée facilite naturellement le développement d'algorithme OCF pour une résolution efficace.

La modélisation des réseaux de transport fait appel, dans la majorité des cas, à la théorie des graphes. Il est facile d'établir une correspondance entre ces derniers et les systèmes de transport. Les stations peuvent être représentées par les nœuds (sommets) et les segments de route liant deux stations successives par les arcs. Les quatre représentations les plus utilisées pour le système de transport multimodal sont le modèle *classique* [24], le modèle à base d'*hypergraphes* [41], le modèle *espace-temps* [46], et le modèle *multi-valué* [55, 9].

1.4.5.1 Modèle classique

Un réseau de transport multimodal est considéré comme une généralisation d'un réseau de transport unimodal. On peut passer d'une représentation unimodal vers une représentation multimodal dans laquelle le modèle doit faire apparaître les différents modes de transport et les dates de départ existants et doit permettre aussi de représenter leurs correspondances. Par conséquent, plusieurs nouveaux éléments (arcs, nœuds) doivent être ajoutés dans le graphe de modélisation.

$G(V, E)$ unimodal vers $G'(V', E')$ multimodal :

Pour chaque mode de transport k :
1. Pour chaque station v, nous ajoutons un nouveau nœud v_k dans V',
2. Remplacer chaque arc (u, v) par trois nouveaux arcs dans E' : (u, u_k), (u_k, v_k) et (v_k, v).

- Le nœud v_k s'appelle le point d'accès (access point) pour le mode k à la station v.
- L'arc (u_k, v_k) s'appelle un arc de déplacement et représente le déplacement de la station u à la station v par le mode de transport k.

Les deux arcs (v_k, v), (u, u_k) sont introduits pour assurer le changement de mode de transport sur les deux sommets u et v :

- L'arc (v_k, v) s'appelle un arc de descente (*alighting edge*) pour représenter le descendant à la station v du mode de de transport k.
- L'arc (u, u_k) s'appelle un arc d'attente (*waiting edge*) pour représenter l'attente de mode de de transport k à la station u.

Le même processus de changement est appliqué pour chaque date de départ t d'un mode de transport k.

Bien que ce modèle soit adapté, il reste limité à des réseaux de taille réduite. Par exemple, pour un système de transport multimodal avec p stations, m modes de transports, et de t date de départ pour chaque mode. La taille de graphe augmente avec $(p \cdot m \cdot t)$ nœud, et $(p^2 \cdot m \cdot t)$ arcs de déplacement, et $(p \cdot m \cdot t)$ arc pour représenter le changement de modes de transport.

1.4.5.2 Modèle hypergraphe

Les hypergraphes généralisent la notion de graphe dans le sens où les arêtes relient un nombre quelconque de sommets (compris entre un et le nombre de sommets de l'hypergraphe).

Un hypergraphe est défini par un graphe $H_G = (H_N, H_A)$ où $H_N = \{n_1, \ldots, n_m\}$ est l'ensemble des nœuds et $H_A = \{a_1, \ldots, a_y\}$ est l'ensemble des hyper-arcs. Un hyper-arc $a_i \in A$ est défini par un couple $(T(a_i), H(a_i))$ où l'ensemble $T(ai) \in N$ représente la base de l'hyper-arc, et l'ensemble $H(a_i) \in N/T(a_i)$ représente sa tête.

Ainsi, si pour chaque $a_i \in A$, $T(a_i) = 1$ et $H(a_i) = 1$, l'hypergraphe est un graphe simple [7].

Lozano et Storchi [41] Modélisent un réseau multimodal par un hypergraphe dans lequel, ils associes pour chaque mode de transport un hyper-arc de sorte que la base u d'un hyper-arc $a = (u, h(a))$ représente le nœud commun aux lignes du même mode. Les successeurs de ce nœud composent la tête $h(a) = \{v_1, v_2, \ldots, v_k\}$ de l'hyper-arc. La modélisation d'hypergraphe est bien adaptée à la représentation de changement de modes de transport, elle permet d'orienter le choix de transport vers le mode de déplacement le mieux adapté selon la préférence de passager (trouver le plus court chemin viable).

1.4.5.3 Modèle espace temps

Cette représentation a été développé pour les réseaux de transport dynamiques discrets avec la prise en compte des temps d'attentes. Les valeurs des dates de départ associées aux nœuds varient dans un ensemble discret $T = \{t_1, \ldots, t_q\}$ de cardinalité q. Pour chaque date t_i, un temps de trajet $\upsilon_{ij}(t)$ est attribué à chaque arc (i, j). De ce fait, les dates d'arrivée possibles sur les nœuds destination sont déterminées par $t_i + d_{ij}(t_i)$.

Le plus court chemin dynamique dans un modèle discret peut être efficacement étudié et résolu en présentant le réseau d'Espace-Temps $R = (V, E)$ ainsi défini :

- $V = \{i_h : i \in N, 1 \leq h \leq q\}$
- $E = \{(i_h, j_k) : (i, j) \in A, t_h + d_{ij}(t_h) = t_k, 1 \leq h < k \leq q\}$

Afin de modéliser la station d'attente, de nouveaux arcs (d'attentes) sont ajoutés dans E de la façon suivante : $(i_h, i_h + 1)$.

Comme un exemple simple, soit le réseau dynamique à trois nœuds de la figure 1.3. les ensembles *date* avec $1 \leq |date| \leq 4$ regroupent les valeurs des temps de trajet associées aux périodes de temps définies par l'ensemble $T = \{t_1, t_2, t_3, t_4\}$.

Cette modélisation est bien adapté au cas des réseaux dynamiques, mais, elle reste limité à des réseaux de taille réduite. Pour un réseau de transport multimodal avec n nœuds, m arcs, x modes et t intervalles de temps, le réseau résultant du modèle espace-temps est de taille $(n \cdot t \cdot x)$ nœuds et un nombre d'arcs $\leq (m + n) \cdot q \cdot t$.

1.4.5.4 Modèle multi-valué (le modèle retenu)

Le réseau de transport est représenté par un graphe $G = (N, A, X, T)$ dans lequel : $N = \{n_1, n_2, \ldots, n_m\}$ est l'ensemble des nœuds représentant les stations du réseau. A représente les arcs modélisant les tronçons de routes. $X = \cup x(n_i)$ définit tous les modes de transport desservant les stations du réseau et finalement $T = \cup t[x(n_i)]$ est l'ensemble des dates de départ associées aux modes de transport de toutes les stations du réseau. La différence entre une date de départ et une date

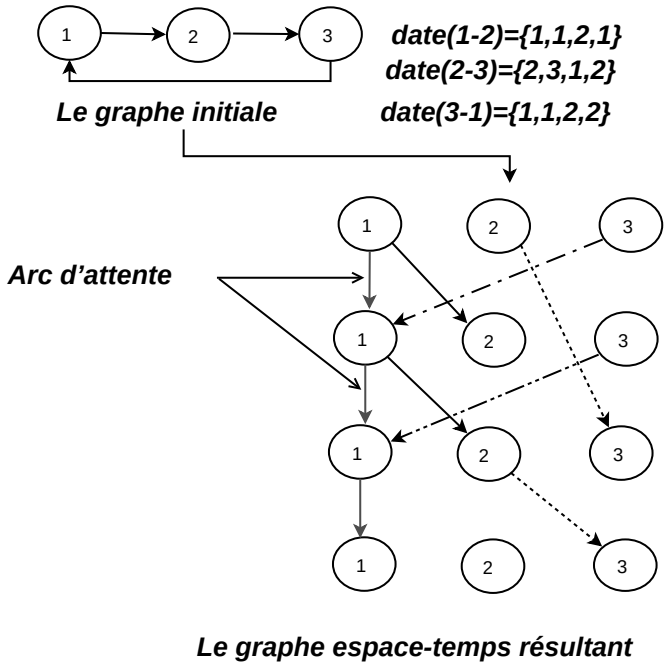

FIG. 1.3 – Exemple de réseau espace temps

d'arrivée sur un nœud quelconque du graphe définit la valeur du temps d'attente sur ce nœud, le temps nécessaire pour effectuer un changement de mode sur ce nœud est aussi considéré comme un temps d'attente. La valeur de ce temps sur le nœud destination D est égale à zéro. Elle est également nulle si le mode d'arrivée à un nœud donné et celui de départ sont identiques (pas de changement de mode). Parfois, en cas de contrainte de départ sur S, cette valeur est représentée par la différence entre la date de départ t choisie pour quitter S et la date de départ td_s souhaitée par le passager [9].

Pour chaque couple (x, t) tel que $x \in X[v]$ et $t \in T[x]$, et selon le nombre d'objectifs, plusieurs valeurs (temps, coût,..., etc.) sont assignées, par exemple $v_{ij(x,t)}$ définit le temps nécessaire pour aller du nœud i vers le nœud j avec un mode de transport x en quittant i à une date t. Donc, chaque arc du graphe porte autant d'étiquettes que de modes de transport et de dates de départ possibles.

1.4.6 Implémentation du modèle retenu

L'utilisation d'une modélisation plus adaptée facilite naturellement le développement d'algorithmes OCF de résolution efficace. Mais l'implémentation du modèle influence tout autant les performances.

Une implémentation du réseau de transport multimodal proposée dans [7] est basée sur des listes chaînées. Les nœuds du graphe sont groupés dans une structure de liste. Chaque nœud est utilisé comme un point d'accès à une structure de liste qui

regroupe l'ensemble de ses successeurs. Les éléments de la liste sont les paramètres associés au graphe : 1) les modes de transport pour lier le nœud base (point d'accès) avec ses successeurs, 2) les dates de départ associées à chaque mode, 3) les valeurs des temps et coût de trajet pour chaque couple (mode, date), ces paramètres sont stockés dans une matrice dont les colonnes sont les modes et les lignes sont les dates de départ. Les éléments situés à l'intersection d'une ligne et d'une colonne fournissent le temps et le coût de trajet pour le couple (mode, date) correspondant. Ce modèle est de taille importante, sa représentation sous forme matricielle, risquant de comporter de trop nombreux éléments nuls.

```
Classe Graphe {
               Vecteur nd (N) : Noeud;
              }

Classe Noeud  {
                Num_nd;
                Vecteur suc (M) : Arc ;
              }

Classe Arc    {
                Num_nd_suc;
                Vecteur mode(X) : Mode ;
              }

Classe Mode   {
                Nom_Mode;
                Vecteur date (Y) : Date ;
              }

Classe Date   {
                Date_Depart;
                Coût;
                Temps;
                Autres paramètres;
              }
```

FIG. 1.4 – Implémentation objet du modèle multi-valué

Notre implémentation se base sur une conception orientée objet, dans laquelle les structures de données utilisées sont représentées par des objets. On peut définir le modèle multi-valué $G(N, A, X, T)$ par une description de composition entre les éléments N, A, X, et T. Un graphe est constitué de nœuds, dont certains sont reliés par des arêtes avec un ensemble de nœuds successeurs, et pour chaque arc un ou plusieurs modes de transport sont associés, et finalement un ensemble de dates de départ est défini pour chaque mode de transport dans lesquels plusieurs

valeurs (temps, coût,. . . , etc.) sont assignées. On se base sur cette description, notre implémentation prend la forme d'une suite de définitions de cinq classes décrites dans la figure 1.4.

Pour notre part, nous ne s'intéressons pas à l'information multimodal, c'est un autre axe de recherche, nous nous contentons dans cette implémentation de présenter les bases nécessaires à la compréhension des approches de résolution proposées dans les chapitres qui suivent.

1.5 Conclusion

Dans ce chapitre, les concepts clés, la formalisation et les principaux efforts de la communauté scientifique relatifs à notre sujet de mémoire y ont été présentés.

Nous avons fait le choix de traiter le problème de transport multimodal. Ce problème n'est pas simple et ne peut être traité dans sa globalité dans le cadre de ce magistère. L'étude bibliographique a montré que la résolution de ce problème constitue un axe de recherche peu exploré à ce jour. Nous avons orienté alors nos travaux vers l'optimisation multi-objectif du transport multimodal. L'utilisation de colonie de fourmis ayant fait l'objet de rares travaux dans le cadre de cette application, nous avons alors opté pour proposer des algorithmes de résolution basé sur cette métaheuristique. Pour finir, nous avons proposé une implémentation objet pour le modèle adopté.

Chapitre 2

OCF mono-objectif appliquée au réseau de transport multimodal

2.1 Introduction

L'évaluation d'un réseau de transport est réalisée en déterminant les caractéristiques des meilleurs itinéraires à emprunter pour réaliser un déplacement entre un point de départ et un point de destination spécifiés. Ceci revient à résoudre une variante particulière du problème de plus court chemin, qui est l'un des problèmes les plus classiques d'optimisation. Comme nous avons vu précédemment, la complexité des réseaux multimodaux ne cesse d'augmenter ; le nombre de stations, de pôles d'échanges et de lignes de transport sont de plus en plus importants, A cette complexité, s'ajoute le caractère dynamique des réseaux de transport. En conséquence, Il est nécessaire d'utiliser les méthodes de résolutions approchées, notre choix s'est porté sur les algorithmes de colonies de fourmis, ces derniers sont reconnus et appréciés pour leur capacité à fournir des solutions de très bonne qualité avec des temps de calculs raisonnables.

Ce chapitre débute par la formulation mathématique du système de transport étudié (section 2). Puis, nous proposons deux méthodes monocritères (OCF1-PCC-RTM, OCF2-PCC-RTM) adaptées à la résolution du plus court chemin multimodal (section 3 et 4). Vu la supériorité de la qualité des solutions de l'OCF2-PCC-RTM, nous l'avons adapté pour l'optimisation du temps de transport qui nécessite une résolution bi-objectif (temps d'attente, temps du trajet), deux variantes sont proposées selon la définition de l'information heuristique (section 5). Enfin, les résultats des tests effectués pour comparer ces différentes versions sont présentés dans la section 6.

Le travail présenté dans ce chapitre a fait l'objet d'une publication scientifique présentée à la conférence internationale META08 [50]

2.2 Formulation mathématique du problème

La recherche de plus court chemin est formulée sous forme d'une requête qui spécifie :

- Le point source : S
- Le point destination : D
- la date de départ du nœud S : td_s

Une solution du problème est représentée par une suite de nœuds caractérisés individuellement par un triplet $(noeud, mode, date)$:

$$\Pi_{SD} = \{(n_1, x_1, t_1), (n_2, x_2, t_2), \ldots, (n_{f_1}, x_{f-1}, t_{f-1}), (n_f = D)\} \qquad (2.1)$$

Le triplet (n_i, x_i, t_i) signifie un déplacement du nœud n_i vers le nœud n_{i+1} par le mode de transport x en quittant n_i à la date t_i.

L'algorithme d'optimisation proposé s'appuie sur la définition et l'évaluation d'un certain nombre de variables :

- le temps total de transport : T_{SD}.
- les valeurs de temps du trajet sur les arcs (n_{i-1}, n_i) en utilisant le mode x_{i-1} à la date t_{i-1} : $\upsilon_{n_{i-1}n_i(x_{i-1},t_{i-1})}$.
- Les dates d'arrivée au plus tôt sur les nœuds n_i : dr_{n_i}.
- le temps d'attente sur chaque nœud n_i : θ_{ni}.
- le coût total de transport : C_{SD}.
- les valeurs de coût du trajet sur les arcs (n_{i-1}, n_i) en utilisant le mode x_{i-1} à la date t_{i-1} : $\phi_{n_{i-1}n_i}(x_{i-1}, t_i)$.

La détermination des valeurs de ces variables, sur le chemin Π_{SD}, est effectuée comme suit :

- La date d'arrivée sur le nœud n_i est égale à la somme de la date de départ t_{i-1} du prédécesseur n_{i-1} et de la valeur du temps de trajet associée à l'arc (n_{i-1}, n_i) en utilisant le mode x_{i-1} :
$$dr_{ni} = t_{i-1} + \upsilon_{n_{i-1}n_i(x_{i-1},t_{i-1})} \qquad (2.2)$$
- La valeur du temps d'attente sur le nœud n_i est égale à la différence entre la date de départ t_i et la date d'arrivée dr_{n_i} :
$$\theta_{ni}(t_{i-1}, t_i) = t_i - dr_{n_i} \qquad (2.3)$$
- Le temps total de transport : c'est la somme des temps de trajets et d'attentes sur Π_{SD} qui est égal à la différence entre la date d'arrivée sur le nœud D et la date de départ du nœud S.
$$T_{SD} = dr_D - t_S \qquad (2.4)$$
- Le coût total de transport :
$$C_{SD} = \sum_{n_i, x_i, t_i \in \Pi} \phi_{n_i n_{i+1}}(x_i, t_i) \qquad (2.5)$$

2.3 OCF pour le problème de Plus Court Chemin Mono-critère dans un Réseau de Transport Multimodal (OCF1-PCC-RTM)

Cette section explique comment l'optimisation par colonie de fourmis a été adaptée pour le problème de plus court chemin mono-critère dans un système de transport multimodal en utilisant la modélisation graphique proposée dans le chapitre précédent.

L'algorithme de base *Ant System* pour le problème de voyageurs de commerce constitue l'algorithme de départ qui sera amélioré par une autre variante (modification par incrément de cette algorithme de départ, plus précisément, on s'appuie sur le *Max-Min Ant System* qui fournit un meilleur compromis entre l'exploitation et l'exploration de l'espace de recherche)

2.3.1 Choix des critères d'optimisation

Dans tous les cas, trouver le chemin optimum revient à chercher l'itinéraire qui minimise ou qui maximise une fonction spécifique. Cette recherche est définie en fonction des objectifs considérés. Les deux objectifs choisis pour l'illustration des résultats sont la minimisation du coût et du temps de transport. Ce choix s'est appuyé sur les raisons suivantes : La durée du trajet est reconnue comme étant l'un des critères qui influencent le plus, explicitement ou non, les choix des usagers. Le coût a lui aussi une importance non négligeable, en particulier pour des trajets à longue distance [37]. L'optimisation du temps nécessite une résolution bi-objectif, elle sera abordée dans la section 5.

2.3.2 Description de l'algorithme OCF1-PCC-RTM

Le pseudo-code de l'OCF1-PCC-RTM est montré dans l'algorithme 2.1, il est constitué de trois phases (initialisation, construction de solutions et finalement la mise à jour de phéromone).

Avant d'écrire l'algorithme, deux éléments essentiels doivent être ajoutés :

1. **Mémoire de la fourmi** : la mémoire $tabou_k(t)$ d'une fourmi k à l'instant t est une simple liste dynamique qui retient par quelles villes la fourmi est déjà passé au cours du cycle courant, y compris la ville sur laquelle est placée. En plus, cette liste permet de sauvegarder la solution courante retenue, et d'effectuer un retour arrière en cas d'échec du deplacement qui sera expliqué dans la suite de cette section.

2. **Mémoire de phéromone** $\tau_{ij(x,t)}$: représente la quantité de phéromone déposée pour aller du nœud i vers le nœud j par un mode de transport x en quittant i à une date t. Pour représenter cette information, il suffit d'ajouter une variable réelle appelée *phéromone* dans la classe *Date*.

```
Classe Date   {
                Date_Depart
                Coût
                Temps
                Phéromone
              }
```

L'information $\tau_{ij(x,t)}$ est représentée dans le modèle objet proposé dans le chapitre précédent par : $nd[i].suc[j].mode[x].date[t].Pheromone$

2.3.2.1 Initialisation

Cette phase consiste tout d'abord à initialiser les paramètres spécifiques à l'OCF tel que le nombre de fourmis, le nombre maximal de cycles, le paramètre d'évaporation de la trace de phéromone et les paramètres de la règle de transition.

Les pistes de phéromone sont initialisées comme suit : $\tau_{ij(x,t)} \longleftarrow p$, où p est une petite constante positive, qui ne peut être nulle (sinon, il y a un problème lors du calcul de probabilité de transition quand $t = 0$).

2.3.2.2 Déroulement d'une itération

Après l'étape d'initialisation, voici le déroulement d'une itération. Elle s'effectue en deux étapes : une étape d'initialisation et une autre sur le choix des transitions.

Etape d'initialisation : Le parcours du graphe pour trouver le plus court chemin entre la source S et la destination D peut s'effectuer de deux façons : un parcours en avant qui commence du point source S ou alors un parcours en sens inverse qui débute du point destination D.

pour diversifier l'espace de recherche, une exploration bidirectionnelle du graphe à partir des deux points S et D est utilisée tel qu'on divise les fourmis en deux ensembles, un ensemble cherche le plus court chemin de la source vers la destination et l'autre de la destination vers la source. Cette différentiation entre les fourmis se fait principalement par une fonction probabiliste (les fourmis sont réparties aléatoirement sur les deux villes S et D).

Si le point de départ est S, l'algorithme progresse dans le graphe en manipulant, à chaque étape de la résolution, les nœuds successeurs du nœud courant (déjà sauvegarder). Dans le cas contraire, les nœuds manipulés sont les prédécesseurs, ce

Algorithm 2.1 OCF1-PCC-RTM

Main()
Début
Initialiser la trace de phéromone à p
tantque $(t < Ncmax)\&\&(\neg Stagnation)$ **faire**
  pour $k \longleftarrow 1, \ldots, m$ **faire**
    ville-départ$\longleftarrow$fourmis-initialisation()
    construction-solution(ville-départ)
  fin pour
  update-phéromone()
fin tantque
Fin

Procédure construction-solutions(ville-départ)
Début
$i \longleftarrow$ville-départ
tantque destination non atteindre **faire**
  si (aucune transition n'est possible) **alors**
    $i \longleftarrow tabou[tabou.size - 1].noeud$ //retour arrière
  sinon
    calculer la transistion $[j, x, t]$ selon la formule 2.6
    $tabou_k \leftarrow (tabou_k, [j, x, t])$
    $i \longleftarrow j$
  finsi
fin tantque
$\Delta^k \leftarrow C_{SD}(tabou_k)$
Fin

Procédure update-phéromone()
Début
pour chaque arête $(i, j) \in A$ **faire**
  pour chaque mode $x \in X$ **faire**
    pour chaque date $t \in T$ **faire**
      $\tau_{ij(x,t)} \longleftarrow \rho \cdot \tau_{ij(x,t)}$
    fin pour
  fin pour
fin pour
pour $k = 1, \ldots, m$ **faire**
  pour $(i, j, x, t) \in tabou_k$ **faire**
    $\tau_{ij(x,t)} \longleftarrow \tau_{ij(x,t)+\Delta^k}$
  fin pour
fin pour
Fin

Fonction formis-initialisation(Fourmis k)
Début
Vider la liste $tabou_k$
Générer une variable aléatoire Ψ uniformément distribuée sur [0,1]
si $(\Psi > 0,5)$ **alors** Retourner S **sinon** Retourner D
Fin

qui nécessite de mémoriser en plus de nœuds successeurs, un ensemble de prédécesseurs pour chaque nœud, ce qui augmente deux fois l'espace mémoire nécessaire (inconvénient en terme d'espace pour des réseaux de taille importante).

Choix des transitions : La construction de solution progresse tant que la fourmi n'arrive pas à la ville destination(S ou D, selon la ville de départ choisie). La transition du nœud courrant i vers le nœud j par un mode de transport x en quittant i à une date t, dépend de :

1. La quantité de phéromone déposée $\tau_{ij(x,t)}$
2. La visibilité $\eta_{ij(x,j)} = \frac{1}{\phi_{ij(x,t)}}$

La règle de déplacement (appelée règle aléatoire de transition) consiste à choisir à la fois le successeur de nœud courant i à visiter, et le mode de transport x à utiliser, et la date de départ t correspondante :

$$P_{ij(x,t)} = \begin{cases} \frac{(\tau_{ij(x,t)})^{\alpha}(\eta_{ij(x,t)})^{\beta}}{\sum_{k \notin tabu} (\tau_{ik\,\forall(w,l)})^{\alpha}(\eta_{ik\,\forall(w,l)})^{\beta}} & j \notin tabou \ et \ j \succ i \\ 0 & sinon \end{cases} \tag{2.6}$$

Remarque :

- Les paramètres α et β sont utilisés afin de déterminer l'importance relative de l'intensité de la trace et de la visibilité de choix de transition.
- Si la transition calculée par la fourmi k est effectuée sur le nœud i vers le nœud j par x à une date t, on ajoute le triplet (j, x, t) à $tabou_k$.
- Un cycle est complété au moment où la dernière des m fourmis a terminé sa construction.

Echec de deplacement : En pratique, un réseau de transport multimodal est un graphe connexe non complet, selon la règle de transition (formule 2.6), une fourmi peut avoir une situation de blocage pendant la recherche d'une solution.

Définition 2. *Une fourmi est en blocage sur une ville j, si aucune transition vers l'un de ses successeurs n'est possible (figure 2.1).*

Les deux conditions de survenue d'un blocage sur une ville j sont :

1. l'ensemble de successeurs de $j \in tabou$
2. n'existe plus une date de départ planifiée sur la ville $j \geq dr_j$.

Pour faire face à ce problème, nous proposons d'effectuer un retour arrière. Selon le schéma précèdent, il faut retourner au nœud i pour trouver un autre chemin,

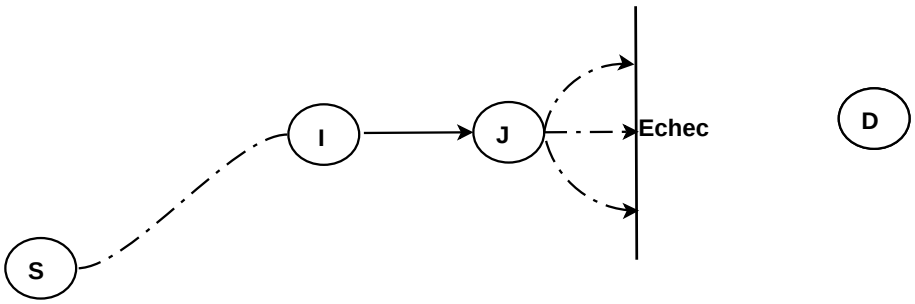

FIG. 2.1 – Echec du deplacement.

en évitant de visiter une autre fois le nœud j. En effet, il suffit de connaître le prédécesseur du nœud de blocage, cette information est déjà sauvegardée dans la liste *tabou*. On peut facilement réduire le nombre d'échec de recherche, en empêchant l'apparition de la deuxième condition de leur existence qui est très fréquente par rapport à la première. La prévention consiste à attribuer une borne maximale pour l'ensemble de dates de départ planifié sur chaque nœud ($Dmax$). Dans ce cas, La transition du nœud courrant i vers le nœud j par un mode de transport x en quittant i à une date t, doit vérifier la condition suivante :

$$j \notin tabou \;\; et \;\; j \succ i \;\; et \, dr_i \leq Dmax(j) \qquad (2.7)$$

2.3.2.3 La mise à jour de phéromone

l'adaptation du problème est réalisée par le *Ant-System*, dont chaque fourmi laisse une certaine quantité de phéromone sur l'ensemble de son parcours. La mise à jour de la trace se fait d'une manière globale, à la fin de chaque cycle et selon la formule 2.8, où γ est le nombre de fourmis élitistes.

$$\tau_{ij(x,t)}(t+1) = \rho \cdot \tau_{ij(x,t)}(t) + \sum_{k=1}^{m} \Delta^k_{ij(x,t)} + \gamma \cdot \Delta^*_{ij(x,t)} \qquad (2.8)$$

$\Delta^k_{ij(x,t)}$ est la quantité de phéromone laissée par une fourmi k sur l'arc (i,j), par le mode de transport x, en quittant i à une date t. Selon la formule 2.1, une solution du problème est représentée par une suite de nœuds caractérisés individuellement par un triplet $(noeud, mode, date)$. Appelons Π^k le tour réalisé par la k^{eme} fourmi dans l'intervalle de temps $[t, t+1]$. Π^k peut s'obtenir en analysant la mémoire de la fourmi k à l'instant t. Chaque chemin est caractérisé par sa longueur représentée par l'évolution de la variable coût C_{SD}. La quantité déposée est bien fonction de la qualité de la solution trouvée : plus le tour est court, plus les arcs qui composent ce tour sont approvisionnés, ce qui permet de diriger la recherche vers les bonnes solutions. On définit alors $\Delta^k_{ij(x,t)}$ comme suite :

$$\Delta^k_{ij(x,t)} = 1/C_{SD}(\Pi^k) \qquad (2.9)$$

La fourmi élitiste effectue une mise à jour supplémentaire de la trace selon le principe d'élitisme [19], c'est-à-dire que la meilleure solution Π^* trouvée par l'algorithme depuis le début de son exécution est utilisée pour renforcer d'avantage la trace de phéromone par une quantité $\Delta^*_{ij(x,t)}(2.10)$, ce qui permet d'accroître la convergence de l'algorithme.

$$\Delta^*_{ij(x,t)} = \begin{cases} \frac{1}{C_{SD}(\Pi^*)} & (i,j,x,t) \in \Pi^* \\ 0 & sinon \end{cases} \qquad (2.10)$$

2.3.2.4 Fin de l'algorithme

L'algorithme progresse jusqu'à atteindre un nombre maximum des cycles *tmax* ou en cas d'une stagnation de la recherche (toutes les fourmis suivent le même chemin et construisent la même solution à plusieurs reprises), il dénote une situation dans laquelle on doit arrêter la recherche. Un test de stagnation a été introduit dans notre approche. Il consiste à calculer à chaque itération l'écart type σ de la longueur de tours des fourmis, si σ est plus proche de zéro, on considère qu'il y a une situation de stagnation et on arrête l'algorithme.

2.4 Version améliorée(OCF2-PCC-RTM)

L'amélioration de l'algorithme précédent est basée sur le *MAX-MIN Ant System* [49]. Cette méthode est considérée par ses auteurs comme meilleur que *l'Ant System* pour le problème de voyageur de commerce et le problème d'affectation quadratique. Ils mentionnent qu'une amélioration de la performance pourrait être rencontrée dans le cadre d'autres problèmes.

OCF2-PCC-RTM est fondé sur des modifications du OCF1-PCC-RTM :

1. La trace de phéromone est limitée à un intervalle $[\tau_{min}, \tau_{max}]$, afin de garantir une bonne exploration de l'espace de recherche, et prévenir la colonie de fourmis de stagner sur une solution sub-optimale.
2. Les pistes sont initialisées à leur valeur maximale τ_{max}, afin de garantir une exploration plus large de l'espace de recherche durant les premiers cycles.
3. Les phéromones sont mises à jour uniquement sur les pistes empruntées par la meilleure fourmi du cycle. La mise à jour doit assurer que la trace de phéromone respecte les limites $[\tau_{min}, \tau_{max}]$. Formellement, cette opération s'effectue en deux étapes selon la formule suivante :

$$(I)\begin{cases} \forall(i,j)\in A\ \tau_{ij\forall(x,t)} = \rho \times \tau_{ij\forall(x,t)} \\ \tau_{ij(x,t)} = \tau_{min}\ si\ \tau_{ij(x,t)} < \tau_{min} \end{cases} \tag{2.11}$$

$$(II)\begin{cases} \forall(i,j,x,t)\in \Pi^{*}\tau_{ij(x,t)} = \ \tau_{ij(x,t)} + 1/C_{SD}(\Pi^{\star}) \\ \tau_{ij(x,t)} = \tau_{max}\ si\ \tau_{ij(x,t)} > \tau_{max} \end{cases}$$

2.5 Minimisation du temps de transport

Le temps du trajet change de façon remarquable d'un mode à un autre, d'une date de départ à une autre, alors ce critère est un élément essentiel pour le choix modal des usagers, dans lequel le client souhaite de minimiser le temps de transport entre deux points source et destination. Comme nous l'avons déjà signalé dans le chapitre précédent (section 4.4), le caractère Non-FIFO du système de transport augmente la complexité du problème. En effet, l'objectif d'arriver le plus tôt possible à un nœud ne se réalise pas forcement en partant le plus tôt possible d'un nœud de départ. Dans ce cas, le temps d'attente intervient dans le calcul du temps de transport, La minimisation de ce dernier dépend de la représentation de l'information heuristique, deux variantes sont proposées :

Variante mono-objectif OCF2-PCC-RTM1(temps) : Le choix de transition d'une fourmi k sur une ville i se fait selon le principe suivant : Le triplet (j,x,t) sélectionné par la règle de transition correspond ici à celui qui, parmi les choix possibles, présente la valeur minimale de la somme du temps de trajet et des temps d'attentes. En effet, la visibilité $\eta_{ij(x,t)}$ utilisée dans la règle de transition devient comme suit :

$$\eta_{ij(x,j)} = \frac{1}{v_{ij(x,t)}+\theta(i)} \tag{2.12}$$

Variante bi-objectif OCF2-PCC-RTM2(temps) : Dans cette variante, l'optimisation du temps consiste à minimiser à la fois, le temps du trajet, et les temps d'attentes, pour tenir de compte ces deux critères, la probabilité de transition décrite dans la formule 2.9 devient comme suit :

$$P_{ij(x,t)} = \begin{cases} \dfrac{(\tau_{ij(x,t)})^{\alpha}(\frac{1}{v_{ij(x,t)}})^{\beta}(\frac{1}{\theta(i)})^{\delta}}{\sum\limits_{k\notin tabu}(\tau_{ik\,\forall(w,l)})^{\alpha}(\frac{1}{v_{ik\,\forall(w,l)}})^{\beta}(\frac{1}{\theta(i)})^{\delta}} & cond \\ 0 & Sinon \end{cases} \tag{2.13}$$

La variable *cond* représente la condition de transition définie dans la formule 2.7.

2.6 Résultats expérimentaux

2.6.1 Protocole de test

Les réseaux de transport sur lesquels des tests ont été effectués sont crées en exploitant un générateur aléatoire que nous avons développé. Ce générateur est conçu afin de fournir des graphes similaires au réseau de transport multimodal en se basent sur la modélisation multi-valué décrite dans le chapitre 1.

Les tests sont réalisés sur des graphes de 30, 40, 60, 100, 300, et du 600 nœuds. le nombre de modes disponibles est fixé à trois, Afin d'augmenter la complexité des problèmes, nous supposons un nombre important de dates de départ possibles pour l'ensemble des nœuds de réseau. Treize problèmes (P1,. . . ,P13) sont construits selon les points de départ et destination sélectionnés dans ces graphes, pour lesquels nous avons effectué plusieurs tests (dix simulation pour chaque problème afin de réaliser quelques informations statistiques sur l'évolution moyenne des critères considérés).

Pour cette étude, aucune comparaison avec les résultats d'autres travaux n'a pu être effectuée. A travers notre recherche bibliographique, nous ne sommes pas parvenus à identifier des travaux traitant la même problématique, avec un descriptif complet de jeux de données publiés qui nous auraient permis d'effectuer ces comparaisons. Néanmoins, les résultats obtenus permettent de juger, la taille des problèmes qui peuvent être résolus et sa rapidité. Afin de comparer entre les versions d'algorithme proposés dans ce chapitre, les algorithmes ont été implémentés en JAVA, et exécutés sur un processeur Pentium 4 (2,79GH) avec 192 Mo de mémoire vive.

2.6.2 Choix des paramètres OCF

Le tableau 2.1 présente les paramètres spécifiques à l'OCF utilisés pour tester les approches proposées dans ce chapitre. Ces paramètres ont été fixés après une série de tests préliminaires appliqués sur des réseaux aléatoires de différentes tailles.

	$tmax$	m	γ	α	β	ρ	$[\tau_{min}, \tau_{max}]$	C
OCF1-PCC-RTM	50	60	30	2	1	0.9	-	1
OCF2-PCC-RTM	50	60	-	2	1	0.9	$[0.007, 3]$	τ_{max}

TAB. 2.1 – Paramètres adoptés pour tester les algorithmes OCF-PCC-RTM

Pour les conditions d'arrêt ($tmax$, stagnation), on a constaté que nous arrivons à une situation de stagnation dans tous les cas de tests avant d'atteindre la valeur $tmax$, ce qui montre la convergence des approches proposées.

2.6.3 Résultats et discussions

Les tableaux 2.2, 2.3 présentent les résultats obtenus selon l'objectif à optimiser. Au tableau 2.2, en considérant la minimisation du coût de trajet, on retrouve dans la première partie du tableau, les résultats d'exécution de l'algorithme de base (OCF1-PCC-RTM), et dans la seconde partie, les résultats obtenus de la version améliorée (OCF2-PCC-RTM). La taille des problèmes considérés est représentée par un triplet $(|N|, |X|, |T|)$ où $|N|$ est le nombre de nœuds, $|X|$ le nombre de modes de transport et $|T|$ le nombre de dates de départ possibles pour l'ensemble des nœuds du réseau. Les deux tableaux présentent les valeurs du coût et du temps minimales (Min), maximale (Max), moyennes (Moy) et les temps moyens d'exécution (Moy-CPU) pour l'ensemble des exécutions réalisées pour chaque instance testée.

	OCF1-PCC-RTM(coût)				OCF2-PCC-RTM(coût)			
	Moy	Min	Max	Moy-CPU	Moy	Min	Max	Moy-CPU
P1(60, 3, 45585)	107.66	102	109	12.33	106.5	102	108	19.36
P2(60, 3, 45585)	176.83	175	181	16.85	175.16	174	177	25.89
P3(40, 3, 32681)	107.16	106	109	3.4	106.66	106	107	5.50
P4(30,3,21570)	113,5	103	122	3.54	111,33	103	117	5.72
P5(30,3,21570)	59	59	59	1.845	59	59	59	4.02
P6(30,3,21570)	103.33	100	106	8.55	100,83	100	103	4.50
P7(40,3,32681)	104.16	103	107	5.3	102.83	102	104	8.21

TAB. 2.2 – Comparaison OCF1-PCC-RTM (coût)/OCF2-PCC-RTM(coût)

La qualité des résultats trouvés dépend de la taille du réseau traité, les deux points source et destination sélectionnés et du nombre d'opérations effectuées afin de trouver la solution.

A partir des résultats illustrés dans le tableau 2.2, on constate que la version OCF2-PCC-RTM est supérieure ou égale à OCF1-PCC-RTM pour tous les problèmes testés en ce qui concerne la fonction objectif à optimiser. L'application de l'OCF2-PCC-RTM sera importante pour l'optimisation du temps de transport qui augmente la complexité du problème. La stagnation rapide de l'OCF1-PCC-RTM explique le pourquoi d'un temps d'exécution faible par rapport au l'OCF2-PCC-RTM.

L'algorithme de minimisation du temps de transport était testé sur le même jeu de données aléatoires. En plus, nous avons élargi l'application de tests sur d'autres problèmes (P8,...,P13). Les résultats présentés dans le tableau 2.3 sont obtenus par l'exécution de l'algorithme OCF2-PCC-RTM, en prenant en considération les deux méthodes d'optimisation du temps (mono-objectif et bi-objectif) citées dans la section 2.5, on retrouve dans la première partie du tableau les résultats d'exécution

	OCF2-PCC-RTM1(temps)				OCF2-PCC-RTM2(temps)			
	Moy	Min	Max	Moy-CPU	Moy	Min	Max	Moy-CPU
P1(60, 3, 45585)	186.33	168	223	30.81	168	168	168	16.92
P2(60, 3, 45585)	396.66	328	521	35.91	237	237	237	19.97
P3(40, 3, 32681)	209.83	193	238	8.79	195.33	193	207	5.02
P4(30,3,21570)	139,25	137.5	142.5	7.94	137.5	137.5	137.5	3.23
P5(30,3,21570)	59.25	59.25	59.25	4.16	59,25	59,25	59,25	2.08
P6(30,3,21570)	123.5	121	136	8.65	121	121	121	5.96
P7(40,3,32681)	136.87	135.75	138.5	9.78	135	135	135	6.35
P8(100,3, 78315)	330.64	295	380	150.02	292.7	290	294.75	111.66
P9(100, 3, 78315)	281.59	226	343	224.94	178.68	163.5	204	177.375
P10(100,3, 78315)	154.46	117.75	269	162.5	117.75	117.75	117.75	104.5
P11(300,3,693660)	105.55	100	116.5	509.61	98.5	98.5	98.5	325.67
P12(300, 3, 693660)	199.87	191	213	754.99	122.75	122.75	122.75	489.59
P13(600, 3, 1182840)	220.83	162.5	294	2987.66	142.85	137	162.5	2044

TAB. 2.3 – Comparaison OCF2-PCC-RTM1(temps)/OCF2-PCC-RTM2(temps)

de la variante d'optimisation mono-objectif (OCF2-PCC-RTM1), et la variante bi-objectif dans la seconde partie du tableau (OCF2-PCC-RTM2).

A partir de ces résultats, La résolution mono-objectif de problème d'optimisation du temps a été éliminée. Elle présentait une qualité des solutions médiocres au prix de temps d'exécution important. La résolution bi-objectif a permis de confirmer ces mauvais résultats.

2.7 Conclusion

Le problème de plus court chemin multimodal considéré est un problème dynamique NP-Difficile. Dans le but de proposer une résolution de ce problème, Nous avons utilisé les algorithmes de colonies de fourmis. Plusieurs approches OCF sont proposées dans cette première contribution. En premier lieu, nous avons présenté l'algorithme de base OCF1-PCC-RTM. Cet algorithme a été amélioré pour obtenir une autre version OCF2-PCC-RTM. Puis nous avons présenté deux variantes pour l'optimisation du temps de trajet en incluant les temps d'attentes. La comparaison des résultats ont montré l'intérêt de l'OCF2-PCC-RTM et la variante bi-objectif d'optimisation du temps de transport.

Les approches de résolution proposées dans ce chapitre fournissent une seule solution, chose qui convient si les critères considérés ne sont pas contradictoires dans l'évaluation des solutions, dans le cas contraire, il nous faut une population de solutions. Les approches développées dans ce sens font l'objet du chapitre suivant.

Chapitre 3

OCF multi-objectif appliquée au réseau de transport multimodal

3.1 Introduction

L'optimisation par colonie de fourmis est une métaheuristique qui a été utilisée avec succès pour résoudre une multitude de problèmes d'optimisation combinatoire. Toutefois, la plupart des travaux ont consisté à solutionner des problèmes à objectif unique. En pratique, les problèmes d'optimisation réels rencontrés sont souvent multi-objectif. Ces problèmes ont la particularité d'être beaucoup plus difficiles à traiter que leur équivalent mono-objectif. La difficulté principale est qu'il n'existe pas une seule solution optimale, mais un ensemble de solutions, connu comme l'ensemble des solutions Pareto optimales. Toutes les solutions de cet ensemble sont optimales dans le sens où le décideur peut simplement exprimer le fait qu'une solution est préférable à une autre, mais il n'existe pas une solution meilleure que toutes les autres.

Dans ce chapitre, nous proposons une nouvelle approche OCF de recherche d'un ensemble de solutions Pareto optimales. L'objectif principal de cette approche est de générer une variété de solutions optimales diversifiées dans l'espace de recherche.

Nous commençons dans ce chapitre par définir quelques notions de l'optimisation multi-objectif (section 2). Puis un état de l'art des approches de résolution OCF proposées dans le contexte multi-objectif (section 3 et 4). Nous présentons ensuite l'algorithme OCF proposé afin de solutionner plus efficacement la variante multi-objectif du problème de transport multimodal (section 5) et nous fournissons, enfin, les résultats des tests permettant de juger des performances et de la validité de notre approche (section 6).

3.2 Optimisation multi-objectif

Cette section est dédiée à l'optimisation multi-objectif. Nous définissons en premier lieu quelques termes liés aux problèmes d'optimisation multi-objectif, puis nous présentons une classification des différentes approches de résolution existantes dans ce domaine.

Définition 3. *Un Problème d'Optimisation Multi-objectif (POM) peut être défini de la manière suivante :*

$$PMO = \begin{cases} \min F(x) = \{f_1(x), f_2(x), \ldots, f_m(x)\} \\ x \in S \end{cases} \tag{3.1}$$

Où $X = \{x_1, x_2, \ldots, x_r\}$ est le vecteur représentant les variables de décision, S représente l'ensemble des solutions réalisables associé à des contraintes d'égalité, d'inégalité et des borne explicites (espace de décisions) et $F(x) = \{f_1(x), f_2(x), \ldots, f_m(x)\}$ est un vecteur de $m > 2$ objectifs à optimiser [51].

3.2.1 Solution d'un problème multi-objectif

L'optimisation multi-objectif consiste à trouver des solutions optimales en utilisant des méthodes efficaces qui offrant un bon compromis entre les différents objectifs. Contrairement à l'optimisation mono-objectif, l'espace de solutions d'un POM n'est pas une solution unique, mais un ensemble de solutions, connu comme l'ensemble des solutions Pareto Optimales. Toute solution de cet ensemble est optimale dans le sens qu'aucune amélioration ne peut être faite sur un composant du vecteur F sans dégradation d'au moins un autre composant de ce vecteur. Un décideur choisit alors parmi les différentes solutions proposées par l'optimiseur, la solution de compromis qui lui convient le mieux. L'ensemble Pareto est identifié par une relation d'ordre entre toutes les solutions trouvées. Cette relation d'ordre est appelée *relations de dominance.*

3.2.2 Notions de base

Définition 4. ***(relations de dominance)*** *Une solution s_i domine une solution s_j si et seulement si :*

$$\forall k \in [1, m]\ f_k(s_i) \leq f_k(s_j)\ et\ \exists k \in [1, m]\ f_k(s_i) < f_k(s_j) \tag{3.2}$$

Où m est le nombre d'objectifs à optimiser

Le fait qu'une solution s_i domine une solution s_j sera noté $s_j \leq s_i$. Si s_i est meilleur que s_j pour tous les objectifs, on notera alors $s_j < s_i$, lorsque on a ni $s_j \leq s_i$, ni $s_i \leq s_j$, on notera alors $s_j <> s_i$.

Définition 5. ***(Ensemble Pareto)*** *Les solutions qui dominent les autres mais ne se dominent pas entre elles sont appelées solutions optimales au sens de Pareto (ou solutions non dominées). Pour un problème d'optimisation multi-objectif donné, l'ensemble Pareto Optimal (PO) est définit comme suit :*

$$PO = \{x \in S, \neg\exists y \in S\, x \leq y\} \qquad (3.3)$$

L'ensemble des solutions Pareto constitue la surface de compromis, appelée aussi *frontière Pareto*. Un exemple d'une surface de compromis d'un problème de minimisation bi-objectif est présenté à la figure 3.1.

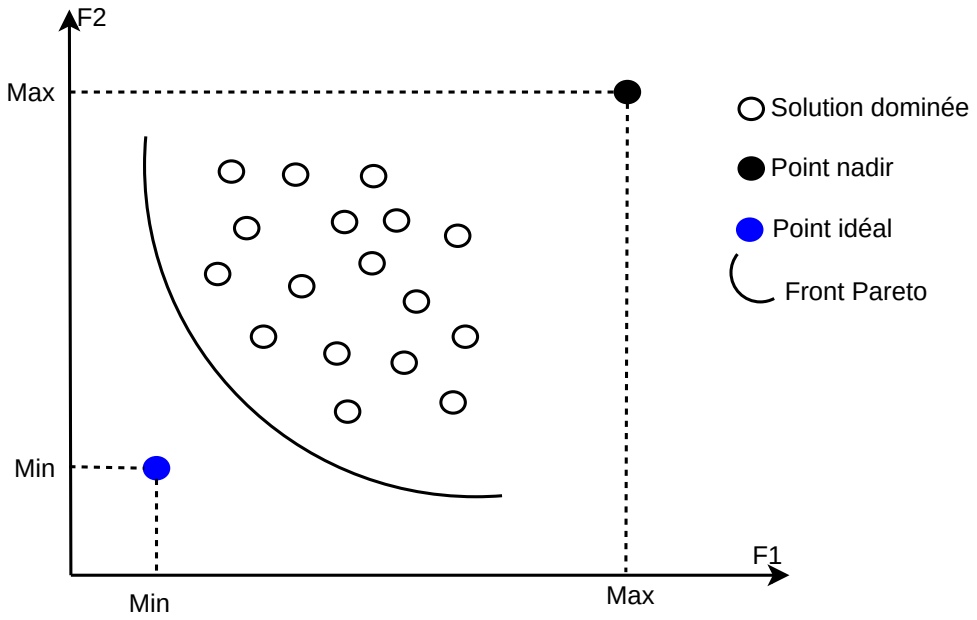

FIG. 3.1 – Représentation de la surface de compromis (cas de minimisation de deux objectifs)

L'objectif principal de la résolution d'un problème multi-objectif est de générer une variété de solutions Pareto optimales diversifiées dans l'espace de recherche.

Définition 6. ***(Point idéal)*** *l'optimisation de chaque objectif pris séparément permet de trouver les différentes solutions optimales. Le vecteur de solutions correspond alors au point idéal et peut être exprimé comme suit :*

$$\begin{cases} F^* = (f_1^*, f_2^*, \ldots, f_n^*) \\ f_i^* = \min_{x \in S} f_i(x) \end{cases} \qquad (3.4)$$

Il représente le but à atteindre et correspond généralement à une solution non réalisable, Le point idéal est utilisé dans plusieurs méthodes d'optimisation comme un point de référence.

Définition 7. ***(Point nadir)*** *Le point nadir représente les valeurs maximales pour chacun des objectifs dans l'ensemble des solutions, il se définit de la manière suivante :*

$$\begin{cases} F^- = (f_1^-, f_2^-, \ldots, f_n^-) \\ f_i^- = \max_{x \in S} f_i(x) \end{cases} \tag{3.5}$$

3.2.3 Classification des métaheuristiques de résolution multi-objectif

Plusieurs adaptations de métaheuristiques ont été proposées dans la littérature pour la résolution de PMO et la détermination des solutions Pareto. Ces approches peuvent être classées en trois catégories [51] :

- **Les approches scalaires** : Plusieurs approches traditionnelles transforment le PMO en un problème mono-objectif. Pour ces approches, une fonction objectif est créée de telle sort qu'elle permet de traiter le problème d'optimisation multi-objectif comme un problème d'optimisation classique à un seul objectif, et donc d'utiliser des méthodes déjà existantes pour traiter de tels problèmes. Parmi les approches proposées dans cette classe, on trouve les méthodes d'agrégation, les méthodes $\epsilon-$contrainte, et les méthodes de programmation par but (goal programming). Le défaut de cette classe est qu'elle ne génère qu'une seule solution optimale et nécessite pour le décideur d'avoir une bonne connaissance de son problème.

- **Les approches non-scalaires et non-Pareto** : Ces approches ne transforment pas le problème en un problème mono-objectif. Elles utilisent des opérateurs de recherche qui traitent séparément les différents objectifs. D'une manière générale, les deux méthodes non-Pareto trouvées dans la littérature sont incluses dans les algorithmes évolutionnaires : la *sélection parallèle* et la *sélection lexicographique*. L'inconvénient de ces méthodes est qu'elles tendent à générer des solutions qui sont largement optimisées pour certains objectifs et très peu pour les autres objectifs. Les solutions de compromis sont négligées.

- **Approches Pareto** : les approches Pareto utilisent directement la notion d'optimalité Pareto dans leur processus de recherche. Ce concept a été introduit initialement dans les algorithmes génétique par Goldberg [29]. Les

processus de sélection des solutions générées sont basés sur la notion de non dominance. Le principal avantage de ces approches est qu'elles sont capables de générer des solutions Pareto optimales dans le sens de ne pas favoriser un objectif plutôt qu'un autre, mais les traitent de manière équitable, ce qui fournit une aide précieuse au décideur.

3.3 OCF pour un problème Multi-objectif

Les algorithmes évolutionnaires ont été utilisés largement pour résoudre les PMO, et une large part de ces algorithmes évolutionnaires sont des algorithmes génétiques. Cependant, peu d'approches OCF sont développées pour les problèmes multi-objectif. Ces approches diffèrent principalement en ce qui concerne les trois points suivants : La structure de phéromone, La méthode de mise à jour, et la définition de facteurs heuristiques [1] :

La structure de phéromone

La quantité de phéromone déposée représente l'expérience de passé de la colonie. Quand il y a seulement une seule fonction objectif, l'expérience de passé est définie en fonction de cet objectif. En cas de plusieurs critères d'optimisation, deux stratégies différentes sont considérées. Une première stratégie considère une seule mémoire de phéromone (structure simple), comme proposée dans [25], dans ce cas, la quantité de phéromone représente une agrégation entre les différents objectifs. Une deuxième stratégie considère plusieurs structures de phéromone, comme proposé dans [35], dans ce cas, une colonie est associée à chaque objectif (chaque colonie ayant sa propre structure de phéromone).

La méthode de mise à jour

En mettant à jour la trace de phéromone, on doit choisir finalement lesquelles des solutions construites mettant à jour la mémoire de phéromone. Une première possibilité est de récompenser les solutions qui trouvent les meilleures valeurs pour chaque critère dans le cycle actuel, comme proposé dans [15]. Une deuxième possibilité est de récompenser les solutions efficaces du cycle actuel, dans ce cas, on peut qualifier toutes les solutions optimales dans le sens Pareto, comme proposé dans [47], ou seulement les nouvelles solutions qui entrent dans l'ensemble Pareto, comme proposé dans [35].

Définition de facteurs heuristiques

Deux stratégies peuvent être envisagées. Une première stratégie considère une agrégation des différents objectifs en construisant une seule information heuristique, comme proposée dans [47]. La deuxième stratégie considère séparément une information heuristique pour chaque fonction objectif, comme proposée dans [25], dans ce cas, une colonie est associée à chaque objectif.

3.4 Les travaux OCF dans le contexte Multi-Objectifs

Dans cette section, nous présentons l'essentiel des travaux utilisant OCF dans le domaine d'optimisation multi-objectif :

Gambardella et al [25] ont développé un algorithme OCF bi-objectif où la meilleure solution globale est partagée par deux colonies pour solutionner un problème de tournées de véhicules. Dans cette approche, une colonie cherche à optimiser la meilleure solution connue sur le premier objectif (Le nombre de véhicules) tandis que l'autre cherche à optimiser le deuxième objectif (Le temps total des tours) tout en préservant la première solution, Les deux colonies partagent la même meilleure solution globale qui est utilisée pour la mise à jour de phéromone.

Mariano et Morales [42] proposent une multi-colonie pour la conception d'un réseau de distribution d'eau où ils associés une colonie à chaque objectif. Chaque objectif est influencé seulement sur une partie d'une solution. La colonie (i) reçoit une solution partielle des fourmis de colonie $(i-1)$, et essaye ensuite d'améliorer cette solution en ce qui concerne le critère (i).

Alaya et al [1] proposent un algorithme générique appelé m-ACO (m est le nombre d'objectifs à optimiser). Le m-ACO est paramétré par le nombre de colonies de fourmis $\#Col$ et le nombre de structures de phéromone considérés $\#\tau$. L'algorithme 3.1 décrit la structure générique de m-ACO, quatre variantes sont proposées selon les valeurs des deux paramètres $\#Col$ et $\#\tau$. Les résultats expérimentaux montrent l'efficacité de la variante 4 qui considère une seule colonie et m structures de phéromone.

Seules approches OCF qui permet de couvrir le front Pareto (trouver un ensemble de solutions non dominées) sont proposées dans [35, 47, 15, 30].

Iredi et al [35] proposent, de leur part, une approche OCF bi-objectif basée sur la coopération de plusieurs colonies pour le problème d'ordonnancement sur machine unique avec temps de réglages dépendants de la séquence. Chacune des colonies possède deux matrices de trace de phéromone $M = \{\tau_{ij}\}, M' = \{\tau'_{ij}\}$. Les fourmis

Algorithm 3.1 m-ACO()

```
1: Début
2: Initialisation
3: répéter
4:    pour chaque colonie c ∈ [1, #Col]  faire
5:       pour chaque fourmis k ∈ [1, n]  faire
6:          Construction d'une Solution
7:       fin pour
8:    fin pour
9:    pour i ∈ [1, #τ]  faire
         Mettre à jour i^eme structure de phéromone
10:   fin pour
11: jusqu'à nombre maximal de cycles atteint
12: Fin.
```

dans une génération mettent leurs solutions dans un ensemble globale Q qui est partagé par toutes les colonies. Q est utilisé pour déterminer le front non dominé de toutes les solutions. Alors, seulement les solutions de l'ensemble Q sont utilisées pour effectuer la mise à jour de la mémoire de phéromone, pour cela, deux variantes différentes sont proposées dans cette approche :

1. **Mise à jour par origine :** une fourmi met à jour la trace de phéromone seulement dans sa propre colonie.
2. **Mise à jour par région :** Les solutions de l'ensemble Q sont triées selon l'un de ses critères d'optimisation. Soit L la liste triée, cette liste est alors divisée en m parties $\{L_1, L_2, \ldots, L_m\}$ pour que leurs tailles diffèrent par un au maximum, la mise à jour de la trace de i^{eme} colonie se fait uniquement par les solutions de la liste L_i.

Dans ce travail, les auteurs ont amené le concept de *Colonies hétérogènes*, ce nouveau concept permet de diversifier la recherche de solutions dans le front Pareto, où les fourmis ont des préférences (importance relative) différentes entre les critères dans la construction d'une solution, plus exactement, chaque fourmi $k \in [1, n]$ utilise un paramètre $\lambda = \frac{k-1}{n-1}$ pour déterminer l'influence relative de chaque critère. En effet, chaque fourmi choisit la prochaine commande j à ordonnancer suite à la commande i précédente par la probabilité suivante :

$$P_{ij} = \frac{[\tau_{ij}]^{\lambda\alpha}\ [\tau'_{ij}]^{(1-\lambda)\alpha}\ [\eta_{ij}]^{\lambda\beta}\ [\eta'_{ij}]^{(1-\lambda)\beta}}{\sum_{h\in S}[\tau_{ih}]^{\lambda\alpha}\ [\tau'_{ih}]^{(1-\lambda)\alpha}\ [\eta_{ih}]^{\lambda\beta}\ [\eta'_{ih}]^{(1-\lambda)\beta}} \qquad (3.6)$$

Ainsi, dans les cas extrêmes, la dernière fourmi n avec $\lambda = 1$ considère seulement le premier critère, tandis que la première fourmi avec $\lambda = 0$ considère seulement le deuxième critère.

Pinto et Baràn [47] proposent un algorithme similaire à celui de iredi et al [35].

Le concept d'hétérogénéité est utilisé afin de modifier les deux algorithmes mono-objectif *Max-Min Ant System* et le *Ant Colony System*(Algorithme 3.2). 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 :

$$P_{ij} = \frac{[\tau_{ij}]^{\alpha}([\eta 1_{ij}]^{\lambda_1}[\eta 2_{ij}]^{\lambda_2}[\eta 3_{ij}]^{\lambda_3})^{\beta}}{\sum_{h \in S}[\tau_{ih}]^{\alpha}([\eta 1_{ih}]^{\lambda_1}[\eta 2_{ih}]^{\lambda_2}[\eta 3_{ih}]^{\lambda_3})^{\beta}} \qquad (3.7)$$

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

Algorithm 3.2 approche Pinto et Baràn

1: **Début**
2: Initialisation
3: **répéter**
4: **pour** $\lambda_1 = 0, \ldots, m-1$ **faire**
5: **pour** $\lambda_2 = 0, \ldots, m-1$ **faire**
6: $\lambda_3 \leftarrow m - 1 - \lambda_2$
7: Construction d'une Solution
8: **fin pour**
9: **fin pour**
10: Mettre à jour la trace de phéromone
11: **jusqu'à** un critère d'arrêt est vérifié
12: Return Q
13: **Fin**.

Doerner et al [15] proposent une résolution OCF d'un problème d'optimisation de portefeuille bi-objectif. Ils associaient une mémoire de phéromone à chaque objectif. Afin d'assurer l'hétérogénéité, chaque fourmi assigne des poids aux informations de phéromone selon un vecteur de poids aléatoire en construisant une solution. La mise à jour de phéromone est faite par les fourmis qui ont trouvé la meilleure ou la deuxième meilleure solution pour chaque objectif.

Guntsch [30] introduit une approche appelée *Population Ant Colony Optimization (PACO)*, *PACO* maintient une population P qui représente les solutions reflétées complètement dans la mémoire de phéromone. Soit Q l'ensemble de solutions non dominées en faveur des solutions qui ont été trouvées après un certain nombre d'itération par l'algorithme PACO, la population active $P \subseteq Q$ est tirée pour mettre à jour la mémoire de phéromone.

Au début, l'algorithme choisit aléatoirement une solution de départ $\pi \in Q$, puis choisir $k-1$ solutions de Q qui sont les plus proches de π selon une certaine mesure de distance, où k est la taille de la population P. La distance est définie simplement

par la somme des différences absolues de la qualité de solutions sur tous les critères. L'ensemble Q est mis à jour après que chaque fourmi y construit une solution. On peut résumer le fonctionnement général de PACO dans l'algorithme 3.3.

Algorithm 3.3 PACO

Début
répéter
  pour chaque fourmis $\in [1, n]$ **faire**
    Construction d'une Solution
    Mettre à jour l'ensemble Pareto Q.
  fin pour
  Choisit aléatoirement une solution $\pi \in Q$.
  $P \leftarrow \{\pi\}$
  tantque $\mid P \mid <$k **faire**
    Trouver $s^+ = \min_{s \in Q/P} \sum_{j=1}^{m} \mid f_j(\pi) - f_j(s) \mid$
    $P \leftarrow P \cup \{s^+\}$
  fin tantque
  Mettre à jour la trace de phéromone
  Mettre à jour le vecteur de poids W
jusqu'à un critère d'arrêt est vérifié
Fin.

Pour calculer la probabilité de transition, il est possible de construire la probabilité individuelle de transition pour chaque critère $l = 1, \ldots, m$ d'une façon directe via la probabilité de base de l'algorithme OCF, en utilisant les informations heuristiques disponibles pour chaque critère, puis on doit agréger de façon significative toutes les probabilités P_l.

$$P = \prod_{l=1}^{m} w_l \; P_l \qquad (3.8)$$

L'idée de base pour accomplir cette accumulation est d'assigner un poids $w_l \in [0, 1]$ pour chaque probabilité de transition P_l. La population P est utilisée pour calculer le vecteur de poids W, le principe est de donner à chaque critère un poids important si les solutions de P sont bonnes pour ce critère comparativement à toutes les autres solutions de Q.

3.5 Notre approche : Pareto multi-objectif d'optimisation par colonie de fourmis (PM-OCF)

Dans cette section, nous proposons un algorithme OCF pour l'optimisation multicritère. Le but est de trouver les différentes solutions qui couvrent le front Pareto optimal. Nous utilisons une colonie hétérogène [35, 47, 15], où chaque fourmi donne

une importance relative (poids) aux critères d'optimisation différemment pour qu'ils soient capables de diversifier la recherche dans l'espace des solutions réalisables. Dans les deux approches [35, 47], les valeurs des poids changent d'une manière déterministe. Dans [15], les valeurs des poids changent d'une manière aléatoire. Afin d'accélérer la convergence de notre approche vers les solutions de compromis, les valeurs des poids changent dynamiquement d'une fourmi à une autre en fonction de la distance normalisée entre l'ensemble des solutions Pareto optimales et le point idéal pour chaque objectif.

Etant donné que l'algorithme OCF2-PCC-RTM (chapitre précédent, section 2.5) pour optimiser le temps du trajet et les temps d'attentes, Dans ce travail, nous modifions cet algorithme afin de résoudre des problèmes d'optimisation multi-objectif, avec les changements suivants :

- Les objectifs à minimiser sont : le temps du trajet (f_1), les temps d'attentes (f_2), et le coût de transport (f_3).

- Le but est de trouver un ensemble Q de solutions Pareto optimales au lieu de trouver une seule solution.

- Pour guider les fourmis dans l'espace de recherche, on propose trois informations heuristiques η_1, η_2, η_3 pour les trois fonctions objectifs f_1, f_2, f_3 et deux autres variables w_1, w_2 pour déterminer l'influence relative de chaque heuristique, où w_1 est le poids accordé à η_1 et η_2, w_2 est le poids accordé à η_3.

- Les informations heuristiques de déplacement du nœud courrant i vers le nœud j par un mode de transport x en quittant i à une date t sont :

$$\eta_{1[ij(x,t)]} = \frac{1}{\upsilon_{ij(x,t)}},\ \eta_{2[ij(x,t)]} = \frac{1}{\theta_{ij(x,t)}},\ \eta_{3[ij(x,t)]} = \frac{1}{\phi_{ij(x,t)}}$$

- On donne la règle de déplacement qui consiste à choisir à la fois le successeur du nœud courant i à visiter, et le mode de transport x à utiliser, et la date de départ t correspondante par :

$$P_{ij(x,t)} = \begin{cases} \frac{\tau^{\alpha}_{ij(x,t)}(w_1 \times (\eta_{1[ij(x,t)]} \times \eta_{2[ij(x,t)]}) + w_2 \times \eta_{3[ij(x,t)]})^{\beta}}{\sum_{k \notin tabu} (\tau_{ik\,\forall(w,l)})^{\alpha}(w_1 \times (\eta_{1[ik\,\forall(w,l)]} \times \eta_{2[ik\,\forall(w,l)]}) + w_2 \times \eta_{3[ik\,\forall(w,l)]})^{\beta}} & j \notin tabou \\ 0 & Sinon \end{cases} \tag{3.9}$$

- L'ensemble Pareto Q est mis à jour après que chaque fourmi y construit une solution.

- Les solutions de l'ensemble Q sont utilisées pour mettre à jour la mémoire de phéromone.

Cette méthode a été formalisée en intégrant les éléments décrits précédemment dans une procédure générique qui se déroule en cinq phases comme la présente

l'algorithme 3.4. Chacune des phases sera maintenant expliquée en identifiant les notions de base à réaliser pour la résolution du problème de transport multimodal.

Algorithm 3.4 PM-OCF

Début
Etape1 : Recherche du point idéal s^*
Etape2 :
  Initialiser les paramètres OCF.
  Initialiser la trace de phéromone à τ_{max}.
  Initialiser l'ensemble Pareto Q à un ensemble vide.
Etape3 :
répéter
  pour chaque fourmis k $\in [1, m]$ **faire**
    $s_k \leftarrow Construction - Solution()$
    Evaluer la solution s_k obtenue selon les différents objectifs et mettre à jour l'ensemble Pareto Q avec les solutions non dominées.
    si (il y a un mis à jour dans l'ensemble Q) **alors**
      Calculer le vecteur de poids $W_{k+1} = (w_1, w_2)$
    finsi
  fin pour
  Etape4 :
  Evaluer les solutions de l'ensemble Pareto Q selon les différents objectifs et Mettre à jour la mémoire de phéromone en respectant les limites $[\tau_{min}, \tau_{max}]$
jusqu'à un critère d'arrêt est vérifié.
Etape5 : Affichage des solutions Pareto Optimales.
Fin.

Etape1 : Recherche du point idéal

L'optimisation de chaque objectif pris séparément ait permis de trouver les différentes solutions optimales. Le vecteur de solutions correspond alors au point idéal et peut être exprimé par : $S^* = \{s_1^*, s_2^*\}$ Où :

- s_1^* : La solution optimale qui minimise le temps total de transport (temps du trajet et les temps d'attentes)
- s_2^* : La solution optimale qui minimise le coût total de transport.

Le point idéal est généralement utilisé dans la programmation par but qui consiste à chercher dans l'espace de solutions admissibles, la solution minimisant la distance de point idéal S^*. Dans notre approche, le point idéal est utilisé plus tard pour calculer les valeurs de poids w_1 et w_2 dans l'étape 3.

Etape2 : Initialisation

Cette phase consiste tout d'abord à initialiser les paramètres spécifiques à l'OCF tel que le nombre de fourmis, le nombre maximal de cycles, le paramètre d'évaporation de la trace de phéromone, l'intervalle $[\tau_{min}, \tau_{max}]$. Les paramètres de la règle de

transition (α, β et le vecteur de poids W). Les pistes sont initialisées à leur valeur maximale τ_{max}, afin de garantir une exploration plus large de l'espace de recherche durant les premiers cycles. L'ensemble Pareto des solutions non dominées Q est initialisé à l'ensemble vide.

Etape3 :Déroulement d'un cycle

La construction de solution progresse tant que la fourmi n'arrive pas à la ville destination. A chaque étape dans la construction, le choix de transition est effectué selon la règle de déplacement présentée par la formule 3.9, cette règle est définie proportionnellement à un facteur phénoménal τ et un facteur heuristique qui regroupe les trois fonctions objectifs à optimiser f_1, f_2, f_3. Ces trois fonctions étant pondérés par $W = \{w_1, w_2\}$ qui déterminent leur importance relative. Notons que les valeurs des poids changent dynamiquement d'une solution à une autre en fonction de la distance normalisée entre l'ensemble de solutions Pareto optimales et le point idéal pour chaque objectif. Formellement, les valeurs de poids sont calculées en utilisant la formule suivante :

$$\begin{cases} w_1 = \sqrt{\frac{1}{|Q|}\sum_{j=1}^{|Q|}(T_{SD}(s_j) - T_{SD}(s_1^*))^2} \\ w_2 = \sqrt{\frac{1}{|Q|}\sum_{j=1}^{|Q|}(f_3(s_j) - f_3(s_2^*))^2} \end{cases} \tag{3.10}$$

Afin de garantir, d'une part, que la valeur de chaque poids appartient à l'intervalle $[0, 1]$ et, d'autre part, pour la somme de ces poids soit égale à 1 : $w_i = \frac{w_i}{w_1 + w_2}$ avec i=1,2.

Les valeurs de poids permettent de guider la construction de solutions en fonction de la direction de recherche à emprunter, la valeur de poids est plus élevée lorsque l'objectif en question est désigné prioritaire. En effet, après la construction de la $k-1^{eme}$ solution, s'il y a une mise à jour dans l'ensemble de solutions Pareto optimales par la fourmi $k-1$ et l'évaluation d'un critère j dans cet ensemble est proche du s_j^*, ce critère aura une valeur de poids plus petite que celles attribuées aux critères éloignés dans la construction de la k^{eme} solution. Par exemple : à un cycle donné soit $Q_{k-1} = \{(350, 10), (200, 30), (150, 50), (100, 55)\}$ l'évaluation de l'ensemble Pareto mise à jour par la fourmi $k-1$, chaque élément de Q contient deux composants qui représentent respectivement le temps total (temps du trajet plus les temps d'attentes) et le coût total de transport. Supposons que l'évaluation du point idéal de ce problème est définie par $(40, 10)$. De ce fait, les valeurs de poids accordées à la fourmi k sont respectivement $0,85$ et 0.15.

Cette méthode de calcul permet d'assurer la notion d'hétérogénéité, afin d'élargir légèrement l'espace de recherche et ainsi être en mesure d'atteindre des compromis intéressants se trouvant a proximité. De plus, elle permet d'obtenir des poids qui convergent vers le point idéal.

Suite à la construction d'une solution, l'évaluation de cette dernière est faite pour chacun des objectifs du problème. Les relations de dominance sont alors vérifiées pour chacune des solutions et les solutions Pareto sont conservées dans un ensemble Q. Formellement, cette opération est représentée par :

$$\text{Si } (S) \text{ n'est pas dominée par } \forall T \in Q \text{ Alors } Q = S \cup Q - \{\forall T \in Q, T \leq S\} \quad (3.11)$$

Etape4 : La mise à jour de phéromone

A la fin d'une itération, chaque fourmi trouve une solution, et les pistes de phéromone doivent être mises à jour sur la base de l'évaluation réalisée sur l'ensemble des objectifs. Il doit être décidé lesquelles des fourmis permettent de mettre à jour la mémoire de phéromone. Ici, nous proposons que l'on permet à toutes les fourmis dans le front non dominé de la génération actuelle (l'ensemble Q) de mettre à jour la mémoire de phéromone. La mise à jour doit assurer que la trace de phéromone respecte les limites $[\tau_{min}, \tau_{max}]$. Formellement, cette opération s'effectue en deux étapes (formule 3.12). Dans une première étape, toutes les traces de phéromone sont diminuées, pour simuler l'évaporation en multipliant chaque composant phéromonal par un ratio de persistance ρ tel que $0 < \rho < 1$, Dans une deuxième étape, les fourmis trouvant des solutions Pareto optimales déposent une quantité Δ de phéromone.

$$(I) \begin{cases} \forall (i,j) \in A \tau_{ij\forall(x,t)} = \rho \times \tau_{ij\forall(x,t)} \\ \tau_{ij(x,t)} = \tau_{min} \; si \; \tau_{ij(x,t)} \prec \tau_{min} \end{cases} \quad (3.12)$$

$$(II) \begin{cases} \forall T \in Q : \forall (i,j,x,t) \in T \tau_{ij(x,t)} = \tau_{ij\forall(x,t)} + \Delta \\ \tau_{ij(x,t)} = \tau_{max} \; si \; \tau_{ij(x,t)} \succ \tau_{max} \end{cases}$$

Etape5 :Vérification de conditions d'arrêt

Le processus itératif prend fin lorsque les conditions d'arrêt sont atteintes. Dans notre approche, l'algorithme s'arrête après un nombre maximum de cycles $tmax$.

Etape6 : Affichage des solutions au décideur

L'ensemble des solutions Pareto optimales trouvé par l'algorithme doit maintenant être présenté au décideur. Pour l'aider à prendre une décision, il convient de lui permettre d'explorer l'ensemble des solutions en fonction de ses préférences.

3.6 Résultats expérimentaux

3.6.1 Métriques de comparaisons

La comparaison d'algorithmes pour la résolution exacte des problèmes d'optimisation est triviale. En effet, il suffit de comparer la taille et le nombre de problèmes résolus par chaque méthode, en tenant compte éventuellement du temps nécessaire pour la découverte de la solution optimale. Dans le cas des métaheuristiques, la comparaison de deux solutions obtenues reste triviale pour l'optimisation mono-objectif (on compare les valeurs trouvées), mais pas pour l'optimisation multi-objectif. En effet, l'évaluation des performances de solutions d'un algorithme multi-objectif n'est pas triviale, puisque celui-ci fournit un ensemble de solutions non dominées et donc non comparable entre elles [3]. Il est donc nécessaire d'utiliser des indicateurs de performance afin d'évaluer ces algorithmes. Peu de mesures sont présentées dans la littérature pour calculer l'efficacité des méthodes d'optimisation pour les problèmes multi-objectif ont été proposées.

Dans nos expérimentations, nous aurons affaire à des problèmes pour lesquels nous ne connaissons pas la frontière Pareto optimale. Pour l'évaluation des différentes approximations Pareto calculées, nous utiliserons deux métriques de comparaison afin de mesurer la convergence et la diversification de l'approche proposée.

3.6.1.1 La métrique C (métrique de Convergence)

cette métrique, proposée par Zitzler [56], calcule la proportion de solutions d'un ensemble potentiellement Pareto optimal Q_2 dominées par des solutions potentiellement Pareto optimal Q_1 :

$$C(Q_1, Q_2) = \frac{|\{b \in Q_2, \exists a \in Q_1 : a > b\}|}{|Q_2|} \qquad (3.13)$$

$C(Q_1, Q_2) = 1$ signifie que toutes les solutions trouvées dans l'ensemble Q_2 sont dominées par celles trouvées dans l'ensemble Q_1. Tandis que $C(Q_1, Q_2) = 0$ indique qu'aucune solution engendrée dans Q_2 n'est dominée par une solution trouvée dans Q_1. Ainsi, plus la valeur $C(Q_1, Q_2)$ se rapproche de 1, il est meilleur de considérer Q1. Comme la relation de dominance n'est pas symétrique, $C(Q_1, Q_2)$ n'est forcément égale à $C(Q_2, Q_1)$, il est donc nécessaire de calculer les deux valeurs $C(Q_1, Q_2)$ et $C(Q_2, Q_1)$.

L'utilisation de la métrique C pour évaluer les performances d'une métaheuristique n'est pas suffisante. Nous avons donc besoin d'un autre indicateur complémentaire à la métrique C permettant d'évaluer la diversité d'un front Pareto. Parmi ces indicateurs se trouve la métrique définie ci-dessous.

3.6.1.2 La métrique d'espacement (Spacing)

cette métrique, proposée par Schott [48], mesure l'uniformité de la répartition des solutions de la surface de compromis dans l'espace de fonctions $(f_1, \ldots, f_m)$. $S(Q)$ est définie par :

$$S(Q) = \sqrt{\frac{1}{|Q|}\sum_{i=1}^{|Q|}(d_i - \bar{d})^2}$$

Où (3.14)

$$d_i = \min_{k \in Q \wedge k \neq i} \sum_{m=1}^{n} |f_m^i - f_m^k|$$

et $\bar{d}$ est la valeur moyenne de la distance d_i : $\bar{d} = \frac{\sum_{i=1}^{|Q|} d_i}{|Q|}$

Cette métrique calcule les écarts type des différentes valeurs de d_i. Ainsi, si les solutions sont uniformément espacées, la distance correspondante sera faible. Donc, plus un algorithme trouve un ensemble Pareto optimal pour lequel cette mesure est faible, meilleur il est.

L'utilisation des deux métriques présentées permet ainsi la comparaison des différentes métaheuristiques et l'ajustement des paramètres de l'approche proposée. Notons que les mesures présentées ne tiennent pas compte des préférences du décideur.

3.6.2 Protocole de test

Comme nous l'avons présenté dans le chapitre précédent, les réseaux de transport sur lesquels des tests ont été effectués sont créées en exploitant un générateur aléatoire que nous avons développé. Dans ce chapitre, les tests sont réalisés sur des graphes de 30, 60, et du 100 nœuds. De plus, le nombre de modes de transport disponibles est fixé à trois. Cinq problèmes (P1,...,P5) sont construits selon les points de départ et destination sélectionnés dans ces graphes, pour lesquels nous avons examiné plusieurs tests (cinq simulations pour chaque problème afin de réaliser quelques informations statistiques sur l'évolution moyenne des résultats trouvés).

Dans le but d'illustrer et de démontrer l'efficacité de l'approche de résolution proposée dans ce chapitre, les travaux de Pinto et Baràn [47] et de Doerner et al [15] (section 4) sont implémentés et testés sur le même jeu de données réelles que celui utilisé par notre approche. Dans le même ordre de test, nous avons aussi adopté et implémenté à nos besoins d'expérimentation, l'algorithme génétique proposé par Boussedjra et al [8] pour résoudre le problème de plus court chemin intermodal.

Pour la comparaison, nous utiliserons les indicateurs de performances C et S afin d'évaluer les ensembles de solutions non dominées trouvés par chaque algorithme sur les différentes instances générées.

Le tableau 3.1 présente les paramètres spécifiques à l'OCF, ces paramètres ont été fixés après une série de tests numérique. Les paramètres de l'algorithme génétique sont ceux utilisés dans [8].

$tmax$	m	α	β	ρ	$[\tau_{min}, \tau_{max}]$	Δ
200	100	2	1	0.9	$[0.007, 6]$	0.05

TAB. 3.1 – Paramètres OCF adoptés pour le test.

3.6.3 Résultat et discussion

Les résultats obtenus, pour les métriques C et S sur les différentes instances sont répertoriés dans les tables 3.2, 3.3, 3.4 et 3.5. Pour chaque problème et pour chaque algorithme considéré, les tableaux donnent le profit trouvé : la valeur moyenne (Avg), La meilleure (Best) et la pire valeur (Worst) pour chaque métrique.

problème	C(Algo-Doerner/PM-OCF)			C(PM-OCF/Algo-Doerner)		
	Best	Avg	Worst	Best	Avg	Worst
P1(30,3,20055)	0	0	0	1	0.9	0.66
P2(100,3,165675)	0	0	0	1	1	1
P3(100,3,81420)	0	0	0	1	1	1
P4(60,3,45585)	0	0	0	1	0.876	0.818
P5 (100,3,149625)	0	0	0	1	1	1

TAB. 3.2 – Métrique C : Comparaison PM-OCF/Algo-Doerner

problème	C(Algo-Pinto/PM-OCF)			C(PM-OCF/Algo-Pinto)		
	Best	Avg	Worst	Best	Avg	Worst
P1(30,3,20055)	0	0	0	0.14	0.035	0
P2(100,3,165675)	0.28	0.106	0	0.4	0.318	0.28
P3(100,3,81420)	0.4	0.19	0	0.66	0.331	0
P4(60,3,45585)	0	0.	0	0.125	0.031	0
P5 (100,3,149625)	0	0	0	0.75	0.3	0

TAB. 3.3 – Métrique C : Comparaison PM-OCF/Algo-Pinto.

problème	C(Algo-Boussedjra/PM-OCF)			C(PM-OCF/Algo-Boussedjra)		
	Best	Avg	Worst	Best	Avg	Worst
P1(30,3,20055)	0	0	0	0.215	0.043	0
P2(100,3,165675)	0.25	0.051	0	0.4	0.2	0
P3(100,3,81420)	0.4	0.21	0	0.66	0.446	0.28
P4(60,3,45585)	0.2	0.108	0	1	0.525	0.3
P5 (100,3,149625)	0.33	0.094	0	1	0.785	0.5

TAB. 3.4 – Métrique C : Comparaison PM-OCF/Algo-Boussedjra.

Concernant la mesure C :

- En observant les résultats du tableau 3.2, nous pouvons constater que la PM-OCF est toujours meilleure que l'approche proposée par Doerner. En effet, les résultats de C(PM-OCF/Algo-Doerner) sont supérieurs à 0.876 en moyenne pour chaque instance traitée, ce qui signifie que 87,6% de solutions Pareto trouvées par l'algorithme de Doerner sont dominées par celles trouvées par l'algorithme PM-OCF. Les résultats de C (PM-OCF/Algo-Doerner) égale à 0 pour toutes les instances, c'est-à-dire, aucune solution Pareto engendrée par PM-OCF n'est dominée par une solution trouvée par Algo-Doerner.

- En observant les résultats du tableau 3.3, Nous constatons que les deux variantes se dominent mutuellement. Cependant, la PM-OCF est meilleure que l'approche proposée par Pinto. En effet, pour les trois instances P2, P3 et P5, le rapport de dominance de PM-OCF dépasse Algo-Pinto par 30% en moyenne. Pour les autres instances la différence est moindre, mais toujours à l'avantage de PM-OCF.

- En observant les résultats du tableau 3.4, nous pouvons prendre les mêmes remarques que celles faites ci-dessus. PM-OCF donne de meilleurs résultats. En effet, pour les deux instances P4 et P5, le rapport de dominance de PM-OCF dépasse Algo-Boussedjra par 52% en moyenne. De même, ce facteur est égal à 44,6% pour P3. En observant les meilleures valeurs de C, toutes les solutions trouvées par Algo-Boussedjra sont dominées par celles trouvées par PM-OCF pour les deux instances P4 et P5.

La métrique S tient compte en partie de la diversité du front. Les calcules effectués pour cette métrique (TAB 3.5) confirment ceux réalisés pour la métrique C. En plus, la métrique S permet de quantifier plus précisément les différences entre les ensembles Pareto obtenus par les quatre algorithmes. En effet, Algo-Pinto et PM-OCF sont plus stable, c'est le cas pour les trois instances P2, P3 et P4. Algo-Dorner donne de meilleurs résultats uniquement sur l'instance P1, ceci s'explique par la

problème	S(Algo-Pinto)			S(Algo-Doermer)			S(PM-OCF)			S(Algo-Boussedjra)		
	Best	Avg	Worst	Best	Avg	Worst	Best	Avg	Worst	Best	Avg	Worst
P1(30,3,20055)	30.20	181.76	419.91	16.57	67.89	215.93	22.16	103.75	221.8	22.83	154.93	456.7
P2(100,3,165675)	70.75	83.42	87.65	51.75	103.18	144.5	70.75	80.91	87.65	86.56	249.13	398.299
P3(100,3,81420)	99.84	106.33	107.8	101.39	123.82	181.5	10.02	77.42	105.93	19.32	119.32	431.47
P4(60,3,45585)	63.72	66.132	72.57	39.49	73.71	110.10	55.07	65.57	78.78	20.24	56.68	100.43
P5 (100,3,149625)	34.84	42.68	75.10	35.89	68.94	105.72	23.61	30.82	35.80	37.24	48.98	77.03

TAB. 3.5 – Métrique S : Comparaison entre Algo-Pinto, Algo-Doerner, Algo-Boussedjra et PM-OCF.

taille de l'ensemble de solutions Pareto trouvées par l'Algo-Doerner pour cette instance (4 solutions au maximaux). En effet, la mesure S ne permet pas réellement de quantifier la différence entre les ensembles Pareto qui se distinguent par une grande différence dans leurs tailles.

Afin de bien visualiser ces résultats, la figure 3.2 représente les fronts Pareto trouvés par les quatre algorithmes sur l'instance P3, nous pouvons remarquer que PM-OCF donne le meilleur front en terme de convergence et diversité.

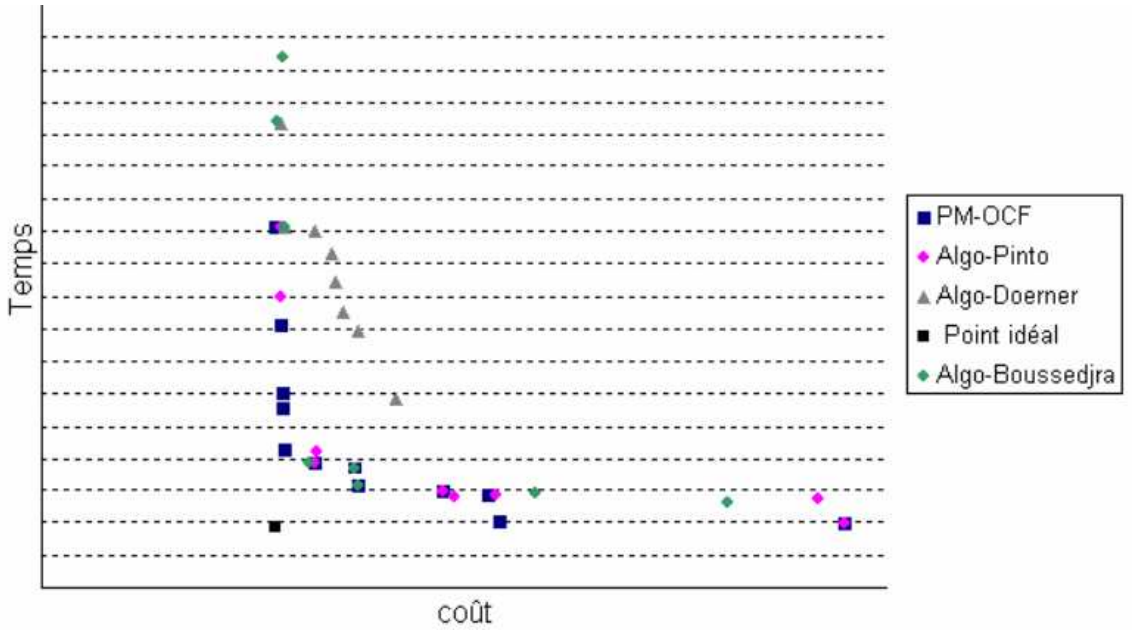

FIG. 3.2 – Comparaison entre Algo-Pinto, Algo-Doerner, Boussedjra-Algo et PM-OCF.

3.7 Conclusion

Dans ce chapitre, nous avons proposé un nouvel algorithme OCF avec une approche Pareto pour l'optimisation multi-objectif (PM-OCF). Nous nous sommes appuyés sur la notion d'hétérogénéité, où chaque fourmi donne une importance relative

aux critères d'optimisation différemment pour qu'elles soient capables de diversifier la recherche dans l'espace des solutions réalisables. A cette effet, nous avons proposé un moyen de calculer dynamiquement les poids pendant la résolution du problème. La méthode de calcul proposée permet d'obtenir des poids qui convergent vers le point idéal. Elle offre également la capacité de traiter un nombre quelconque d'objectifs et d'assurer un compromis important entre les objectifs considérés.

L'application de PM-OCF a été réalisée pour résoudre la variante multi-objectif du problème de transport étudié dans ce mémoire. La variante considérée nécessite d'optimiser les déplacements selon trois objectifs conflictuels : (1) le temps du trajet, (2) les temps d'attentes et (3) le coût total de transport.

Les résultats expérimentaux ont montré l'efficacité de l'approche PM-OCF par rapport aux autres algorithmes OCF hétérogènes proposés dans [15, 47]. Les ensembles Pareto obtenus par PM-OCF étant généralement meilleurs et plus diversifiés. Nous avons aussi comparé notre approche avec l'algorithme génétique proposé par Boussedjra et al [8] pour résoudre le problème du plus court chemin intermodal. Les résultats obtenus sont encore une fois probants.

Chapitre 4

Bi-colonie pour l'optimisation d'un déplacement multimodal en mode perturbé

4.1 introduction

La prise en compte des perturbations dans un réseau de transport collectif multimodal constitue un enjeu important d'un point de vue opérationnel, car les passagers ne disposent d'aucune information sur le trafic et les congestions au niveau du réseau, ces informations sont difficiles à mettre en oeuvre pour de nombreuses raisons : organisationnelles, économiques, juridiques et techniques,... [53]

Plusieurs types de perturbations peuvent affecter la régularité du trafic, en général, une perturbation concerne un ensemble de stations dans une zone ou bien un ensemble de modes de transport. Afin de réagir en cas de perturbation, la gestion en temps réel est très importante. Dans ce cas, un processus de régulation est réalisé afin de contourner ou d'amortir ces perturbations, qui peuvent aussi apparaître simultanément, ce qui constitue un autre axe de recherche dans le domaine du transport. Ce chapitre s'inscrit dans l'optique de venir en aide au régulateur, en lui proposant deux populations de solutions, la première population contient un ensemble de solutions meilleures selon le choix des usagers et qui lui convient dans le fonctionnement normal du réseau, la deuxième population renferme les solutions de secours en cas de perturbation.

Le chapitre est organisé en quatre parties, la première partie est consacrée au

contexte de la régulation, afin de montrer la nécessité des systèmes d'aide à la régulation. La deuxième partie du chapitre présente le principe de base que nous avons utilisé dans notre contribution. La troisième partie est consacrée à la présentation de la méthode de colonie de fourmis proposée pour optimiser l'itinéraire multimodal en mode perturbé. Enfin, les résultats expérimentaux obtenus par l'exécution des algorithmes proposés sont présentés dans la section 5.

Le travail présenté dans ce chapitre a été accepté pour être publié et présenté à LT2009(Workshop International : Logistique et Transport 2009) [32]

4.2 Contexte de la régulation

Le processus de régulation contient plusieurs tâches difficiles allant de la détection des perturbations à la prise de décision [6], le régulateur doit constamment faire face à tous les incidents en temps réel, il doit prendre, en un temps limité, des décisions immédiates pour traiter ces incidents.

Dans [6], [54], les auteurs divisent le processus de régulation en deux modules : Systèmes d'Aide à l'Exploitation (SAE), Système d'Aide à la Régulation (SAR). Les SAE permettent le suivi en temps réel de l'exploitation d'un réseau de transport et traitent des quantités très importantes d'informations relatives au réseau. La surveillance en temps réel permet de détecter plus rapidement les perturbations, cette détection s'effectue essentiellement par la comparaison entre les horaires réels des voyages en cours et les horaires définis dans le tableau de marche théorique. Ensuite, chaque perturbation est identifiée par plusieurs paramètres (la source de perturbation, la cause de perturbation,...). Le SAR commence par une analyse rapide de l'incident identifié. Il s'agit d'un premier diagnostic pouvant donner une estimation de l'impact de la perturbation afin d'évaluer leur gravité, puis le SAR propose au régulateur des actions ainsi que leurs effets sur l'état actuel du réseau. Ce dernier (le régulateur) choisit une action et le SAR effectue l'implémentation.

Le régulateur se trouve parfois devant des cas de perturbations très difficiles, ceci explique l'importance d'améliorer les services de SAR. L'objectif de ce chapitre est donc de développer un SAR, nous voulons proposer au régulateur pendant le voyage plusieurs solutions pour l'aider à choisir la plus adéquate en cas d'une perturbation. Ainsi, la tâche du régulateur sera simplifiée, ce qui améliorera la qualité du service rendu aux voyageurs. La figure 4.1 illustre plus concrètement les étapes de base du processus de régulation.

Malgré l'importance des SAE et SAR, la littérature ne rapporte que très peu de travaux, parmi les approches de résolution proposées dans ce domaine, on peut citer : les algorithmes génétique [53, 11], les systèmes multi-agent [40] et la théorie de la logique floue [45] .

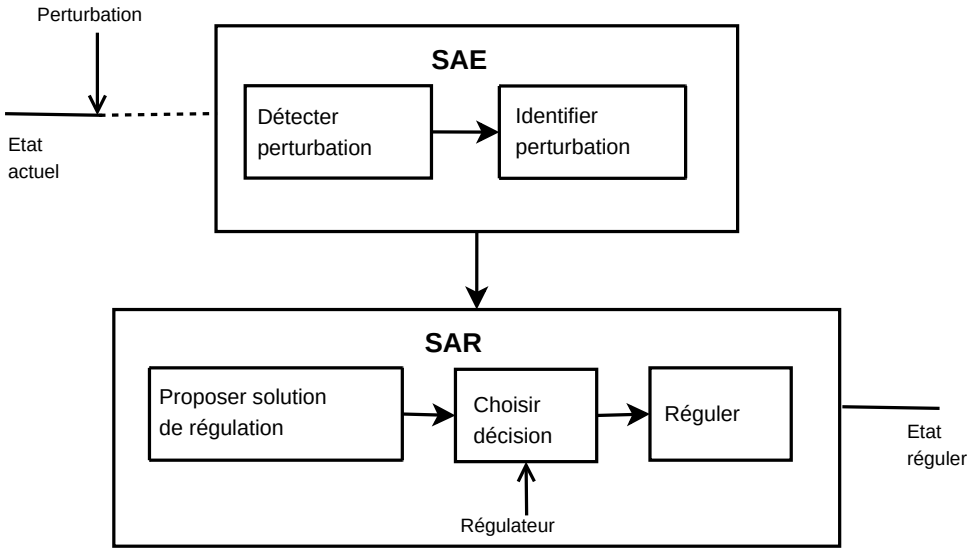

FIG. 4.1 – Le déroulement du processus de régulation en temps réel

4.3 Principe de résolution adopté

Pour gérer les cas de perturbations, nous avons besoin d'une population finale dont les solutions sont dispersées (diversifiées) sur toute la zone et utilisant le plus de modes possibles. Autrement dit, si nous disposons de solutions robustes représentatives des différentes régions et des différents modes de transport, nous augmentons aussi la probabilité de trouver une solution de secours en temps réel lors des perturbations. Sur ce principe Zidi et Hammadi [53] ont développé une méthode d'optimisation génétique qui permet de proposer plusieurs solutions au régulateur pour prendre rapidement la meilleure décision en cas de perturbation. La diversification des solutions est assurée par deux variables A_{ij}, M_i permettent de contrôler les opérateurs de croissement génétique. A_{ij} est le nombre d'arcs communs entre deux solutions i et j. Minimiser cette variable permet d'augmenter l'exploration des solutions sur la zone de transport. La deuxième variable M_i est le nombre de modes utilisés dans une solution i, la maximisation des M_i permet d'améliorer la population en garantissant la multimodalité des solutions ainsi que la diversification.

L'inconvénient majeur de la stratégie de Zidi et Hammadi [53] est d'éliminer arbitrairement de nombreuses solutions optimales, qui pourraient pourtant s'avérer intéressantes. Comme, il est plus difficile d'établir en même temps la diversification et l'intensification des solutions dans la population finale, notre proposition OCF sépare le contrôle de l'intensification et la diversification entre deux colonies de fourmis (C1, C2). La colonie C1 construit ses solutions en favorisant l'intensification et la colonie C2 en favorisant la diversification. De cette manière, nous pouvons confirmer que le principal apport de notre contribution par rapport à l'approche de Zidi et Hammadi [53] réside dans la qualité des solutions proposées au régulateur.

4.4 Approche Bi-colonie d'aide à la régulation

4.4.1 Système d'optimisation proposé : stratégie de résolution

L'objectif principal n'est pas de définir une seule solution d'une perturbation, mais de rechercher un ensemble de meilleures solutions possibles. L'adaptation de ce principe par la méthode d'optimisation par colonie de fourmis, repose sur une coopération de deux colonies (C1, C2) qui s'exécutent séquentiellement l'une après l'autre (C1 puis C2), tel que : C1 propose une population (P1) de k meilleures solutions, ce qui permet à l'utilisateur de choisir celle qui lui convient le mieux en cas de perturbation lors de son passage. Malheureusement, il n'existe aucun mécanisme de recherche qui permet de diversifier les solutions trouvées par cette première colonie, c'est à dire, il y a une possibilité ou une perturbation peut affecter toutes les solutions proposées par C1, c'est à dire, aucune solution de P1 ne convient pour continuer le trajet prévu, pour surmonter ce problème, nous avons ajouté une deuxième colonie C2.
C2 une colonie utilise un mécanisme de diversification qui évite les solutions de C1, afin de proposer une autre population (P2), appelée solutions de secours. Autrement dit, en cas de perturbation qui touche toutes les solutions de P1, cette population ne représentera habituellement plus de solutions valides à la nouvelle instance du problème, et la meilleure solution pour la nouvelle instance peut être trouvée facilement et rapidement à partir des solutions de secours proposées par C2.

4.4.2 Contribution à la résolution : deux versions d'algorithmes

Dans cette section, nous décrivons en détaill comment adapter le principe de Bi-colonie aux deux approches : Algorithme bi-objectif d'optimisation du temps de transport (OCF2-PCC-RTM du chapitre 2) et l'algorithme Pareto Multi-objectif (PM-OCF du chapitre 3).

Les deux algorithmes OCF2-PCC-RTM et PM-OCF adaptent cette stratégie de résolution du problème de perturbation en assurant les deux points suivants :

1. Un mécanisme de diversification de recherche entre les deux colonies C1 et C2.
2. Les meilleures solutions de chaque colonie sont conservées dans une population finie.

En effectuant ces deux modifications aux OCF2-PCC-RTM et PM-OCF, nous obtenons deux autres versions notées Bi-OCF2-PCC-RTM et Bi-PM-OCF qui seront présentées dans la suite de ce chapitre.

4.4.2.1 La diversification de recherche entre les deux colonies C1 et C2

La deuxième colonie augmente la probabilité de trouver une solution de secours en temps réel lors des perturbations qui surviennent dans le réseau de transport, ceci sera possible grâce à la diversification de recherche de la population finale de C2 par rapport à celles proposées par C1. L'exécution séquentielle des deux colonies facilite cette tâche de diversification, elle est favorisée dans l'étape d'initialisation, et consiste à décroître la trace de phéromone qui correspond à la population de solutions P1, dans le but de pousser la recherche vers les autres zones non visitées par la première colonie et vers l'utilisation d'autres modes de transport pour augmenter la chance de trouver une solution de secours en cas de perturbation.

Le mécanisme de diversification proposé est appliqué de la même façon dans les deux versions Bi-OCF2-PCC-RTM et Bi-PM-OCF.

4.4.2.2 Les meilleures solutions sont conservées dans une population finie

1. Bi-Colonie pour l'optimisation du temps de transport

Guntsch et Middendorf [31] proposent une variante appelée P-ACO (A population based approach for ACO), qui permet de trouver une population de solutions, cette approche est bien adaptée pour la résolution des problèmes d'optimisation dynamique.
En se basant sur l'approche PACO, la génération des solutions fonctionne comme dans l'algorithme OCF2-PCC-RTM, telle que :

- On maintient une population de solutions (P) de k individus.
- la meilleure solution de P est appelée p^*, tandis que la mauvaise solution est appelée p^- .
- Chaque nouvelle solution p^+ est entrée dans la population P, si $p^+ \notin P$.
- Après k générations, on a exactement k solutions dans la population, à la $k+1^{eme}$ $(i > 0)$ génération, la nouvelle solution p^+remplace la mauvaise solution p^- de la population P si seulement si $p^+ \notin P \; et \; T_{SD}(p^+) < T_{SD}(p^-)$.
- La mise à jour de la mémoire de phéromone est effectuée à chaque itération par la meilleure solution p^*.

La différence principale entre notre approche et la stratégie P-ACO, réside dans l'égalité des individus de la population finale de P-ACO, en outre, il n'y a pas d'évaporation de la trace de phéromone à chaque cycle.

L'algorithme 4.1 montre le pseudo-code détaillé des deux colonies Ci (i=1,2) de l'approche Bi-OCF2-PCCM. Les quatre méthodes : *formis-initialisation()*, *construction-solution()*, et *update-phéromone()* sont définis de la même manière que celles présentées en chapitre 2.

Algorithm 4.1 Bi-OCF2-PCC-RTM(i),où i=1,2 ;

Début
Initialisation des paramètres OCF
initialisation-phéromonneCi() i=numéro de la colonie
tantque $(t < tmax)$ **faire**
  pour $k = 1, \ldots, m$ **faire**
    formis-initialisation()
    $p^+ \leftarrow$construction-solution()
    si $(p^+ \notin P_i)$ **alors**
      si $(\mid P_i \mid < k)$ **alors**
        Ajouter p^+ dans P_i
      sinon
        si $(T_{SD}(p^+) < T_{SD}(p^-))$ **alors**
          Remplacer p^- par p^+ dans P_i
        finsi
      finsi
    finsi
  fin pour
  update_phéromone()
fin tantque
Fin

Ainsi, afin d'éviter les solutions proposées par C1 (assurer la diversification), nous avons introduit une modification dans la méthode d'initialisation de la mémoire de phéromone de la deuxième colonie, comme montrée par l'algorithme 4.2 noté *initialisation-phéromone-C2()*. Notons que les pistes de phéromone de la première colonie sont initialisées à leur valeur maximale τ_{max}.

Algorithm 4.2 initialisation-phéromone-C2()

pour chaque arête $(i, j) \in A$ **faire**
  pour chaque mode $x \in X$ **faire**
    pour chaque date $t \in T$ **faire**
      si $(composante(i, j, x, t)) \in P1$ **alors**
        $\tau_{ij(x,t)} \longleftarrow \varepsilon$
        $\{\varepsilon$ petit nombre $\neq 0\}$
      sinon
        $\tau_{ij(x,t)} \longleftarrow \tau_{max}$
      finsi
    fin pour
  fin pour
fin pour

2. Bi-Colonie pour l'optimisation Pareto-Multi-Objectif

Il est facile de définir une population de solutions P dans l'algorithme PM-OCF, puisque celui-ci fournit un ensemble de solutions connu comme l'ensemble des

solutions Pareto optimal Q. La définition de la population P varie selon la taille de l'ensemble Q :

$$P = \begin{cases} Q & \mid Q \mid \geq k \\ Q \cup R & Sinon \end{cases} \qquad (4.1)$$

Dans le deuxième cas, il faut compléter l'ensemble Pareto Q par un ensemble R qui contient $k - \mid Q \mid$ solutions dominées, ici nous proposons de construire l'ensemble R par la sélection des solutions les plus proches de la frontière Pareto. cette sélection est effectuée grâce à une fonction $d(s)$ qui peut être définie dans l'espace de solutions réalisables en exploitant le concept de dominance Pareto comme suit :

$$d(s) = \mid \forall t \in Q \;\; t > s \mid \qquad (4.2)$$

Rappelons que $t > s$ signifie que la solution t domine la solution s.

Afin de mieux expliquer le principe de base de cette fonction, considérons l'exemple de minimisation de deux objectifs F1 et F2 de la figure 4.2. Dans cet exemple, les huit solutions s_i $(i = 1, \ldots, 8)$ sont représentées dans le plan défini par les deux fonctions F1 et F2, le front Pareto de ce plan est défini par quatre solutions. L'évaluation de la fonction $d(s)$ des solutions non dominées au sens de Pareto est indiquée dans le tableau de la même figure.

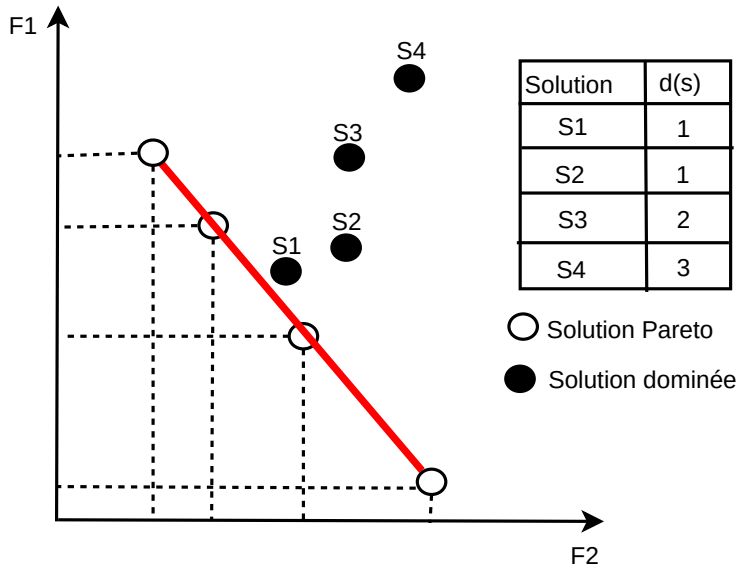

Fig. 4.2 – Classement des solutions dominées en utilisant la fonction d(s)

En observant la figure 4.2, nous pouvons constater que :

– La méthode de calcul proposée permet d'affecter aux meilleures solutions dominées la valeur du $d(s)$ la plus petite. Ainsi, les solutions ayant la plus petite

valeur de $d(s)$ auront plus de chances d'être choisies dans la construction de l'ensemble R.

- $d(s_i) = d(s_j) \nRightarrow s_i <> s_j$, c'est le cas par exemple pour les deux solutions s_1 et s_2 ($d(s_1) = d(s_2)$, mais s_1 domine s_2).

Le pseudo-code de l'approche Bi-PM-OCF des deux colonies Ci (i=1,2) est identique à celui que nous avons présenté dans le chapitre précédent (section 2.3), sauf que nous avons ajouté une procédure pour mettre à jour la population de solutions P. En premier lieu, pour chaque solution fournie s, les relations de dominance sont vérifiées pour construire l'ensemble Pareto Q. la deuxième phase de cette procédure consiste à utiliser l'ensemble Q pour ne conserver dans P que les solutions Pareto et les meilleures solutions dominées (ensemble R) identifiées par la méthode de classement proposée. Rappelons que nous avons utilisé le même mécanisme de diversification entre les deux colonies C1 et C2 que celui utilisé dans l'approche Bi-OCF2-PCC-RTM.

Algorithm 4.3 update-population(Q, s, R)

Début
$N \leftarrow \emptyset$
si s n'est pas dominée par $\forall t \in Q$ **alors**
  $N \leftarrow Q - \{\forall t \in Qs > t\}$
  $Q \leftarrow Q - N \cup \{s\}$
sinon
  $R \leftarrow R \cup \{s\}$
finsi
si $|Q| < k$ **alors**
  $R \leftarrow R \cup N$
  si $|R| + |Q| < k$ **alors**
    classement(R)
    supprimer les $|R| + |Q| - k$ derniers solutions de R
  finsi
sinon
  $R \leftarrow \emptyset$
finsi
$Q \leftarrow Q \cup R$
Fin

Algorithm 4.4 update-population(R)

Début
Les solutions de l'ensemble R sont triées dans un ordre croissant en fonction de la valeur de $d(s)$.
Les solutions ayant la même valeur de $d(s)$ sont triées en ordre croissant en fonction de l'évaluation de la somme $\sum_{i \leq |F|} f_i(s)$
Fin

La procédure de classement décrite dans l'algorithme 4.4 est appliquée pour trier les solutions dominées afin de construire l'ensemble R.

En observant le pseudo-code de l'algorithme 4.4, nous pouvons remarquer que la deuxième étape de classement permet de trier les solutions ayant la même valeur de $d(s)$ en évitant le cas suivant :

$$d(s_i) = d(s_j)\ et\ s_i > s_j \Longrightarrow s_i\ est\ classee\ avant\ s_j \qquad (4.3)$$

Ce cas est impossible de se produire puisque l'équation 4.4 est toujours valide dans la procédure de classement proposée.

$$\sum_{t \leq |F|} f_t(s_i) < \sum_{t \leq |F|} f_t(s_j) \Longrightarrow s_j \not\succ s_i \qquad (4.4)$$

4.5 Résultats expérimentaux

Pour valider notre approche, plusieurs tests sont réalisés sur des graphes de 60 et 100 nœuds, le nombre de modes disponibles est fixé à trois. Cinq problèmes (P1,...,P5) sont construits selon les points de départ et destination sélectionnés dans ces graphes, pour lesquels nous avons examiné plusieurs tests.

Les tableaux 4.1 et 4.2 présentent, respectivement, les paramètres spécifiques à l'OCF des deux colonies (C1 et C2) utilisés pour tester les deux approches proposées dans ce chapitre. Ces paramètres ont été fixés après une série de tests préliminaires appliqués sur des réseaux aléatoires de différentes tailles. Les paramètres de la 2^{eme} colonie doivent être choisis afin d'éviter la convergence vers la population de solutions proposée par la 1^{ere} colonie. Les deux paramètres les plus sensibles pour contrôler la convergence d'un algorithme sont le nombre maximum de cycles $tmax$ et le paramètre d'évaporation ρ

pr	$tmax$	M	α	β	ρ	$[\tau_{min},\tau_{max}]$	Δ	K
C1	60	60	2	1	0.9	[0.007,6]	0.05	15
C2	30	60	2	1	0.6	[0.007,6]	0.05	15

TAB. 4.1 – Paramètres OCF adoptés pour tester l'algorithme Bi-OCF2-PCC-RTM

pr	t_{max}	M	α	β	ρ	$[\tau_{min},\tau_{max}]$	Δ	K
C1	150	80	2	1	0.9	[0.007,6]	0.005	15
C2	60	60	2	1	0.6	[0.007,6]	0.05	15

TAB. 4.2 – Paramètres OCF adoptés pour tester l'algorithme Bi-PM-OCF

La détermination de la taille de la population k est très spécifique aux caractéristiques du problème traité, en particulier aux difficultés rencontrées pour produire

des solutions faisables pour faire face aux perturbations. Dans cette expérimentation, une valeur limite admissible de la taille de la population est de 15 solutions pour chaque colonie. En dessous de cette valeur, un manque de diversification est constaté.

Notre système propose au client la meilleure solution trouvée par l'algorithme Bi-colonie. A ce niveau, on distingue deux types de perturbations : une perturbation se produit avant le début de voyage et une autre pendant le voyage. Le premier type de perturbation demande juste de rechercher une autre solution admissible dans la population de solutions, pour le deuxième cas, on doit déterminer tout d'abord la position courante du client. Le régulateur se charge de déduire les meilleurs scénarii possibles qui permettent au client de continuer son déplacement sans problème vers le nœud prévu.

Après avoir choisi une solution finale (la meilleure solution), nous générons des perturbations aléatoires pour calculer les chances de trouver une solution alternative dans les deux populations proposées par l'exécution de l'algorithme Bi-colonie. Notant P la probabilité de trouver une solution alternative en cas de perturbations.

Après plusieurs scénarii de perturbations générées aléatoirement, nous avons obtenu des statistiques montrant l'évolution de la probabilité moyenne (avec et sans) considération des solutions trouvées par la deuxième colonie. Les résultats obtenus sont présentés dans les tableaux 4.3, 4.4.

		P1	P2	P3	P4	P5
C1	P	80%	60%	60%	90%	80%
C1,C2	P	100%	96.33%	96%	98%	96%

TAB. 4.3 – Résultats de tests obtenus (perturbation pendant le voyage) de l'approche Bi-OCF2-PCC-RTM

		P1	P2	P3	P4	P5
C1	P	96%	80%	65.25%	90%	90%
C1,C2	P	100%	96%	95%	100%	98%

TAB. 4.4 – Résultats de tests obtenus (perturbation pendant le voyage) de l'approche Bi-PM-OCF

Ces tableaux montrent que les probabilités de trouver une solution alternative en cas d'une perturbation sont plus élevées si l'on considère les solutions trouvées par la 2^{eme} colonie, ce qui explique l'intérêt de cette colonie en terme de diversification. Notons que dans le cas où une perturbation se produit avant le début de voyage, il existe toujours une chance (100%) de trouver une autre solution de secours. ce qui n'est pas le cas (P = 80%) si on s'appuie seulement sur la 1^{ere} colonie.

4.6 Conclusion

La régulation du trafic en temps réel des réseaux de transport multimodal est une tâche de plus en plus complexe qui nécessite l'élaboration de système d'aide à la régulation (SAR). Dans ce but, nous avons commencé dans ce chapitre par expliquer le contexte de régulation en démontrant la nécessité des SAR. Nous avons ensuite présenté le principe de l'algorithme génétique de Zidi et Hammadi [53], l'inconvénient majeur de cette stratégie réside dans la difficulté d'établir en même temps la diversification et l'intensification des solutions dans la population finale. Pour pallier à cet inconvénient, nous avons proposé une approche d'optimisation par colonie de fourmis adaptée au problème d'aide à la régulation, Il s'agit d'une nouvelle idée de recherche d'itinéraire, qui s'appuie sur une coopération de deux colonies pour générer une population de chemins optimums et diversifiés sur toute la zone de transport. Dans ce cadre, nous avons modifié les deux algorithmes OCF2-PCC-RTM et PM-OCF proposés dans les deux chapitres précédents selon notre stratégie de Bi-colonie, deux nouvelles versions ont été présentées dans ce chapitre. Les résultats obtenus par ces deux versions montrent l'efficacité et l'intérêt pratique de notre approche.

Conclusion et Perspectives

DANS ce mémoire, nous avons traité le problème du déplacement dans les réseaux de transport multimodal. Ce problème préoccupe de plus en plus les compagnies de transport, car il a un impact direct sur la qualité du service offerte aux usagers. Le contexte dans lequel s'inscrit notre étude correspond à l'une des variantes les plus complexes du problème. Nous avons orienté nos travaux vers l'optimisation de plusieurs objectifs, à partir de données dynamiques relatives à de nombreux modes de transport. Ceci a mis en évidence la nécessité de développer des algorithmes d'optimisation approchés capables de traiter ces problèmes avec des temps de réponse raisonnables.

L'objectif global de ce travail est de proposer un algorithme d'optimisation par colonie de fourmis pour la résolution du problème de déplacement multimodal, comprenant des éléments adaptés spécifiquement pour ce dernier. Dans ce contexte, nous avons abordé et étudié le problème du plus court chemin sous plusieurs aspects. Ainsi, plusieurs contributions ont été apportées.

Nos travaux ont débuté par une étude bibliographique consacrée essentiellement à l'optimisation par colonie de fourmis et le problème du plus court chemin multimodal. Une classification des différentes variantes du problème et des modèles graphiques pour la représentation du réseau est présentée. Enfin, nous avons proposé une implémentation objet du modèle retenu.

Nous avons adopté une méthode progressive dans la présentation des approches de résolutions proposées. Notre contribution est ainsi exposée dans ce manuscrit sous la forme de trois chapitres complémentaires. Le chapitre 2 introduit, dans un premier temps deux méthodes OCF monocritère adaptée à la résolution du problème de plus court chemin multimodal. Ensuite, nous avons proposé deux variantes pour l'optimisation bi-objectif du temps de transport. Ces approches nous permettent de trouver une seule solution optimale. De ce fait, elles ne seraient sans doute pas exploitables pour la variante multi-objectif du problème de transport considéré. Ceci mettait en évidence la nécessité de développer des approches Pareto pour répondre à ce besoin. C'est pourquoi nous avons proposé dans le chapitre 3 un nouvel algorithme OCF avec une approche Pareto (PM-OCF) pour l'optimisation multi-objectif. Le principal apport de cette approche réside dans le calcul des poids. La méthode de calcul proposée se base sur l'échange dynamique des poids en fonction de la distance

normalisée entre l'ensemble des solutions Pareto optimales et le point idéal pour chaque objectif. Ensuite, le PM-OCF a été appliqué avec succès pour résoudre la variante multi-objectif du problème de transport étudié. Dans le dernier chapitre, nous avons proposé un système interactif d'aide aux déplacements en mode dégradé de fonctionnement du réseau de transport. Dans ce but, nous avons proposé une nouvelle approche OCF qui s'appuie sur une coopération de deux colonies de fourmis. Cette approche nous a permis de trouver des solutions optimisées à présenter aux voyageurs et d'avoir une autre population de solutions à utiliser en cas de perturbations.

Les perspectives de recherche que nous pouvions donner concernent les améliorations et les validations des résultats de nos algorithmes.

En premier lieu, il sera intéressant de continuer à travailler au niveau de la recherche d'un compromis acceptable entre la diversification et l'intensification dans les approches proposées. En effet, il reste sans doute, notamment d'envisager d'autres informations heuristiques, sous formes de paramètres supplémentaires spécifiques au problème de transport multimodal. Deuxièmement, nous souhaitons étudier l'hybridation avec d'autres méthodes de recherche locale à partir de chemins identifiés comme intéressants.

Dans nos expérimentations, nous ne sommes pas parvenus à obtenir un descriptif complet de données réelles de test. Il est donc important d'évaluer les performances des algorithmes proposés sur des réseaux réels de transport.

Afin de mieux valider les résultats de l'approche PM-OCF , il serait révélateur de les tester sur d'autres problèmes que le problème de transport multimodal. En effet, cette approche a été définie de manière générique, de telle sorte qu'elle puisse être facilement appliquée à d'autres problèmes. Il serait donc intéressant de tester aussi bien l'approche PM-OCF sur d'autres problèmes multi-objectif.

Enfin, le système d'aide à la régulation proposé se trouve parfois devant des cas de perturbations simultanées ou très compliquées. Dans ces cas, il serait obligatoire de reconfigurer totalement ou partiellement la planification du réseau de transport.

Bibliographie

[1] I. Alaya, C. Solnon, and K. Ghédira. Ant colony optimization for multi-objective optimization problems. In *19th IEEE International Conference on Tools with Artificial Intelligence(ICTAI'07)*, pages 450–457, 2007.

[2] V. Bachelet, Z. Hafidi, P. Preux, and E. G. Talbi. Vers la coopération des métaheuristiques. *Calculateurs parallèles*, 9(2), 1998.

[3] M. Basseur. *conception d'algorithme coopératifs pour l'optimisation Multiobjectif : application aux problèmes d'ordonnancement de type flow-shop.* PhD thesis, USTL, 2005.

[4] R. E. Bellman. On a routing problem. *In Quart. Appl. Math*, 16 :87–90, 1958.

[5] C. Blum and A. Roli. Metaheuristics in combinatorial optimization : Overview and conceptual comparison. *ACM Computing Surveys*, 35(3) :268–308, 2003.

[6] K. Bouamrane, T. Bonte, M. Sevaux, and C. Tahon. Sart : un système d'aide a la décision pour la régulation d'un réseau de transport bimodal. In *Proceedings of the workshop Méthodologies et Heuristiques pour l'Optimisation des Systèmes Industriels, MHOSI 2005*, Hammamet ,Tunisie, 2005.

[7] M. Boussedjra. *Contribution à la résolution du problème de plus court chemin multiobjectif par algorithme évolutionnistes : Application aux systèmes de transport intermodal.* PhD thesis, Université de Franche-Comté et Université de Technologie de Belfort-Montbéliard, 2005.

[8] M. Boussedjra, C. Bloch, and A. El-Moudni. Single and multi-objective optimization using evolutionary algorithms. In *17ème Congrès Mondial IMACS Calcul Scientifique, Mathématiques Appliquées et Simulation*, Paris, France, 2005.

[9] M. Boussedjra, C. Bloch, and A. El-Moudni. Une fonction multiobjectif pour le problème de plus court chemin intermodal. In *IEEE Conférence International Francophone en Automatique*, Douz, Tunisie, 2005.

[10] B. Bullnheimer, R. F. Hartl, and C. Strauss. A new rank based version of the ant system. *A Computational Study. Central European Journal for Operations Research and Economics*, 7(1) :25–38, 1999.

[11] B. F. Chaar and S. Hammadi. Régulation spatio-temporelle d'un réseau de transport multimodal. *e-STA -Sciences et Technologie de l'Automatique*, 2, 2005.

[12] D. T. Connolly. An improved annealing scheme for the QAP. *European Journal of Operational Research*, 46(1) :93–100, 1990.

[13] P. Delisle. Parallélisation d'un algorithme d'optimisation par colonies de fourmis pour la résolution d'un problème d'ordonnancement industriel, 2002. Mémoire de maîtrise, in Département d'Informatique et de Mathématique, Université du Québec à Chicoutimi.

[14] D. Dijkstra. A note on two problems in connexion with graphs. *Numerische, Mathematik*, 1 :269–271, 1959.

[15] K. F. Doerner, W. J. Gutjahr, C. Hartl, R .F.and Strauss, and C. Stummer. Pareto ant colony optimization with ilp preprocessing in multiobjective project portfolio selection. *European Journal of Operational Research*, 171 :830–841, 2006.

[16] M. Dorigo. *Optimization learning and natural algorithm.* PhD thesis, Ecole polytechnique de Milano, Italie, 1992.

[17] M. Dorigo, M. Birattari, and T. Stützle. Ant colony optimization– artificial ants as a computational intelligence technique. *IEEE Computational Intelligence Magazine*, 2006.

[18] M. Dorigo and L. M. Gambardella. Ant colony system : A cooperative learning approach to the traveling salesman problem. *IEEE Transactions on Evolutionary Computation*, 1(1) :53–66, 1997.

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

[20] J. Dréo, A. Pétrowski, and P. Siarry. *Métaheuristiques pour l'optimisation difficile.* Eyrolles, 2003.

[21] J. D. Farmer, N. Packard, and A. Perelson. The immune system, adaptation and machine learning. *Physica D*, 22 :187–204, 1986.

[22] R. W Floyd. Algorithm 97, shortest path. *Comm. ACM*, 5 :345, 1959.

[23] Thomas A. Féo and Mauricio G. C. Resende. A probabilistic heuristic for a computationally difficult set covering problem. *Operations Research Letters*, 8 :67–71, 1989.

[24] H. M. Foo, H. W. Leong, Y. Lao, H. C. Lau, and D. Singer. A multi-criteria, multi-modal passenger route advisory system. In *Proceedings of 1999 IES-CTR Int'l Symp*, Singapore, 1999.

[25] L. Gambardella, Taillard E. D, and G. Agazzi. Macs-vrptw : A multiple ant colony system for vicule routing problems with time windows. *In D. Corne. M. Dorigo and F. Glover Editors. New ideas in optimization, McGraw-Hill*, pages 63–76, 1999.

[26] L. M. Gambardella and M. Dorigo. Ant-Q : A reinforcement learning approach to the traveling salesman problem. In *Proceedings of ML-95, Twelfth International Conference on Machine Learning*, pages 252–260, Tahoe City, CA, A. Prieditis and S. Russell (Eds.), Morgan Kaufmann, 1995.

[27] F. Glover. Future paths for integer programming and links to artificial intelligence. *Computers and Operations Research*, 13(5) :533–549, 1983.

[28] F. Glover and M. Laguna. Tabu search. *Kluwer Academic Publishers*, 1997.

[29] D. E. Goldberg. Genetic algorithms in search,optimization, and machine learning, 1989.

[30] M. Guntsch. *Ant algorithms in stochastic and multi-criteria environments*. PhD thesis, Institut für Angewandte Informatik und Formale Beschreibungsverfahren (AIFB), 2004.

[31] M. Guntsch and M. Middendorf. A population based approach for aco. In *In S. C. et al.,editor, Applications of Evolutionary Computing -EvoWorkshops*, 2002.

[32] Z. Habbas, A. Aloui, and M. A. Tahraoui. Bi-colonie pour l'optimisation d'un réseau de transport multimodal perturbé. *Workshop International : Logistique et Transport, LT2009. Accepté*, 2009.

[33] J. K. Hao, P. Galinier, and M. Habib. Métaheuristiques pour l'optimisation combinatoire et l'affectation sous contraintes. *Journal of Algorithms*, 13(2) :283–324, 1999.

[34] J. H. Holland. Adaptation in natural and artificial systems. *Ann Arbor, Michigan, The University of Michigan Press*, 1975.

[35] S. Iredi, D. Merkle, and M. Middendorf. Bi-criterion optimization with multi colony ant algorithms. *Lecture Notes in Computer Science*, 1993 :359–372, 2001.

[36] S. Jodogne. Optimisation par une colonie d'agents coopérants, 2001. Université de Liège.

[37] V. Kaufmann. The rationality of perception and modal choice. is quickest best ? *Recherche Transports Sécurité*, 75 :131–143, 2002.

[38] J. Kennedy and R. C. Eberhart. Particle swarm optimization. In *Proceedings of the IEEE International Conference on Neural Networks*, pages 1942–1948, Perth, Australia, IEEE Service Center, Piscataway, NJ, IV, 1995.

[39] S. Kirkpatrick, C. D. Gelatt Jr, and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220 :671–680, 1983.

[40] H. Laichour. *Modélisation multi-agent et aide à la décision : application à la régulation des correspondances dans les réseaux de transport urbain*. PhD thesis, thèse de doctorat, Université de Lille, 2002.

[41] A. Lozano and G. Storchi. Shortest viable hyperpath in multimodal networks. *Transportation Research Part B*, 36 :853–874, 2002.

[42] C. M. Mariano and E. Morales. A Multiple Objective Ant-Q Algorithm for the Design of Water Distribution Irrigation Networks. Technical Report HC-9904, Instituto Mexicano de Tecnologia del Agua, 1999.

[43] S. Morin. Algorithme de fourmis avec apprentissage et comportement spécialisés pour l'ordonnancement de voitures, 2005. Mémoire de maîtrise, in Département d'Informatique et de Mathématique, Université du Québec à Chicoutimi.

[44] A. Orda and R. Rom. Shortest-path and minimum-delay algorithms in networks with timedependent edge length. *ACM*, 37(3) :607–625, 1990.

[45] M. Ould sidi, S. Hammadi, S. Hayat, and P. Borne. Approche floue d'un système d'évaluation des stratégies de régulation d'un réseau de transport perturbé. In *LFA 2004*, Nantes, Novembre 2004.

[46] S. Pallottino and M. G. Scutellà. Shortest path algorithms in transportation models : classical and innovative aspects. *In Equilibrium and Advanced Transportation Modelling, Kluwer, Zurich*, pages 245–281, 1998.

[47] D. Pinto and B. Baran. Solving multiobjective multicast routing problem with a new ant colony optimization approach. In *Proceedings of the 3rd international IFIP/ACM Latin American conference on Networking*, pages 11–19, Cali, Columbia, 2005.

[48] J. R. Schott. *Fault Tolerant Design Using Single and Multicriteria Genetic Algorithm Optimization.* PhD thesis, Université de Metz, 1998.

[49] T. Stützle and H. H. Hoos. MAX-MIN ant system. *Future Generation Computer Systems*, 16(8) :889–914, 2000.

[50] A. Tahraoui, Z. Habbas, and A. Aloui. Using aco for solving a multimodal transport problem. In *Seconde international conference on Metaheuristics and Nature Inspired Computing, META'08*, Hammamet ,Tunisie, 2008.

[51] E. G Talbi. Métaheuristiques pour l'optimisation combinatoire multi-objectif : etat de l'art. Technical report, Université de Lille, 1999.

[52] D.J. M. Van Laarhoven, E. H. Aarts, and J. K. Lenstra. Job shop scheduling by simulated annealing. *Operational Research*, 40(1) :113–125, 1990.

[53] K. Zidi and S. Hammadi. Algorithme génétique avec contrôle des opérateurs pour l'optimisation multicritère d'un déplacement dans un réseau de transport multimodal. In *Workshop avec école intégrée Méthodologies et Heuristiques pour l'Optimisation des Systèmes Industriels*, Hammamet ,Tunisie, 24-26 Avril, 2005.

[54] S. Zidi. *SARR : Système d'aide a la régulation et la reconfiguration des réseaux de transport multimodal.* PhD thesis, Université de Science et Technologies de Lille. Ecole Centrale de Lille, 2007.

[55] A. Ziliaskopoulos and W. Wardell. An intermodal optimum path algorithm for multimodal networks with dynamic arc travel times and switching delays. *Elsevier Science, European Journal Of Operational Research*, 125 :468–502, 2000.

[56] T. Zitzler and L. Thiele. Multiobjective Evolutionary Algorithms : A Comparative Case Study and the Strength Pareto Approach. *IEEE Transactions on Evolutionary Computation*, 3 :257–271, 1999.