Université de Montréal

Gestion de l'information en temps réel pour un répartiteur de véhicules

par

Ying Xu

Département d'informatique et de recherche opérationnelle

Faculté des arts et des sciences

Mémoire présenté à la Faculté des études supérieures

en vue de l'obtention du grade de

Maître ès sciences (M.Sc.)

en informatique option informatique

Mai, 2003

Université de Montréal

Faculté des études supérieures

Ce mémoire intitulé:

Gestion de l'information en temps réel pour un répartiteur de véhicules

présenté par:

Ying Xu

a été évalué par un jury composé des personnes suivantes:

Michaël Florian

(président-rapporteur)

Jean-Yves Potvin

(directeur de recherche)

Ilham Benyahia

(co-directeur)

Michel Gendreau

(membre du jury)

Mémoire accepté le:

18 août 2003

Sommaire

Nous exploitons un modèle général d'architecture distribuée multi-agents pour la gestion de l'information en temps réel, afin de réaliser une application pour la répartition et le routage dynamique de véhicules. Ici, des requêtes de clients sont reçues de façon continue au cours de la journée et doivent être incorporées au sein des routes planifiées des véhicules de façon à minimiser les coûts d'exploitation (i.e., temps total de parcours) et maximiser la satisfaction des clients (i.e., respecter leur fenêtre de temps). Les temps de parcours entre les clients sont dynamiques, et ne sont pas connus de façon certaine au moment de la génération des routes planifiées. Un agent "répartiteur" réalise la fonction d'affectation des nouvelles requêtes au sein des routes tandis que des agents subalternes fournissent des informations en temps réel sur la situation courante de répartition.

Des simulations sont réalisées afin d'évaluer les mérites de différentes stratégies de répartition.

Mots-clés : Répartition et routage de véhicules en temps réel, fenêtres de temps, temps de parcours dynamiques, application distribuée, Java RMI.

Abstract

We specialize a generic distributed multi-agent architecture for real time information management to address a dynamic vehicle routing and dispatching problem. Here, customers requests are received continuously over the day and must be assigned and inserted into the vehicles planned routes at minimum cost. In our case, the cost is related to the total travel time plus a penalty for lateness at customer locations. The innovative feature of our work is that travel times between customers are dynamic and are influenced by different unexpected events on the transportation network. Various dispatching strategies for reacting to these unexpected events are considered.

In our implementation, a dispatcher agent assigns new requests to vehicles, while subordinate agents provide information in real time on the current dispatching environment. Simulations are carried out in order to evaluate the merits of the proposed dispatching strategies.

Key-words : Dynamic vehicle routing and dispatching, time windows, dynamic travel times, distributed computing, multi-agent architecture, Java RMI.

Table des matières

Remerciements **vii**

1 Introduction **1**

1.1 Motivation 1

1.2 Description du problème 2

1.3 Organisation du mémoire 5

2 Revue de littérature **6**

2.1 Différences entre les problèmes statiques et dynamiques 7

2.2 Quelques applications en routage et répartition de véhicules 9

2.2.1 Problèmes de transport sur demande ("dial-a-ride") 9

2.2.2 Services de courrier rapide 10

2.2.3 Services de réparation 10

2.2.4 Services de courrier international (portion locale) 10

2.2.5 Services d'urgence 11

2.3 Degré de dynamisme 11

2.4 Méthodologies générales de résolution 14

2.4.1 Adaptation d'algorithmes statiques 14

2.4.2 Méthodes d'insertion 15

2.4.3 Recherche Tabou 15

2.4.4 Méthodes stochastiques 18

2.4.5 Méthodes basées sur la théorie des files d'attente 18

3 Vehicle Routing and Scheduling with Dynamic Travel Times 19

3.1 Introduction 21

3.2 The static problem 21

3.3 The dynamic problem 23

3.3.1 Dynamic elements 23

3.3.2 Operating mode 24

3.4 Dispatching algorithm 25

3.4.1 Initial solution 25

3.4.2 Solution construction. 26

3.5 Computational results 31

3.6 Conclusion 34

4 Implémentation 35

4.1 Introduction aux agents 35

4.2 Architecture du système 36

4.2.1 Tâches réalisées par l'Agent Superviseur 37

4.2.2 Tâches réalisées par l'Agent Environnement 37

4.2.3 Tâches réalisées par l'Agent Complexe 37

4.3 Les outils pour l'implémentation de notre application 38

4.3.1 Technologie RMI 38

4.3.2 L'interface Java Gui 39

4.4 Implémentation 40

4.4.1 Agent Superviseur 40

4.4.2 Agent Environnement 41

4.4.3 Agent Complexe 42

4.5 Interfaces utilisateur associées aux Agents : 43

4.5.1 Interface Agent Environnement : 44

4.5.2 Interface Agent Complexe : 44

4.5.3 Interface Agent Superviseur : 45

5 Conclusion **49**

Bibliographie **52**

Liste des figures

1.1 Une route de véhicule dans un contexte dynamique 4

2.1 Degré de dynamisme avec fenêtres de temps 13

3.1 Reassignment of customers 29

4.1 Agent Environnement 45

4.2 Choix de scénario 46

4.3 Agent Complexe 47

4.4 Agent Superviseur 48

5.1 Réaffectation en bloc des clients u, v, w selon l'estimé 51

Liste des tableaux

3.1 Problem class R2 32

3.2 Problem class C2 33

3.3 Problem class RC2 33

Remerciements

Ce mémoire a pu être complété grâce à l'aide et aux conseils de mon directeur Jean-Yves Potvin et de ma co-directrice Ilham Benyahia. Je leur témoigne une grande reconnaissance pour leurs encouragements soutenus ainsi que pour le soutien financier. Je voudrais remercier le professeur Potvin pour m'avoir fait bénéficier de son expérience dans le domaine des problèmes de routage dynamique de véhicules, ainsi que pour son support et sa disponibilité. Je tiens aussi à remercier la professeure Benyahia pour son support et son aide.

Je profite de cette occasion pour présenter mes compliments aux professeurs, chercheurs, professionnels et étudiants du Centre de recherche sur les transports pour leur amabilité et leur aide.

Chapitre 1

Introduction

1.1 Motivation

Les problèmes de routage et répartition dynamique de véhicules consistent à déterminer un ensemble de routes de coût minimal qui satisfont certaines contraintes d'opération, comme la capacité des véhicules, les fenêtres de temps pour la cueillette ou la livraison aux clients, etc.

De nos jours, l'économie mondiale s'oriente de plus en plus vers l'ouverture. Cette nouvelle tendance rend la gestion efficace des ressources de plus en plus importante. Notons que chaque petite amélioration dans les techniques de résolution de ces problèmes peut avoir un impact majeur sur tous les clients, parce que les coûts pour les services logistiques constituent une fraction non négligeable du prix des marchandises qu'ils achètent.

Les problèmes de routage peuvent se diviser en deux grandes classes : les problèmes

statiques et les problèmes dynamiques (temps réel, en ligne). Dans les problèmes statiques, l'information est connue avant que les routes ne soient exécutées. Par contre, dans les problèmes dynamiques, l'information est révélée au fur et à mesure que le temps avance. Un tel scénario temps réel devient de plus en plus commun aujourd'hui, et sera certainement encore plus répandu dans le futur.

Les progrès dans les techniques de l'information et de la communication (e.g., positionnement par satellite, téléphonie cellulaire, télé-détéction, systèmes d'information géographique, etc.) permettent d'obtenir une grande quantité d'informations en temps réel plus facilement. De plus, l'augmentation de la vitesse de calcul et le traitement distribué offrent de nouvelles possibilités pour l'élaboration de méthodologies capables de transiger avec la composante "dynamique" de ces problèmes. Ce mémoire s'inscrit dans le cadre de cette nouvelle direction de recherche, puisque nous nous intéressons tout particulièrement aux aspects dynamiques des problèmes de routage.

1.2 Description du problème

Dans ce mémoire, le problème considéré s'inspire des problèmes rencontrés dans les systèmes de courier rapide. Il s'agit essentiellement de problèmes de cueillette, sans contraintes de capacité, mais avec fenêtres de temps aux clients. Une description précise du problème est fournie au Chapitre 3, mais les caractéristiques principales sont les suivantes :

Etant donné une flotte de véhicules et des requêtes qui arrivent de façon continue au cours de la journée, le but est d'intégrer chaque nouvelle requête à coût minimum aux routes courantes. De plus,

1. Les routes des véhicules commencent et se terminent à un dépôt central.
2. Chaque véhicule dessert seulement une route.
3. Chaque requête est desservie une seule fois par un seul véhicule.
4. La capacité est infinie (parce que les items transportés sont petits).
5. Chaque client ou point de service i a sa propre fenêtre de temps $[e_i, l_i]$, où e_i est le temps de début de service au plus tôt et l_i est le temps de début de service au plus tard. Ces fenêtres de temps sont souples, moyennant une pénalité au niveau de l'objectif.
6. L'objectif consiste à minimiser le temps de parcours total des véhicules plus une pénalité liée à la violation des fenêtres de temps (voir Chapitre 3).

Dans un contexte dynamique, une route pour un véhicule se divise à tout moment en trois parties :

1. La portion de route déjà exécutée (qui ne peut plus être optimisée).
2. Le mouvement courant du véhicule vers sa destination courante.
3. La route planifiée, qui est constituée par l'ensemble des requêtes affectées au véhicule mais non encore desservies par ce véhicule (cette partie peut être optimisée).

Lorsqu'une nouvelle requête arrive, elle ne peut évidemment être insérée que dans la route planifiée. Dans ce mémoire, nous explorons différentes façons de réagir et de réajuster les routes planifiées lorsque des événements imprévus surviennent.

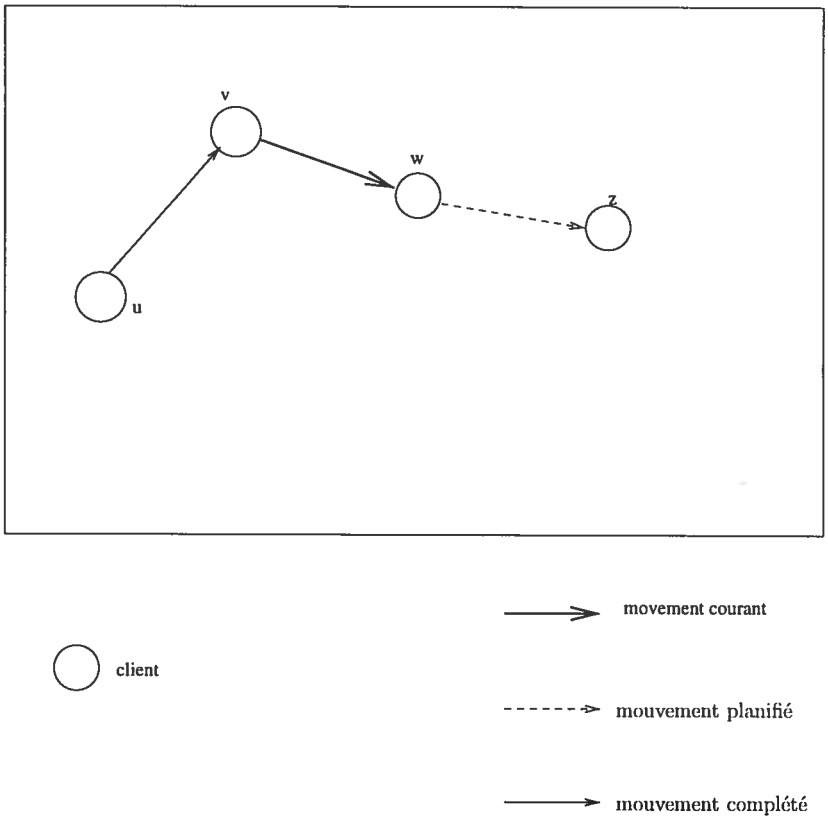

FIG. 1.1 – Une route de véhicule dans un contexte dynamique

1.3 Organisation du mémoire

Hormis ce chapitre, ce mémoire se compose de quatre autres chapitres.

Le chapitre 2 est une revue de la littérature portant sur les problèmes de routage et répartition dynamique de véhicules.

Le chapitre 3 constitue le corps du mémoire. Il s'agit d'un article soumis à *European Journal of Operational Research* qui décrit différentes façons de réagir et de réajuster la solution planifiée lorsque des événements inattendus empêchent le véhicule de se présenter à sa destination courante au moment prévu. Dans le problème qui nous intéresse, on retrouve donc à la fois des arrivées dynamiques de requêtes et des temps de parcours dynamiques. À notre connaissance, c'est la première fois que ces deux types d'événements sont tenus en compte dans un algorithme de répartition.

Le chapitre 4 porte sur l'implantation du modèle décrit au chapitre 3. Le code s'intègre dans un cadre général fondé sur une architecture distribuée multi-agents.

Le chapitre 5 résume les principales contributions du mémoire et propose quelques avenues de recherche qu'il serait pertinent d'explorer. En particulier, une nouvelle version du modèle présenté au chapitre 3 est proposée, où l'on suppose qu'un événement imprévu est communiqué au répartiteur dès qu'il survient.

Chapitre 2

Revue de littérature

Le routage et la répartition dynamique de véhicules représente une vaste classe de problèmes où des informations sont révélées en temps réel au planificateur. Ces problèmes ont été étudiés en recherche opérationnelle avec grand intérêt au cours des dernières années. D'ailleurs, il existe probablement autant de variantes de ces problèmes que d'applications pratiques dans le monde réel. Dans ce chapitre, nous passons en revue les travaux réalisés en routage et répartition dynamique de véhicules. D'abord, nous présentons les différences principales entre les problèmes statiques et dynamiques. Ensuite, nous présentons un certain nombre d'applications pratiques. Dans la troisième partie, nous discutons de la mesure du degré de dynamisme d'un problème. En dernière partie, nous abordons les méthodologies générales adoptées pour traiter de l'aspect dynamique de ces problèmes.

2.1 Différences entre les problèmes statiques et dynamiques

Dans cette section, nous établissons des distinctions entre le problème dynamique de routage de véhicules et le problème statique. Plus précisément, dans le problème dynamique, on retrouve les caractéristiques suivantes :

1. **La dimension temporelle est essentielle.** Dans la version dynamique, de nouvelles informations sur le problème sont révélées, telles l'arrivée de nouvelles requêtes, à mesure que le temps passe et que les routes courantes sont exécutées.
2. **L'information sur le futur est imprécise ou bien inconnue.** Dans le problème dynamique, des événements aléatoires se produisent. Au mieux, on a une connaissance probabiliste de ces événements. Celà s'oppose au problème statique où toutes les informations nécessaires à la résolution sont connues avec certitude.
3. **Les événements à court terme sont plus importants.** Dans le problème dynamique, les événements à court terme sont plus importants que ceux à plus long terme. Par exemple, des requêtes dont la fenêtre de temps est relativement éloignée par rapport au temps courant peuvent être temporairement mises de côté, car des événements subséquents sont susceptibles de modifier la situation et rendre caduque toute affectation qui serait faite immédiatement.
4. **le problème est plus ouvert.** Au contraire du problème statique qui est bien balisé, le problème dynamique est de nature plus ouverte. Par exemple, la route courante planifiée peut ne pas se terminer au dépôt, mais plutôt au dernier client visité dans

la route courante, parce que le chauffeur retourne ensuite directement à son domicile avec le véhicule.

5. **Des mécanismes de mise à jour de l'information sont essentiels.** Les entrées d'un problème de routage dynamique de véhicule sont soumises à des changements continuels pendant la journée d'opération. Il est donc essentiel que des mécanismes de mise à jour de l'information soient intégrés dans la méthode de résolution.

6. **La remise en question de décisions antérieures est essentielle.** L'arrivée de nouveaux événements force le répartiteur à reconsidérer ses décisions antérieures, par exemple en modifiant l'affectation des clients ou le séquencement des clients dans les routes planifées courantes, pour répondre à la nouvelle situation.

7. **Des temps de calcul plus rapides sont nécessaires.** Dans les problèmes dynamiques, des temps de réponse rapides sont importants, car les événements peuvent se produire à intervalles rapprochés (dépendant du degré de dynamisme du problème). On fait donc appel le plus souvent à des heuristiques afin d'ajuster la solution planifiée. De plus, un traitement distribué où plusieurs composantes du système collaborent ensemble, peut apporter une contribution importante à la réduction des temps de calcul.

8. **La fonction objective est souvent différente.** La fonction objective peut, par exemple, introduire des éléments qui ont pour but d'exploiter une certaine connaissance probabiliste des événements futurs. Ceci n'a évidemment aucun sens dans un contexte statique.

9. **Les contraintes de temps sont souvent traitées de façon différente.** Dans un contexte dynamique, on tolère souvent des écarts par rapport à la fenêtre de temps associée au client. Cette contrainte est donc en général traitée de façon plus souple.

10. **Des considérations de type file d'attente peuvent être utiles.** Si le nombre de requêtes dépasse un seuil, le système devient congestionné comme dans un système de file d'attente. Bien que le routage de véhicules et les systèmes de file d'attente soient deux disciplines bien étudiés dans la littérature, peu d'études touchent à leur interface.

2.2 Quelques applications en routage et répartition de véhicules

Les principales applications reliées au routage et à la répartition de véhicules sont les suivantes :

2.2.1 Problèmes de transport sur demande ("dial-a-ride")

Dans ce problème, une flotte de véhicules avec contraintes de capacité assure le transport de clients ayant des besoins particuliers (e.g., personnes âgées ou handicapées, transport scolaire, etc...). Chaque client a un point de cueillette et un point de livraison. Une fenêtre de temps est associée au point de cueillette, au point de livraison ou aux deux. Dans cette application, il est très important que les contraintes de capacité des véhicules soient respectées.

2.2.2 Services de courrier rapide

Les services de courrier rapide opèrent généralement dans une région dont le périmètre est limité (e.g. une zone urbaine). Les caractéristiques de ce problème sont semblables à celles des problèmes de transport sur demande, sauf qu'il n'y a pas de contraintes de capacité (en effet, les colis et lettres transportés sont de petite taille). De plus, on peut oublier certaines considérations associées au transport de personnes, comme la durée du trajet entre le point de cueillette et le point de livraison.

2.2.3 Services de réparation

Ces problèmes sont souvent abordés comme des problèmes de file d'attente avec serveurs mobiles. En effet, le temps de service est typiquement beaucoup plus long que dans les applications précédentes et présente également une variance importante d'un client à un autre. Par ailleurs, les fenêtres de temps sont souvent déterminées par la compagnie offrant le service de réparation plutôt que par les clients. Il faut aussi noter que chaque client est associé à un seul point de service. Plusieurs applications pratiques s'inscrivent dans ce cadre. Notons, en particulier, les compagnies de service public (i.e. électricité, gaz, eau, égout, etc.).

2.2.4 Services de courrier international (portion locale)

Dans ces problèmes, une camionnette doit ramasser le courrier à différents points de cueillette dans une zone locale restreinte pour ensuite le retourner à un dépôt central, où le

courrier sera ultérieurement traité (pour envoi international). Ce problème est semblable aux problèmes rencontrés dans les services de courrier rapide intra-urbains évoqués plus haut, sauf que chaque client est associé ici à un seul point de service.

2.2.5 Services d'urgence

Les services d'urgence correspondent aux patrouilles de police, aux services de pompiers, ambulances et services de taxi. Dans ces applications, une flotte de véhicules doit servir un ensemble de clients dans une zone géographique donnée. Les caractéristiques principales de ces problèmes sont l'urgence des requêtes et le fait que chaque véhicule doit servir un seul client à la fois (i.e., pas de consolidation des requêtes au sein d'un véhicule).

2.3 Degré de dynamisme

Il est important de pouvoir mesurer la performance d'un système de routage dynamique de véhicules. À l'opposé du problème de routage statique, la performance du problème dynamique dépend non seulement du nombre de clients et de leur distribution mais aussi du nombre d'événements dynamiques, du moment où ces événements ont lieu et des temps entre l'arrivée de deux événements.. Donc, une mesure simple pour décrire le dynamisme du système a une valeur importante quand on veut examiner la performance d'un algorithme spécifique dans des conditions variées. Dans cette section, nous discutons de mesures pour décrire le dynamisme d'un système de routage dynamique de véhicules [23].

Le degré de dynamisme d'un problème est fonction de deux éléments. Le premier est

la fréquence des changements dans les entrées du problème, alors que la deuxième est la rapidité avec laquelle il faut prendre les décisions.

Fréquence des changements des informations. Il en existe deux types. Le premier type est la fréquence d'apparition des nouvelles requêtes. Ainsi, plus la fréquence est élevée, plus le degré de dynamisme est élevé. Le second type est le degré de "stochasticité" des attributs des nouvelles requêtes (i.e., date d'arrivée, date de début de service, largeur de la fenêtre de temps, demande, etc...). Ainsi, plus ces attributs varient d'un client à un autre et d'une période à l'autre, plus le degré de dynamisme du problème est élevé.

Rapidité avec laquelle une décision doit être prise. Ceci correspond au délai entre la date d'apparition d'une requête et le temps de début de service. La Figure 2.1 présente deux scénarios pour un problème avec fenêtres de temps. Dans le scénario C, la fenêtre de temps $[e_i, l_i]$ d'un client $i = 1, 2$ est plus large par rapport aux fenêtres dans le scénario D. Mais, plus encore, le délai entre la date d'arrivée de la requête r_i et la date de début de service au plus tard l_i est plus long dans le scénario C que dans le scénario D. Clairement, le scénario C est préférable pour le répartiteur, parce qu'il dispose de plus de marge de manoeuvre pour prendre une décision et qu'il y a plus de place pour insérer une requête [32]. Dans la prochaine section, nous classerons les problèmes présentés dans la section 2.2 en fonction de leur degré de dynamisme, qui est soit faible, moyen ou élevé.

• **Dynamisme faible.** Cette catégorie inclut les problèmes de réparation à domicile et les problèmes de transport sur demande. Dans ces problèmes, une majorité de clients sont connus au moment de la construction de la route. En outre, dans les services de réparation à domicile les fenêtres de temps, sont souvent déterminées par la compagnie plutôt que par les clients. Ainsi l'impact de la composante "temps réel" du problème est faible. Par

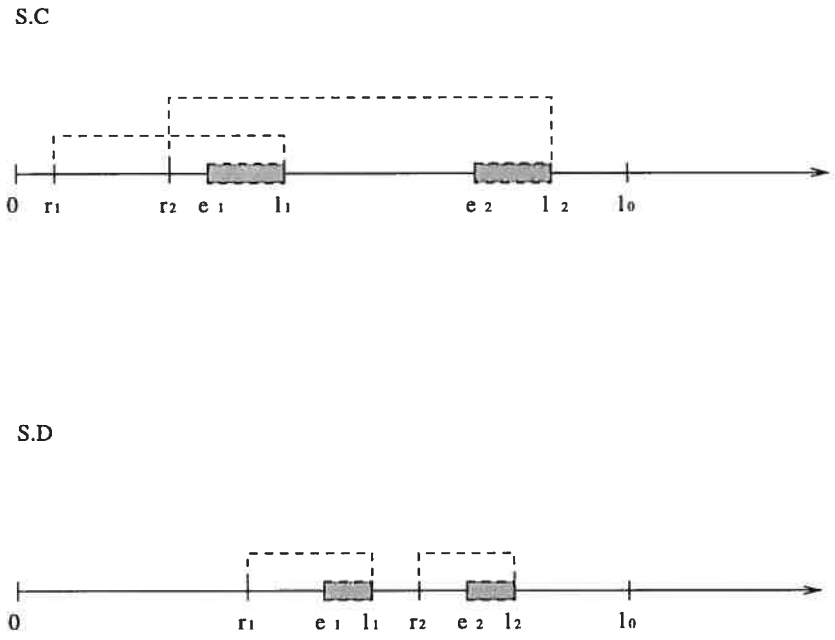

FIG. 2.1 – Degré de dynamisme avec fenêtres de temps

ailleurs, dans les problèmes de transport sur demande, les personnes faisant appel à ces services doivent normalement planifier à l'avance les dates de leurs déplacements. Donc le pourcentage de requêtes dynamiques est assez faible.

• **Dynamisme moyen**. Cette catégorie comprend en particulier les services de courrier international (portion locale). Dans ces problèmes, les requêtes dynamiques constituent une portion considérable du nombre total de requêtes servies dans une journée. Mais leur dégré d'urgence n'est pas aussi important que dans le cas des applications avec dynamisme élevé (voir plus bas).

• **Dynamisme élevé**. Cette catégorie inclut les services de courrier rapide intra-urbains et les services d'urgence. Ces applications sont caractérisées par la vitesse rapide de changement des données et le degré d'urgence des requêtes reçus.

2.4 Méthodologies générales de résolution

Dans cette section, nous abordons les principales approches de résolution proposées dans la littérature récente pour aborder les probèmes dynamiques de routage.

2.4.1 Adaptation d'algorithmes statiques

Cette approche est souvent fondée sur la notation d'horizon fuyant ("rolling horizon"). Il s'agit ici de résoudre, pour chaque nouvelle entrée, un problème statique basé sur les données connues avec certitude à l'intérieur de l'horizon fuyant (par exemple, on optimise les routes sur la base des requêtes reçues jusqu'à date et dont la fenêtre de temps est suffisamment rapprochée dans le temps). Un tel problème statique doit donc être résolu à chaque fois qu'un nouvel événement survient. L'avantage de cette approche est qu'elle repose sur des méthodes qui ont déjà été développées pour résoudre des problèmes statiques. Par exemple, dans [46], un algorithme de programmation dynamique qui traite un problème statique de transport sur demande avec un seul véhicule et sans fenêtre de temps a été adapté au cas dynamique. Cet algorithme est appliqué de façon répétitive à chaque fois qu'une nouvelle requête survient. Toutefois, à cause de la complexité exponentielle de la programmation dynamique, seuls de petits problèmes peuvent être traités. Ceci illustre l'importance de faire appel à des méthodes qui, même si elles ont été développées dans un contexte statique, sont suffisamment rapides.

2.4.2 Méthodes d'insertion

Cette approche appartient en fait à la catégorie précédente, mais vu son importance pratique, nous la distinguons ici. Le mécanisme général consiste à introduire chaque nouvelle requête à l'intérieur des routes courantes en l'insérant à moindre coût entre deux clients consécutifs. Différentes régles peuvent être appliquées afin de choisir le meilleur endroit pour l'insertion. L'avantage de l'approche est sa simplicité de conception et d'implantation, et la facilité de tenir en compte divers types de contraintes. De plus, il s'agit d'une approche très rapide qui se prête bien à un contexte dynamique. Par exemple, on retrouve une telle approche dans [50] pour le transport des handicapés. Dans ce problème, la majorité des requêtes sont connues à l'avance. Ainsi, les routes initiales basées sur les requêtes statiques sont également construites à l'aide de la méthode d'insertion.

2.4.3 Recherche Tabou

La recherche Tabou est une technique de recherche locale itérative. Cette recherche commence par la génération d'une solution initiale. À chaque itération, on construit un voisinage de la solution courante et on cherche la meilleure solution dans ce voisinage. Cette solution devient ensuite la solution courante et la procédure est répétée à nouveau. À l'opposé d'une méthode pure de descente, on permet une dégradation de l'objectif (si la meilleure solution dans le voisinage est pire que la solution courante). Dans [16], la recherche tabou a été appliquée à un problème de type courrier rapide intra-urbain. Les éléments principaux de cet algorithme sont les suivants :

Mémoire adaptative

Une mémoire adaptative est utilisée pour sauvegarder les meilleures solutions rencontrées antérieurement. Cette mémoire est ensuite utilisée afin de créer de nouvelles solutions de départ pour la recherche tabou. Plus précisément, des routes appartenant à des solutions différentes en mémoire sont combinées afin de créer une nouvelle solution (on retrouve une idée similaire dans les algorithmes génétiques). Cette mémoire adaptative permet une *diversification* de la recherche, en lançant la recherche tabou à partir de différents points de départ dans l'espace des solutions.

Structure de voisinage

La structure de voisinage se fonde sur des échanges de type "CROSS exchanges". Deux séquences de clients de longueur arbitraire sont sélectionnés dans deux routes différentes et sont ensuite échangées. Cette structure de voisinage est très puissante et, de plus, ne modifie pas l'orientation des routes, ce qui est très souhaitable en présence de fenêtres de temps.

Décomposition

L'algorithme tabou fait appel à une procédure de décomposition où la solution courante, soit l'ensemble des routes, est partitionnée en sous-ensembles de routes. Une recherche tabou est ensuite appliquée à chaque sous-ensemble. Cette approche permet de réaliser une *intensification* de la recherche, car la recherche tabou s'attaque alors à des sous-problèmes plus petits.

Parallélisme

Grâce aux techniques de calcul parallèle, il est possible d'optimiser davantage entre l'arrivée des événements dynamiques. Dans [16, 23], le parallélisme est réalisé à deux niveaux. Premièrement, des fils ("threads") de recherche sont exécutés en parallèle, à partir de solutions de départ différentes, générées à l'aide de la mémoire adaptative. Deuxièmement, à l'intérieur de chaque fil de recherche, des recherches tabou sont appliquées en parallèle à chacun des sous-problèmes générés par la méthode de décomposition (présentée plus haut).

Etant donné la plate-forme disponible, en l'occurence un réseau de stations de travail, un schéma maître-esclave a été retenu. Le maître gère la mémoire adaptative et crée les solutions de départ pour les processus tabou esclaves.

Dans ce travail, deux types d'événements interrompent la procédure d'optimisation :

1. L'arrivée d'une nouvelle requête.
2. La fin du service d'un véhicule à son client courant.

Quand une nouvelle requête arrive, les processus de recherche tabou sont arrêtés et envoient leur meilleure solution au processus maître pour inclusion (possiblement) dans la mémoire adaptative. Ensuite, la nouvelle requête est insérée dans chacune des solutions en mémoire adaptative. Par la suite, de nouvelles solutions de départ sont générées à l'aide de la mémoire adaptative afin de "repartir" les processus tabou.

Lorsqu'un véhicule a fini son service au client courant, on arrête aussi les processus de recherche tabou. On indique alors au véhicule le prochain client à servir selon la meilleure solution actuellement en mémoire adaptative. Les autres solutions dans la mémoire sont

ensuite mises à jour et de nouvelles solutions de départ sont générées à l'aide de la mémoire adaptative afin de "repartir" les processus tabou.

2.4.4 Méthodes stochastiques

Il est évidemment souhaitable d'intégrer les aspects dynamiques et stochastiques d'un problème dans la modélisation même. La programmation stochastique et les modèles de décision markoviens ouvrent une voie en ce sens. De plus, ces modèles permettent de considérer la connaissance probabiliste que l'on a des événements futurs. Dans [37], par exemple, on trouve un exemple d'un modèle de décision markovien pour un problème de livraison. Malheureusement, ces approches sont encore relativement peu développées et mènent la plupart du temps à des espaces de recherche de trop grande taille pour être exploités efficacement. Ainsi, seules des applications de très petite taille sont rapportées dans la littérature.

2.4.5 Méthodes basées sur la théorie des files d'attente

On peut trouver dans [7, 8] des travaux où l'on exploite la théorie des files d'attente afin d'établir des *politiques* de routage dans des contextes dynamiques. Cette approche est toutefois assez peu courante dans la littérature et se retrouve essentiellement dans les travaux théoriques ayant pour but de décrire le comportement de certaines politiques lorsque l'arrivée des requêtes est soumise à différentes distributions de probabilités.

Chapitre 3

Vehicle Routing and Scheduling with Dynamic Travel Times

La référence de cet article est :

Y. Xu, J.-Y. Potvin, and I. Benyahia, "Vehicle Routing and Scheduling with Dynamic Travel Times", soumis à *European Journal of Operational Research*.

Vehicle Routing and Scheduling with Dynamic Travel Times

Ying Xu*†, Jean-Yves Potvin*† and Ilham Benyahia‡

*Centre de recherche sur les transports, Université de Montréal, C.P. 6128, succursale Centre-ville, Montréal, Québec, Canada H3C 3J7.

†Département d'informatique et de recherche opérationnelle, Université de Montréal, C.P. 6128, succursale Centre-ville, Montréal, Québec, Canada H3C 3J7.

‡Département d'informatique et d'ingénierie, Université du Québec en Outaouais, C.P. 1250, succursale B, Hull, Québec, Canada J8X 3X7.

Abstract The field of dynamic vehicle routing and scheduling is growing at a fast pace nowadays, due to many potential applications in courier services, emergency services, truckload and less-than-truckload trucking, and many others. In this paper, a dynamic vehicle routing and scheduling problem with time windows is described where both real-time customer requests and dynamic travel times are considered. Different reactive dispatching strategies are defined and compared through the setting of a single "tolerance" parameter. The results show that some tolerance to deviations with the current planned solution usually leads to better solutions.

Keywords. Transportation, Routing, Dynamic, Heuristics, Tolerance.

3.1 Introduction

Real-time vehicle routing and scheduling has been the subject of many studies during the last few years, as reviewed in [12, 17, 42]. Different questions and issues arising from a dynamic context are also discussed in [45, 46]. Although some recent work handles either dynamic customer requests or dynamic travel times, these two types of events have seldom (if never) been addressed concurrently. In this paper, a simplified vehicle dispatching environment is devised to analyze different reactive strategies in a context where both dynamic customers and dynamic travel times occur.

This paper will demonstrate that "extreme" reactive strategies should be avoided when facing dynamic travel times. Extreme refers here to either a reaction to any deviation with the current planned solution or no reaction at all. Some tolerance to deviations leads to better overall results, as shown in Section 3.5. This tolerance, however, depends on the magnitude of the events.

3.2 The static problem

The problem is motivated from the management of local routes in international courier services. This is a many-to-one type of problems where express mail is collected at different customer locations and brought back to a central depot, for further shipping. The static version of the problem can be characterized as an uncapacitated vehicle routing problem with time windows (VRPTW), which is formally stated in the following.

Graph.

We have a complete graph $G = (V, E)$ where $V = \{0, 1, 2, .., n\}$ is the vertex set and E is the edge set. Vertices $i = 1, ..., n$ correspond to customers, whereas vertex 0 is the depot. A non negative travel time t_{ij} is associated with each edge $(i, j) \in E$.

Customers.

Each customer i is characterized by a pick-up location, a service time s_i, a time window $[e_i, l_i]$ and a vehicle planned arrival time t_i. If $t_i < e_i$, the vehicle has to wait up to e_i before servicing the customer. If $t_i > l_i$, a penalty is incurred in the objective.

Depot.

The depot is characterized by a location, a time window $[e_0, l_0]$ and a vehicle return time t_0^k for each vehicle $k \in K$, where K is the set of vehicles. The service time at the depot is assumed to be $s_0 = 0$.

Vehicles.

We have a fixed number of identical uncapacitated vehicles, where each vehicle, if used, travels along a single route that starts and ends at the depot.

Objective.

The objective is to minimize, over all vehicles, a weighted summation of (1) travel time, (2) sum of lateness at customer locations and (3) lateness at the depot. Given a solution $S = \bigcup_{k \in K} S^k$, where $S^k = \{i_0^k, i_1^k, ..., i_{m_k}^k\}$ is the sequence of customer locations visited by vehicle k, with $i_0^k = i_{m_k}^k = 0$, the objective function can be expressed as follows :

$$
\begin{aligned}
f(S) &= \sum_{k \in K} f\left(S^k\right) \\
&= \sum_{k \in K} \left(\alpha_1 \sum_{p=1}^{m_k} t_{i^k_{p-1}, i^k_p} + \alpha_2 \sum_{p=1}^{m_k - 1} \left(t_{i^k_p} - l_{i^k_p} \right)^+ + \alpha_3 \left(t^k_0 - l_0 \right)^+ \right)
\end{aligned}
$$

where α_1, α_2 and α_3 are weighting parameters and $(x-y)^+ = max\{0, x-y\}$.

3.3 The dynamic problem

A dynamic version of the problem presented in section 3.2 is now considered. The dynamic elements include both new request arrivals and dynamic travel times. These will be described below, followed by a description of the system's operating mode.

3.3.1 Dynamic elements

Customer requests

New requests continuously occur during the day and must be inserted into the current planned routes to minimize the additional cost to the solution.

Travel times

Three different elements are considered to set the travel time between two customer locations :

- Long term forecasts. These forecasts represent long-term trends that are known well in advance and are used to evaluate the planned routes. These forecasts are time-dependent as they vary depending on the time period (e.g. morning, lunch time, afternoon). The time-dependent model proposed in [26] is used to calculate the route schedule. In this model, the travel time calculation is updated each time a boundary between two consecutive time periods is crossed to satisfy the FIFO property (i.e., a vehicle leaving earlier must arrive earlier at destination).
- Short-term forecasts. When the vehicle is ready to depart from its current customer location, the travel time to its next destination is modified by a random uniform amount (positive or negative). This modification, which impacts the scheduling of the planned route, represents short-term forecasts or nowcasts based on any new information available at this time.
- Dynamic perturbation. When the vehicle is ready to depart from its current customer location, the travel time to its next destination is further modified by adding a value generated with a normal probability law of mean 0 (and different standard deviations, see Section 3.5). This perturbation represents any unforeseen events that may occur along the current travel leg. This is known to the dispatching system only when the vehicle arrives at its planned destination.

3.3.2 Operating mode

A number of assumptions about the system's operating mode, which impact the algorithmic developments presented in the next section, are introduced here.

- A number of requests that have been received at the end of the day (but too late

to be serviced the same day) are scheduled during the night for the next day. These requests are said to be "static" as they are used to construct in advance a set of initial planned routes.

- A central dispatching office receives the customer calls and manages the planned routes.
- Communication between the drivers and the dispatch office takes place at customer locations only.
- Drivers are informed of their next destination when they depart from a customer location (i.e., they have no knowledge of their current planned route).

3.4 Dispatching algorithm

An insertion heuristic is used to dispatch requests to vehicle routes. In the following, we first describe how the initial routes are generated using the set of static requests. Then, the discrete-event simulation scheme used to handle dynamic events is described.

3.4.1 Initial solution

The algorithm for generating initial routes considers the static customer requests one by one, in random order, and finds the insertion place with minimum additional cost.

Let S be the current solution and V_s be the set of static customer requests. Then, the insertion algorithm is :

1. $S \leftarrow \emptyset$;

2. While $V_s \neq \emptyset$ do :

   2.1 $s \leftarrow$ random selection in V_s ;

   2.2 for each insertion place in S, find the one that minimizes $\Delta f = f(S+s) - f(S)$ and insert s at this place ;

   2.3 $V_s \leftarrow V_s - \{s\}$.

As the number of vehicles is fixed, there may be one or more vehicles with empty routes. Accordingly, the insertion of customer s in some empty route is also considered in step 2.2. The notation $f(S+s)$ simply represents the solution cost after the insertion of customer s.

3.4.2 Solution construction.

Once the initial static routes are generated, the simulation of the operations day can start. During the simulation, different types of events trigger different types of responses. These events are described below, using the following notation for each vehicle $k \in K$.

- i^k the current destination.
- j^k the next customer location to be serviced (which may be different from i^k, see below).
- $(j+1)^k$ the successor of j^k
- TTL^k the tolerance time limit. If TTL^k is exceeded, j^k is removed from the route of vehicle k and reassigned to some other route (see below).

Occurrence of a new request

The new customer request s is inserted at minimum additional cost into one of the planned routes (including empty routes), using the approach previously described in subsection 3.4.1. If we assume that the chosen insertion place is in the route of vehicle k, then the following steps are performed :

1. $S^k \leftarrow S^k + s$;
2. Update the planned arrival time of s and all following locations in S^k.

TTL^k is reached for some $k \in K$.

The tolerance time limit on some vehicle route has been reached. Consequently, customer j^k is removed from the route of vehicle k and inserted at minimum additional cost into the route of another vehicle k'. Note that vehicle k must still reach its current destination i^k, without servicing it, before going to the next location $(j+1)^k$. More precisely, the following steps are performed :

1. Update route of vehicle k.
   - 1.1 $S^k \leftarrow S^k - \{j^k\}$;
   - 1.2 $j^k \leftarrow (j+1)^k$; (j^k is now different from i^k)
   - 1.3 $TTL^k \leftarrow \left(t_{j^k} + t_f\right) - t_{i^k,j^k}$.
2. Update route of vehicle k'.
   - 2.1 $S^{k'} \leftarrow S^{k'} + j^k$;
   - 2.2 update the planned arrival time of j^k and all following locations in $S^{k'}$.

Parameter t_f corresponds to the maximum delay, with regard to the vehicle planned arrival, which can be tolerated before taking a reassignment action. Figure 3.1 provides an illustrative example. Each customer location in this figure is labeled with the corresponding planned arrival time. We assume here that every customer is currently serviced within its time window (so, there is no waiting time) and that the travel time between any two customer locations is 5 time units. The vehicle is currently moving towards customer u and is expected to reach it at time 10. The following customers v, w, z will be visited at time 15, 20 and 25, respectively. Assuming that t_f is set to 1, customer u will be moved to another route if the vehicle does not arrive by the time $10 + 1 = 11$. Customer v will also be moved to another route at the same time. If the vehicle has still not reached location u at time $(20 + 1) - 5 = 16$, then customer w will also be moved, as it is not possible for the vehicle to reach w before time $20 + 1 = 21$. The situation illustrated in the figure would thus occur at any time t, $16 < t \leq 21$ (with $u = i^k$ and $z = j^k$). After time 21, customer z would also be moved to another route. Parameter t_f is important as it leads to different ways of handling dynamic travel times. When $t_f = 0$, the reaction is immediate : an action is taken as soon as the vehicle does not arrive at the planned time. At the other end of the spectrum, $t_f = \infty$ means that no action is taken (i.e., we simply wait for the vehicle; no customers are reassigned to other routes).

Note that the reassignment (from vehicle k to vehicle k') of the customer associated with location i^k can be canceled, if vehicle k actually arrives at i^k before vehicle k'. If, at this time, vehicle k' has not yet reached the predecessor of i^k in its current planned route, i^k is simply removed from it. Otherwise, vehicle k' reaches i^k without servicing it. In Figure 1, customer $i^k = u$ may thus still be serviced by the same vehicle, if the latter arrives at u before the vehicle to which u has been reassigned.

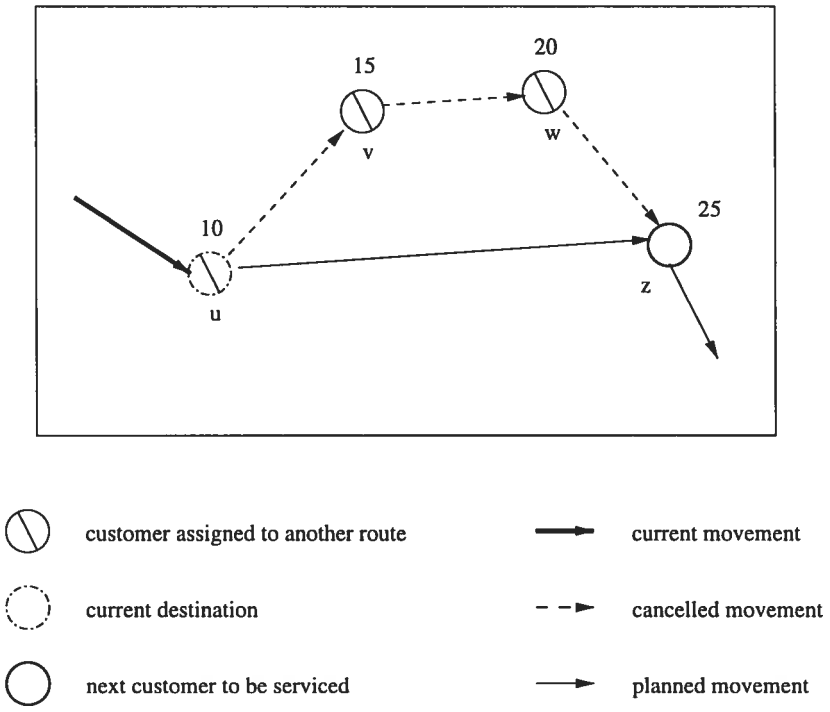

FIG. 3.1 – Reassignment of customers

Vehicle arrival at current destination

When the actual arrival time of the vehicle is known, the route schedule is updated. The response to this event is described below in the "standard situation" with no cancellation of a previous customer reassignment.

1. if $\left(i^k = j^k\right)$ then

   1.1 $S^k \leftarrow S^k - \{j^k\}$;

   1.2 $j^k \leftarrow (j+1)^k$;

2. if the actual arrival time at i^k is different from t_{i^k} then update the planned arrival times in S^k;

3. $TTL^k \leftarrow \infty$.

Vehicle departure

Just before departure from the current location, the short-term forecast is introduced to calculate the travel time to the next customer location and the route schedule is updated accordingly. The TTL of the vehicle route is also calculated. More precisely :

1. $i^k \leftarrow j^k$;

2. introduce the short term forecast into the travel time and update the planned arrival times in S^k;

3. $TTL^k \leftarrow t_{j^k} + t_f$.

In the following section, we analyze the impact of various tolerance levels t_f on solution quality.

3.5 Computational results

For testing purposes, Solomon's VRPTW benchmark problems were used. These 100-customer Euclidean problems are divided into six different classes, depending on the spatial distribution of customers and length of the scheduling horizon. In problems of type C (clustered), the customers are grouped into well-defined geographic clusters; in problems of type R (Random), they are randomly distributed; and in problems of type RC, both clustered and randomly generated customer locations are found. Here, the focus is on the three problem classes C2, R2 and RC2, which are particularly interesting for this study because the time horizon is larger and more customers are scheduled on each route. In all these problems, the travel times correspond to Euclidean distances.

Half of the customers were used to generate the initial set of routes. The occurrence of each remaining (dynamic) customer i was generated at time $e_i * r$, where r is a random number uniformly distributed between 0 and 1. Three different time periods of equal length were defined (morning, lunch time, afternoon). Time-dependency was obtained by multiplying the travel times by coefficients 1.25, 0.5 and 1.25, respectively. Thus, the vehicles travel faster during lunch time. The short-term bias was obtained by perturbing the coefficients with a random value uniformly chosen in the interval [-0.1,+0.1]. The dynamic perturbations were generated according to a normal law of mean 0, with small ($\sigma = 1$, 4) or large ($\sigma = 32$, 64) standard deviations. In this study, only delays to the current schedule were considered. Thus, a negative value was simply reset to 0 (no perturbation).

The number of vehicles for each problem instance was set to the number of routes associated with the best known (static) solution. Different settings for the tolerance level t_f were considered, ranging from $t_f = 0$ to $t_f = \infty$. In the first case, the reassignment of the

t_f	$\sigma = 1$	$\sigma = 4$	$\sigma = 32$	$\sigma = 64$
0	2.064	2.119	4.006	4.799
1	1.278	1.666	—	—
2	1.040	1.699	3.688	4.743
4	1.012	1.549	3.602	4.893
8	1.016	1.254	**2.884**	4.288
16	1.024	1.232	3.410	**3.808**
32	1.003	1.168	3.488	4.208
64	**1.000**	**1.158**	4.555	5.712
96	—	—	4.755	6.554
128	—	—	4.631	6.810
∞	1.009	1.198	4.683	6.973

TAB. 3.1 – Problem class R2

next customer to be serviced is performed as soon as the vehicle does not show up at the planned arrival time. In the second case, nothing is done, as we always wait for the vehicle. Intermediate values correspond to different levels of tolerance (i.e., we are willing to wait for the vehicle, but only to a certain extent).

In Tables 3.1 to 3.3, the solution values correspond to $f(S)$ with $\alpha_1 = \alpha_2 = \alpha_3 = 1$ (see Section 3.2). Each entry in these three tables is an average taken over 11, 8 and 8 problem instances, respectively. Each entry in a given table is also normalized using the best average obtained over all σ and t_f values. Dotted lines correspond to parameter values that were not tested (due to their limited interest).

These results show that problems associated with larger σ values are more difficult to solve (as the magnitude of the dynamic perturbations increases). For problems with $\sigma = 1, 4$, the best results are obtained with relatively large values of t_f (i.e. $t_f \geq 4\sigma$). Thus, it is better to wait for the vehicle's arrival, given that events with more than four standard deviations are unlikely to occur. It also means that the small variations observed between

t_f	$\sigma = 1$	$\sigma = 4$	$\sigma = 32$	$\sigma = 64$
0	3.153	3.038	4.165	5.128
1	1.501	2.781	—	—
2	1.088	2.431	3.808	4.454
4	1.022	1.788	3.758	4.375
8	1.001	1.240	3.652	4.263
16	1.018	1.164	2.369	4.633
32	1.018	**1.161**	2.362	2.750
64	**1.000**	1.195	**1.740**	2.428
96	—	—	1.944	**2.132**
128	—	—	1.891	2.689
∞	1.012	1.184	2.300	3.173

TAB. 3.2 – Problem class C2

t_f	$\sigma = 1$	$\sigma = 4$	$\sigma = 32$	$\sigma = 64$
0	1.716	1.832	2.915	3.460
1	1.237	1.660	—	—
2	1.025	1.616	2.915	2.990
4	1.026	1.219	2.593	2.665
8	1.008	**1.203**	2.608	2.601
16	**1.000**	1.253	2.615	**2.451**
32	1.006	1.266	**2.139**	2.854
64	1.013	1.246	2.962	3.309
96	—	—	3.295	3.832
128	—	—	3.346	3.800
∞	1.005	1.276	3.545	4.013

TAB. 3.3 – Problem class RC2

t_f=4 and $t_f = \infty$ with $\sigma = 1$, and between t_f=16 and $t_f = \infty$ with $\sigma = 4$, are mostly due to randomness. The picture is quite different with σ=32, 64. Although there is some variation with regard to the best t_f value, the latter is always less than or equal to 2σ. The reassignment strategy is thus applied in a non negligible number of cases and with substantial benefits, if one considers the degradation observed when no such reassignment takes place (c.f., $t_f = \infty$).

3.6 Conclusion

The insertion of new customer requests into the current planned routes, as it is done here, closely follows the problem-solving behavior of human dispatchers. The results of this study indicate that some tolerance to unforeseen delays is indicated for better solutions to emerge. Clearly, more sophisticated optimization techniques based on local search could be applied to generate better solutions. The use of these powerful tools might lead to smaller tolerance factors, as reoptimization would alleviate the undesirable effects associated with modifications to the current solution. Of course, the computation times of these sophisticated methods could not be considered negligible anymore, and would lead to considerations about CPU time/solution quality trade-offs. More sophisticated ways of setting the tolerance factor t_f would also be indicated. Rather than using fixed values, customer and solution-dependent tolerance factors should be devised. For example, different classes of customers may lead to different tolerances. Similarly, the tolerance on routes with more slack time should probably be larger than on "tight" routes.

Chapitre 4

Implémentation

L'algorithme présenté au chapitre 3 a été intégré dans un cadre général conçu pour le développement d'applications temps réel distribuées et dynamiques [4]. Ce cadre repose sur une architecture multi-agents qui est décrite dans la suite.

4.1 Introduction aux agents

Il n'y a pas une définition largement acceptée de la notion d'agent. Généralement, le terme "Agent" s'applique à un programme qui exécute des tâches de façon autonome et communique avec d'autre agents. Plus précisément, on retrouve habituellement les caractéristiques suivantes :

- **Autonomie :** les agents opèrent sans l'intervention directe d'humains et peuvent contrôler leurs activités et états.

- **Reactivité** : les agents sont sensibles à leur environnement et répondent aux changements de façon opportune.
- **Complexité** : l'environnement est souvent mal connu et les événements qu'il émet demeurent imprévus dans le temps et parfois par leur nature même.
- **Pro-activité** : un agent ne fait pas que simplement réagir à son environnement. Il est capable de démontrer un comportement plus complexe, en prenant des initiatives visant un but particulier.
- **Capacité sociale** : les agents communiquent entre eux (et possiblement avec un humain).

4.2 Architecture du système

Le cadre que nous réutilisons et spécialisons pour notre application en répartition de véhicules est caractérisé par une architecture distribuée multi-agents et se compose de trois classes d'agents appelés Agent Environnement, Agent Complexe et Agent Superviseur. L'Agent Environnement modélise tout objet qui émet des événements au répartiteur, comme les clients qui envoient des requêtes au répartiteur. L'Agent Complexe joue le rôle du répartiteur. Il est responsable de réagir adéquatement aux événements et implémente les algorithmes présentés au Chapitre 3. L'Agent Superviseur voit à la bonne marche du système et essaie de répartir les tâches adéquatement lorsque plusieurs Agents Complexes sont impliqués (ce qui n'est pas le cas ici ; dans notre contexte, nous supposons qu'il n'y a qu'un seul répartiteur).

4.2.1 Tâches réalisées par l'Agent Superviseur

L'Agent Superviseur réalise les tâches suivants :

- ◇ Il maintient un registre d'objets reflétant l'état des Agents Complexes.
- ◇ Il reçoit les requêtes d'enregistrement des Agents Complexes.
- ◇ Il désigne un Agent Complexe et lui transmet une requête de délégation lorsque celle-ci ne peut être traitée avec les contraintes temporelles de l'Agent Complexe qui l'a reçue au préalable.
- ◇ Il sauvegarde l'identification de l'Agent Environnement.

4.2.2 Tâches réalisées par l'Agent Environnement

L'Agent Environnement réalise les tâches suivantes :

- ◇ Génère les événements destinés aux Agents Complexes. Les événements sont générés selon des scénarios qui modélisent un environnement particulier. Par exemple, différentes lois de probabilité pour l'arrivée des requêtes peuvent être considérées.

4.2.3 Tâches réalisées par l'Agent Complexe

Cet agent recherche le type d'actions (méthodes) à déclencher à partir des caractéristiques de l'événement émis par l'environnement. Dans notre cas, l'Agent Complexe représente un répartiteur avec un comportement défini par les algorithmes présentés dans les chapitres

précédents. Ainsi, dépendant de l'événement (nouvelle requête, retard, ...), il déclenche un traitement qui modifie la solution courante de façon appropriée.

4.3 Les outils pour l'implémentation de notre application

4.3.1 Technologie RMI

Dans notre système, les communications entre les agents sont réalisées avec Java RMI (Remote Method Invocation). Il s'agit d'un mécanisme qui permet d'invoquer de façon transparente un objet qui existe dans un autre espace d'adresses (correspondant, par exemple, à une autre machine). Il diffère ainsi des autres systèmes lors d'une exécution à distance, puisqu'il permet de transmettre n'importe quel type d'objet utilisé par Java, et n'est pas restreint aux types simples de données. Java/RMI repose sur un protocole, appelé "Java Remote Method Protocol (JRMP)", et dépend beaucoup du concept d''Objet-Serialization", qui permet de transmettre les objets sous forme de flot de données.

Java RMI utilise le mécanisme client-serveur comme modèle de communication entre les objets avec la possibilité pour le même objet de se comporter comme serveur pour les uns et clients pour d'autres objets. A chaque objet serveur Java/RMI (Java Virtual Machine - JVM) est associée une interface qui peut être utilisée pour accéder au serveur à partir d'une autre machine. L'interface déclare un ensemble de méthodes qui sont des indications des services fournis par l'objet serveur. Pour qu'un client puisse localiser un objet serveur la première fois, RMI utilise un mécanisme de dénomination, appelé RMIRegistry. Un client

Java/RMI acquiert une référence à un objet serveur Java/RMI en consultant le registre, ce qui lui permet ensuite d'invoquer des méthodes sur l'objet serveur comme s'il était dans son espace d'addresses. Comme Java/RMI est construit sur Java, il peut être employé sur des plates-formes qui opèrent avec des systèmes d'exploitations très différents, comme UNIX et Windows.

4.3.2 L'interface Java Gui

Dans notre système, nous utilisons la technologie Java GUI (Graphical User Interface) afin de rendre l'interface la plus facile d'utilisation possible. Une interface graphique GUI est une interface de dialogue entre un utilisateur et une application. Elle présente en général divers composants sur l'écran (boutons, listes, zones de texte, etc...), de façon compréhensible et conviviale pour un utilisateur. Une telle interface doit être dynamique et réagir aux actions de l'utilisateur (clic sur un bouton, choix d'un élément dans une liste, saisie de texte, déplacement du curseur, sélection d'une commande dans un menu, etc.).

Une interface graphique se compose d'une fenêtre principale (de type *Windows* pour une application, ou de type *panel* pour une *Applet*), dans laquelle on peut placer d'autres composants. Il existe des méthodes pour ajouter, retirer, redimensionner et positionner des composants. Le paquet java.swing contient l'ensemble des classes utilisées pour construire et gérer l'interface dans une application. Pour utiliser l'ensemble de ces classes, il faut mettre en tête du fichier source :

```
import javax.swing.*
```

On peut également dessiner directement sur un composant graphique, et y poser des

chaînes de caractères, des dessins et des images. De plus, la méthode getGraphics() permet de récupérer le contexte graphique de tout composant (couleurs, taille, dessin,...). Ainsi, le programmeur peut à la fois recueillir des informations sur le composant et modifier son aspect.

4.4 Implémentation

4.4.1 Agent Superviseur

L'Agent Superviseur est représenté par une instance de la classe SupAgent. Le registre des Agents Complexes est un objet de la classe Registre qui offre une structure de sauvegarde (Vector, Hashtable...) et des méthodes pour :

- Ajouter un élément (ajouter(EtatAgent))
- Modifier un élément (modifier(EtatAgent))
- Chercher un élément (trouver())

La classe SupAgent possède deux objets membres qui permettent de réaliser les communications. Il s'agit de deux instances, respectivement de la classe imbriquée (Inner class) DialogServer et de la classe imbriquée DialogClient. Ceci nous permet de distinguer deux volets de communication :

Côté Serveur :

La classe DialogServer implémente l'interface RMI de l'Agent Superviseur(SupInterface). Nous présentons ci-dessous les méthodes principales dans cette classe :

- DeclarEnv() : Pour déclarer l'Agent Environnement.
- Inscrire(EtatAgent) : Pour déclarer un Agent Complexe auprès de l'Agent Superviseur. L'Agent Complexe envoie un objet EtatAgent en paramètre.
- Informer(EtatAgent) : Utilisé par un Agent Complexe pour informer d'un changement d'état.
- Show() : Utilisé pour récupérer le contenu du registre des Agents Complexes.

Côté Client :

La classe DialogClient fait appel à la méthode setDelegEvt (Evenement) de l'interface RMI de la classe Agent Complexe.

4.4.2 Agent Environnement

Il s'agit d'une instance de la classe EnvAgent qui possède les membres suivants :

◇ Un objet de la classe DialogServer :

Cette classe implémente l'interface RMI de l'Agent Environnement (EnvInterface). Elle définit donc la méthode suivante :

- GetMyEvts(IdAgent) : elle permet à un Agent Complexe de récupérer les événements qui lui sont destinés en communiquant avec l'Agent Environnement.

◇ Un objet de la classe DialogClient :

On retrouve deux méthodes :

- Inscrire (IdEnv) : Pour déclarer l'identité de l'Agent Environnement auprès de l'Agent Superviseur.
- Liste() : pour récupérer la liste de tous les Agents Complexes déclarés auprès de l'Agent Superviseur

◇ Un objet de la classe SortieMessage

Cette classe, qui permet d'acheminer des messages à l'interface graphique de l'Agent Environnement, a été conue afin de séparer l'interface de la partie traitement au sein de cet agent.

Le simulateur qui génère les événements (pour les résultats numériques décrits au Chapitre 3) s'intègre dans cet agent et envoie les événements à l'Agent Complexe.

4.4.3 Agent Complexe

Un ensemble d'objets instanciant des classe imbriquées :

◇ Classe DialogServer :

Implémente l'interface RMI de l'Agent Complexe. Elle propose à ses clients les principales méthodes suivantes :

- SetDelegEvt(Evenement) : Utilisé par l'Agent Superviseur pour déléguer un événement à l'Agent Complexe.
- SetEvt(Evenement) : Utilisé par l'Agent Environnement pour envoyer un événement à l'Agent Complexe.

- Inserer_dans_la_route(...) : Traite une nouvelle requête provenant de l'Agent Environnement.
- Inserer_dans_la_route_a(...) : Traite la situation où un véhicule ne peut arriver à sa destination courante au temps de tolérance limite.
- Si_inserer_la_route(...) : Traite la situation où un véhicule ne peut arriver à un client dans la route planifiée à son temps de tolérance limite.

  Les algorithmes qui réalisent le traitement des événements transférés par l'Agent Environnement sont intégrés dans cette classe. Une déclaration de ces méthodes est faite dans l'interface RMI de l'Agent Complexe.

◇ Classe DialogClient :

Cette classe définit des méthodes qui permettent d'adresser des requêtes aux interfaces des autres agents (Agent Superviseur, Agent Environnement) :

- Inscrire(EtatAgent) : Pour se déclarer auprès de l'Agent Superviseur
- Informer(EtatAgent) : Pour informer l'Agent Superviseur d'un changement d'état.
- Deleguer(EtatAgent, Evenement) : Pour adresser une requête de délégation d'un événement à l'Agent Superviseur.

4.5 Interfaces utilisateur associées aux Agents :

À chaque Agent est associé une interface graphique pour pouvoir l'initialiser, changer ses paramètres de configuration ou pour suivre son évolution.

4.5.1 Interface Agent Environnement :

Cette interface offre des zones pour la localisation de l'Agent Superviseur et pour l'identification de l'Agent Environnement, tel qu'illustré à la Figure 4.1. En cliquant sur le bouton "Scenario" on obtient ensuite l'interface illustrée à la Figure 4.2. En choisissant un des scénarios dans la liste déroulante obtenue, on peut générer des événements selon l'environnement particulier que nous modélisons.

Une fois le choix effectué, on doit le valider pour le rendre effectif en cliquant sur le bouton "Valider".

4.5.2 Interface Agent Complexe :

Cette interface offre des zones pour le choix du nom de l'Agent Superviseur et de l'Agent Complexe, tel qu'illustré à la Figure 4.3.

Le comportement de l'Agent Complexe intègre deux bases de règles. La première permet de raffiner les événements (e.g., associer une durée à un événement). La seconde permet de faire associer aux événements la meilleure strategie (e.g., si l'événement est une nouvelle requête alors déclencher l'algoritheme d'insertion).

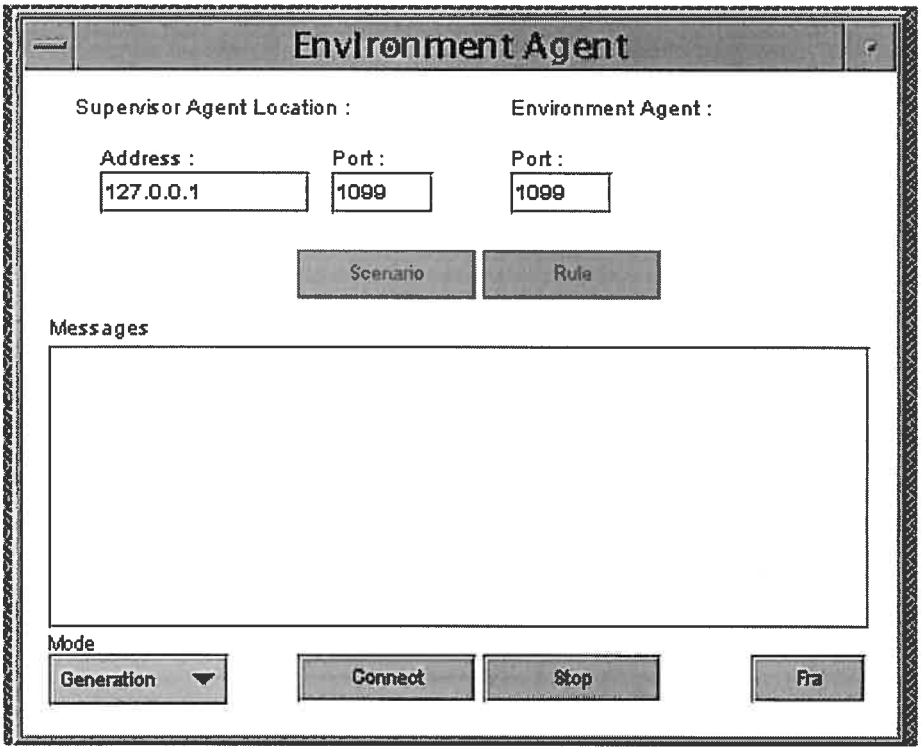

FIG. 4.1 – Agent Environnement

4.5.3 Interface Agent Superviseur :

L'interface de l'Agent Superviseur, illustrée à la Figure 4.4, offre une vision de l'évolution de l'état de saturation des Agents Complexes (non utilisé ici).

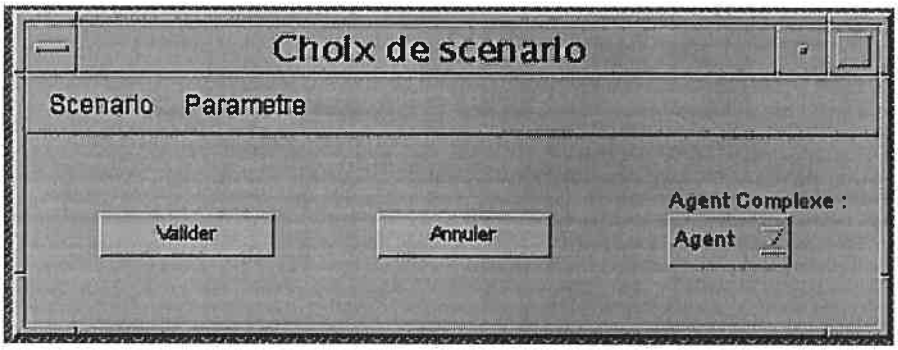

FIG. 4.2 – Choix de scénario

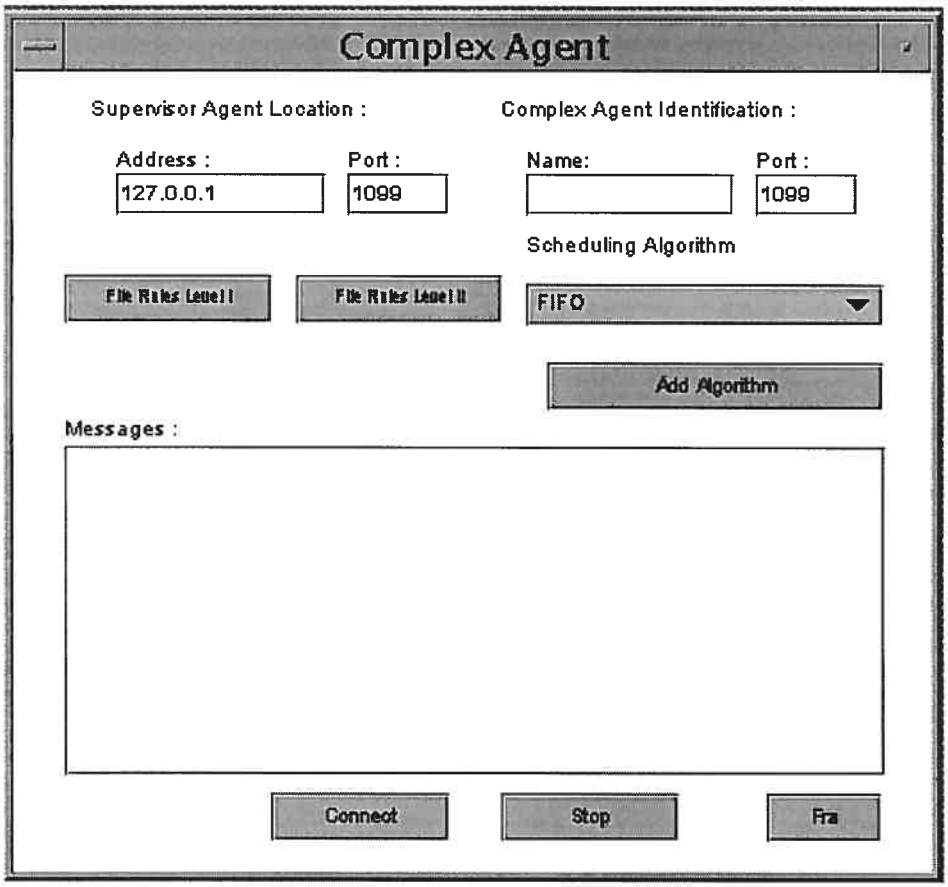

FIG. 4.3 – Agent Complexe

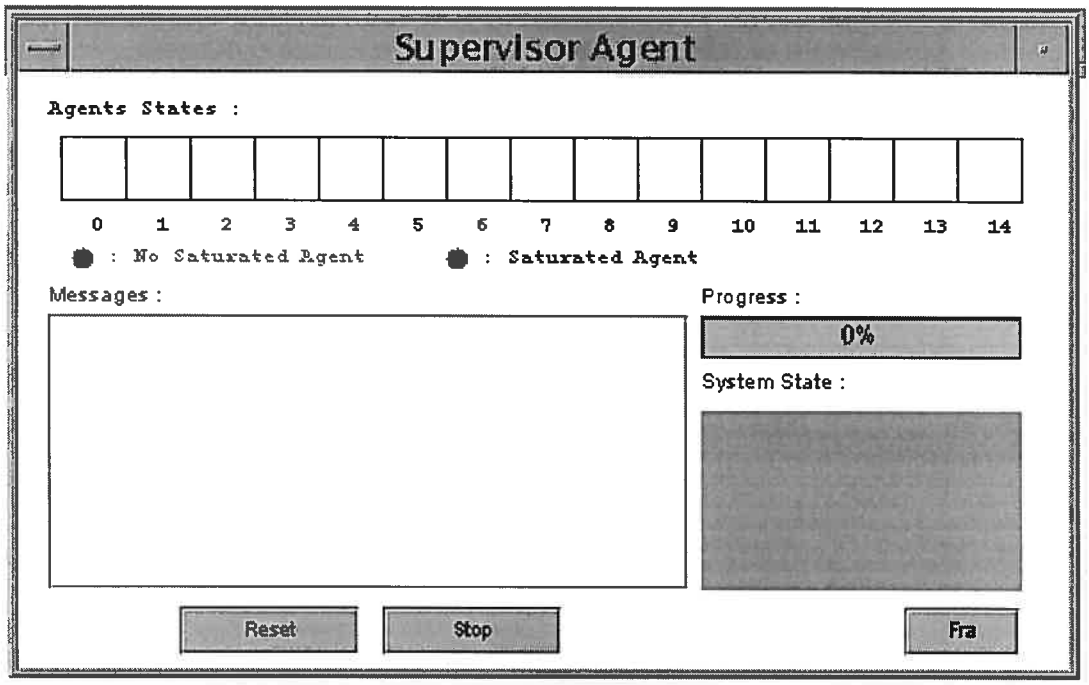

FIG. 4.4 – Agent Superviseur

Chapitre 5

Conclusion

Dans ce mémoire, nous avons étudié différentes façons de réagir lorsque des événements imprévus, qui affectent les temps de parcours des véhicules, surviennent (dans le contexte d'un service de courrier rapide). C'est une des premières études, sinon la première, où l'on considère à la fois des requêtes dynamiques et des temps de parcours dynamiques. Les résultats démontrent que les réactions "extrêmes" ne sont pas profitables (i.e., ne pas réagir du tout ou réagir dès qu'un retard, si minime soit-il, est observé).

Les développements futurs sont de plusieurs ordres et sont énumérés dans la suite.

Distributions de probabilité variées pour les événements : On pourrait évidemment analyser l'effet de distributions de probabilité variées sur les résultats. Différentes lois de distribution pourraient être envisagées autant pour l'arrivée des requêtes que pour la génération des événements imprévus affectant les temps de parcours.

Tolérances variées : Dans ce travail, la tolérance est fixe. On pourrait toutefois utiliser des

tolérances différentes, en considérant la route planifiée courante ou le client. Par exemple, si la route contient beaucoup de temps d'attente, on pourrait se permettre d'être plus patient et considérer une tolérance plus élevée (et inversement). Un raisonnement similaire s'applique en fonction de l'importance du client.

Extensions au code : Le présent code peut être étendu en lui ajoutant de nouvelles fonctionnalités. Par exemple, l'Agent Complexe pourait être exécuté automatiquement avec plusieurs scénarios d'opération différents [6]. On pourrait également considérer plusieurs Agents Complexes qui représenteraient plusieurs répartiteurs qui communiquent et collaborent ensemble afin d'augmenter la qualité de service..

Modélisation plus réaliste : Une variante du modèle présenté au chapitre 3, et qui colle davantage à la réalité, a été abordée mais n'a pas encore été complètement implémentée. On assume ici que la connaissance d'un événement ayant un impact sur le temps de parcours d'un véhicule est immédiate (ou quasi-immédiate). Autrement dit, on n'a pas à attendre jusqu'au moment d'arrivée prévu du véhicule à sa destination courante pour réaliser qu'un événement anormal s'est produit. Bien sûr, la durée ou l'impact de l'événement sur le temps de parcours du véhicule doit alors être estimé, car on ne peut alors en avoir une connaissance exacte. Dans un premier temps, on pourrait toutefois supposer que cet estimé est exact. Ceci implique également que les autres véhicules pourraient être redirigés vers des requêtes se trouvant sur la route planifiée du véhicule en difficulté, et ce avant même d'arriver à leur destination courante ce qu'on appelle "diversion" dans la littérature. Le même raisonnement s'applique dans le cas du véhicule en difficulté (i.e., il n'y aurait plus nécessité pour lui de se rendre à sa destination courante, et il pourrait donc être redirigé ailleurs). Ce modèle est évidemment plus complexe que le précédent, mais il est aussi plus réaliste, car il suppose un mode de communication plus continu entre la centrale de répartition et les véhicules.

En se basant sur l'estimé (qu'on considère fiable), il deviendrait alors possible de réaffecter en bloc toutes les requêtes pour lesquelles le nouveau temps d'arrivée prévu du véhicule dépasse la tolérance (voire Figure 5.1).

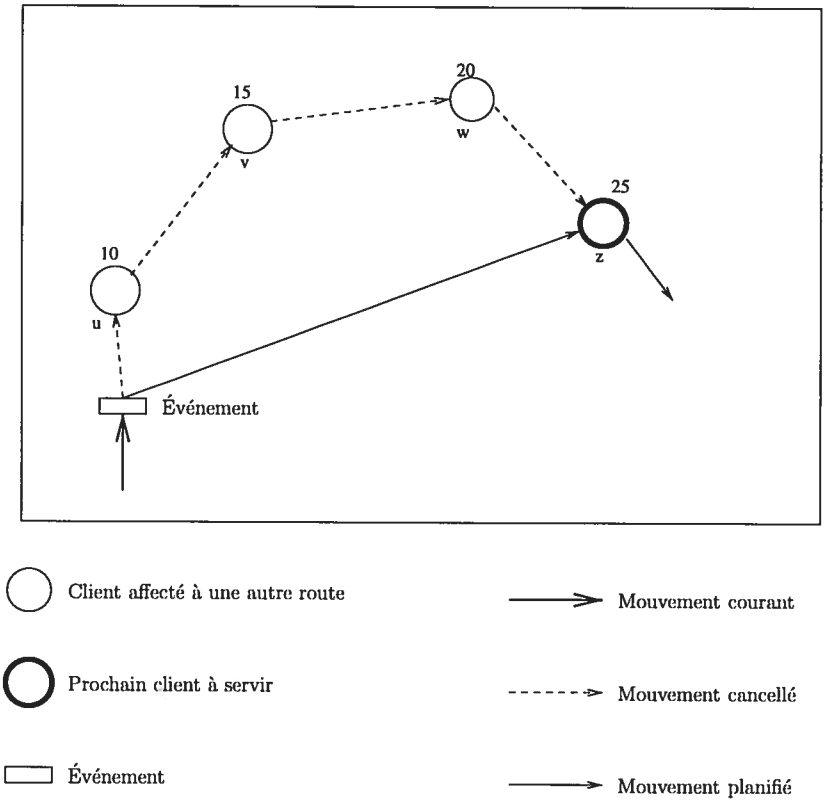

FIG. 5.1 – Réaffectation en bloc des clients u, v, w selon l'estimé

Il existe donc de multiples avenues de recherche qu'il serait intéressant d'explorer. En ce sens, ce mémoire ne représente qu'un premier pas dans l'étude des problèmes de nature dynamique.

Bibliographie

[1] K. Baclawski. Java rmi tutorial. *Northeastern University, Boston, USA.* http ://www.ccs.neu/home/kenb/com3337/rmi_tut.html.

[2] M.O. Ball, T.L. Magnanti, C.L. Monna, and G.L. Nemhauser. Network routing, handbooks in operations research and management science 8, North-Holland : Amsterdam, 1995.

[3] W. Bell, L.M. Dalberto, M.L. Fisher, A.J. Greenfield, R. Jaikumar, P. Kedia, R.G. Macj, and P.J. Prutzman. Improving the distribution of industrial gases with an on-line computerized routing and scheduling optimizer. *Interfaces* 13, 4–23, 1983.

[4] I. Benyahia and M. Hilali. An adaptive framework for distributed complex applications development. *IEEE TOOLS USA 2000 International Conference on Technology of Object-Oriented Languages and System*, Santa Barbara, 2000.

[5] I. Benyahia and J.Y. Potvin. Decision support for vehicle dispatching using genetic programming. *IEEE Transaction on Systems* 28, 306–314, 1998.

[6] I. Benyahia and J.Y. Potvin. A framework architecture for information management in dynamic vehicle dispatching. *The 34th Hawaii International Conference on System Sciences*, 2001.

[7] D.J. Bertsimas and G. Van Ryzin. Stochastic and dynamic vehicle routing problem in the euclidean plane. *Operations Research* 39, 601–615, 1991.

[8] D.J. Bertsimas and G. Van Ryzin. Stochastic and dynamic vehicle routing in the euclidian plane with multiple capacitated vehicules. *Operations Research* 41, 60–76, 1993.

[9] L. Bianchi. Notes on dynamic vehicle routing. *Technical Report IDSIA-05-01*, 1989.

[10] L. Brotcorne, L. Farand, G. Laporte, and F. Semet. Impacts des nouvelles technologies sur la gestion des systémes de véhicules d'urgence. *Technical Report CRT-99-39, Centre de recherche sur les transports, Université de Montréal, Canada*, 1999.

[11] H. M. Deitel and P. J. Deitel. *Java how to program.* Prentice Hall, 2e edition.

[12] M. Dror and W.B. Powell (Eds.). Special issue on stochastic and dynamic models in transportation. *Operations Research* 41, 1–235, 1993.

[13] ENSTA. Interface graphique utilisateur(gui). http ://www.ensta.fr/∼in204td2/gui/.

[14] M. Gendreau, P. Badeau, F. Guertin, J.Y. Potvin, and E. Taillard. A solution procedure for real-time routing and dispatching of commercial vehicles. *Proceedings of the Third World Congress on Intelligent Transportation Systems*, Orland, FL, 1996.

[15] M. Gendreau, F. Guertin, J.Y. Potvin, and R. Séguin. Neighborhood search heuristics for a dynamic vehicle dispatching problem with pick-ups and deliveries. *Technical Report CRT-98-10, Centre de recherche sur les transports, Université de Montréal, Canada*, 1998.

[16] M. Gendreau, F. Guertin, J.Y. Potvin, and E. Taillard. Parallel tabu search for real-time vehicle routing and dispatching. *Transportation Science* 33, 381–390, 1999.

[17] M. Gendreau and J.-Y. Potvin. Dynamic vehicle routing and dispatching. in Fleet Management and Logistics, T.G. Grainic and G. Laporte (Eds.) 115–126, Kluwer : Boston, 1998.

[18] F. Glover. Tabu search-part I. *ORSA journal on computing* 1, 190–206, 1989.

[19] F. Glover. Tabu search-part II. *ORSA journal on computing* 2, 4–32, 1990.

[20] F. Glover. Ejection chains, reference structures and alternating path methods for traveling salesman problems. *Discrete Applied Mathematics* 65, 223–253, 1996.

[21] T.G. Grainic and G. Laporte (Eds). *Fleet management and logistics*. Kluwer : Boston 1998.

[22] S. Grolimund and G. Ganascia. Recherche tabou, intensification. *Université Pierre et Marie Curie-Paris, France*.
http ://www.liafa.jussieu.fr/web9/rapportrech/rapport/1995/cabatase.ps.

[23] S. Ichoua. Problème de gestion de flottes de véhicule en temps réel. *Thèse de Doctorat, Université de Montréal, Canada*, 2001.

[24] S. Ichoua, M. Gendreau, and J.Y. Potvin. Diversion issues in real-time vehicle dispatching. *Transportation Science* 34, 426–438, 2000.

[25] S. Ichoua, M. Gendreau, and J.Y. Potvin. Exploiting knowledge about future demandes for real-time vehicle dispatching. *Technical Report CRT-2001-45, Centre de recherche sur les transports, Université de Montréal, Canada*, 2001.

[26] S. Ichoua, M. Gendreau, and J.Y. Potvin. Vehicle dispatching with time-dependent travel times. *European Journal of Operational Research* 144, 379–396, 2003.

[27] Sun Java. Java(TM) Foundation Classes : Cross-Platform GUIs & Graphics. http ://java.sun.com/products/jfc/.

[28] Sun Java. *Trail : Creating a GUI with JFC/Swing.*
http ://java.sun.com/docs/books/tutorial/uiswing/index.html.

[29] Sun Java. Java(tm) look and feel design guidelines : Adv topic.
http ://developer.java.sun.com/developer/Books/gui/lookandfeel/.

[30] Sun Java. Java(tm) remote method invocation.
http ://java.sun.com/products/jdk/rmi/.

[31] J.J. Jaw, A.R. Odoni, H.N. Psaraftis, and N.H.M. Wilson. A heuristic algorithm for the multi-vehicle advance request dial-a-ride problem with time windows. *Transportation Research* 20B, 243–257, 1986.

[32] A. Larsen. The dynamic vehicle routing problem. *Phd thesis, Technical University of Denmark*, 2000.

[33] Y.P. Ma. Xmlfinder : an intelligent agent based on cbr for e-commerce. *Mémoire de maîtrise, Université de Montréal, Canada*, 2001.

[34] H.S. Mahmassani, Y.Kim, and P.Jaillet. Local optimization approaches to solve dynamic commercial fleet management problems. *Transportation Research Record* 1733, 71–79, 2000.

[35] C. Malandraki and M.S. Daskin. Time-dependent vehicle routing problems : Formulations, properties, and heuristic algorithms. *Transportation Science* 26, 185–200, 1990.

[36] M. Millier. Software agents.
http ://www.acm.org/sigchi/chi97/proceedings/ tutorial/mm.htm.

[37] A.S. Minkoff. A markov decision model and decomposition heuristic for dynamic vehicle dispatching. *Operations Research* 41, 77–90, 1993.

[38] J.M. Pelleu-Tchetagni. Procédure de diversification pour la résolution des problèmes stochastiques de tournées de véhicules par l'algorithme tabou. *Mémoire de Maîtrise, Université de Montréal, Canada*, 2001.

[39] C. Petrie and S. Goldmann. Agent-based project management. http ://www-cdr.stanford.edu/ProcessLink/papers/DPM/dpm.html.

[40] J.Y. Potvin. Generic algorithms for the traveling salesman problem. *Annals of Operation Research*, 63, 339–370, 1993.

[41] J.Y. Potvin. The traveling salesman problem : A neural network perspective. *ORSA journal on computing* 5, 328–348, 1993.

[42] W.B. Powell, P. Jaillet, and A. Odoni. Stochastic and dynamic networks and routing, network routing, Handbooks in operations research and management science. M.O. Ball et al. (Eds.) 8, 141–295, Elsevier, 1995.

[43] W.B. Powell, M.T. Towns, and A. Marar. On the value of optimal myopic solutions for dynamic routing and scheduling problems in the presence of user non compliance. *Transportation Science* 34, 67–85, 2000.

[44] H.N. Psaraftis. A dynamic programming solution to the single vehicle many-to-many immediate request dial-a ride problem. *Transportation Science* 2, 130–154, 1980.

[45] H.N. Psaraftis. Dynamic vehicle routing problems, dans vehicle routing : Methods and studies. B.L. Golden and A.A. Assad (Eds), 223–248, North–Holland : 1988.

[46] H.N. Psaraftis. Dynamic vehicle routing : Status and prospects. *Annals of Operations Research* 61, 143–164, 1995.

[47] S.J. Raftis. Routing and scheduling of vehicles and crews. *Computers and Operations Research* 10, 65–211, 1983.

[48] G.S. Raj. A detailed comparition of corba, dcom and java/rmi. http ://my.execpc.com/ gopalan/mics/compare.html.

[49] D. Reilly. Introduction to java rmi. http ://www.javacoffeebreak.com/articles/javarmi.html.

[50] S. Roy. Routing and scheduling of transportation services for the disabled : Summary report. *Technical Report CRT-473A, Centre de recherche sur les transports, Université de Montréal.*, 1984.

[51] E.M. Schaffer. How to develop an effective gui standard. 1993. http ://www.humanfactors.com/downloads/guistandards.pdf.

[52] Y. Shen, J.Y. Potvin, J.M. Rousseau, and S. Roy. A computer assistant for vehicle dispatching with learning capabilities. *Annals of Operations Research* 61, 189–211, 1995.

[53] M.M. Solomon. Algorithms for the vehicle routing and scheduling problems with time window constraints. *Operations Research* 35, 254–265, 1987.

[54] P. Soriano and M. Gendreau. Fondements et applications des méthodes de recherche avec tabous. *Technical Report CRT-969, Centre de recherche sur les transports, Université de Montréal, Canada*, 1994.

[55] E.D. Taillard. Parallel iterative search methods for the vehicle routing problem. *Networks* 23, 661–673, 1993.

[56] E.D. Taillard, P. Badeau, M. Gendreau, F. Guertin, and J.Y. Potvin. A tabu search heuristic for the vehicle routing problem with soft time windows. *Transportation Science* 31, 170–186, 1996.

[57] The Intelligent Software Agents Lab-The Robotics Institute-Garnegie Mellon University. Multi-agent systems. http ://www-2.cs.cmu.edu/ softagents/multi.html.

[58] L. Vanhelsuwe. *Java.* Publishing house of electronics industry, 1e edition, 1997.

[59] N. Wijngaards and F.Brazier. Distributed agents. Intelligent Interactive Distributed Systems Department of Computer Science, Vrije Universiteit Amsterdam. http ://dsonline.computer.org/agents/agentsabout.htm.