Université de Montréal

Recherches coopératives pour la résolution de problèmes d'optimisation combinatoire

par
Alexandre Le Bouthillier

Département d'informatique et de recherche opérationnelle
Faculté des arts et des sciences

Thèse présentée à la Faculté des études supérieures
en vue de l'obtention du grade de Philosophiæ Doctor (Ph.D.)
en informatique

Août, 2006

Université de Montréal
Faculté des études supérieures

Cette thèse intitulée:

Recherches coopératives pour la résolution de problèmes d'optimisation combinatoire

présentée par:

Alexandre Le Bouthillier

a été évaluée par un jury composé des personnes suivantes:

Jean-Yves Potvin,	président-rapporteur
Peter Kropf,	directeur de recherche
Teodor Gabriel Crainic,	codirecteur
Michel Gendreau,	membre du jury
Éric Taillard,	examinateur externe
Gilbert Babin,	représentant du doyen de la FES

Thèse acceptée le: 13 mars 2007

RÉSUMÉ

Faire plus avec moins : un concept d'actualité que les entreprises, les organismes gouvernementaux et les individus doivent appliquer afin de préserver l'environnement, maximiser leurs investissements, augmenter la satisfaction des clients ou rester compétitifs dans un marché de mondialisation. Tous ces objectifs ont en commun l'optimisation, que ce soit de ressources financières, humaines, temporelles ou matérielles. Chacune d'entre elles engendre un ensemble de décisions complexes qui ne peut pas être résolu efficacement avec des outils informatiques de gestion ou encore moins manuellement de par leur nature combinatoire.

Un des objectifs de la recherche opérationnelle est de fournir des méthodes de résolution efficaces aux problèmes d'optimisation à nature combinatoire. Ces méthodes d'optimisation s'appuient sur des algorithmes traditionnellement séquentiels. Cependant, pour les problèmes de plus grande taille ou même certains problèmes complexes de taille moyenne, auxquels appartiennent les exemples cités ci-dessus, la puissance de calcul d'un seul ordinateur n'est pas suffisante. Grâce à l'accessibilité grandissante de systèmes parallèles qui combinent la puissance de calcul de plusieurs ordinateurs ou CPU, une approche très prometteuse est apparue pour la résolution de ces problèmes.

Nous étudions un des paradigmes de parallélisme, à savoir la recherche métaheuristique coopérative, qui permet l'échange d'informations entre diverses méthodes de recherche. Les mécanismes d'identification de patrons d'éléments de solutions sont explorés afin de guider la recherche de solutions pour en améliorer la robustesse, la qualité et la performance.

La résolution de problèmes complexes comme ceux de tournées de véhicules avec fenêtres de temps (VRPTW) démontre que l'objectif est atteint. Il se manifeste par une accélération linéaire de la coopération et une amélioration des résultats connus.

Mots clés: parallélisme, optimisation, métaheuristique, recherche coopérative, problèmes de tournées de véhicules avec fenêtres de temps.

ABSTRACT

Doing more with less: this concept of current interest is one that businesses, government organizations and individuals must put into practice in order to preserve the environment, maximize their investments, improve customer satisfaction, and remain competitive in a global market. These objectives have the optimization of resources in common, be they financial, human, temporal or physical. Such optimization require for a set of complex decisions that cannot be effectively resolved using traditional computer management tools, or even less so with a manual approach. This is due to their combinatorial nature.

One of many objective of operations research is to allow the effective resolution of a number of combinatorial optimization problems traditionally using sequential optimization methods. However, for large-sized problems, and even for some complex medium-sized problems, including those listed above, the computing power of a single computer is not sufficient. Thanks to the increasing accessibility of parallel systems, which combine the computing power of several computers or CPUs, there is now a very promising approach to efficiently resolve these types of combinatorial problems.

We study cooperative searching, one of the paradigms of parallelism, which allows information to be exchanged between various meta-heuristics. The objectives of the study deal with the creation of mechanisms to identify promising search pattern, and to dynamically guide the search to improve performance and robustness of the search. The resolution of complex problems such as vehicle-routing problems with time windows (VRPTW) shows that the objective is achieved, as demonstrated by a linear acceleration of the cooperation and by an improvement in the known results of the VRPTW. Thus, the proposed dynamic guiding method improves the quality and robustness of the cooperative search.

Keywords: parallel computing, combinatorial optimization, meta-heuristics, cooperative search, vehicle routing problems with time windows.

TABLE DES MATIÈRES

LISTE DES TABLEAUX

LISTE DES FIGURES

LISTE DES SIGLES

CNV	Cumulative Number of Vehicle - Nombre total de véhicules
CPU	Central Processing Unit - Unité centrale de traitement
CTD	Cumulative Total Distance - Distance cumulative totale
EA	Evolutionary Algorithms - Algorithmes évolutifs
ER	Edge recombination - Combinaison génétique basée sur les arcs
GA	Genetic Algorithm - Algorithme génétique
GENIUS	GENeralized Insertion + US - Insertion généralisée + US
LNS	Large Neighborhood Search - Recherche à large voisinage
MIMD	Multiple Instruction, Multiple Data - Instructions multiples - Multiples données
MPI	Message Passing Interface - Librairie de gestion de message entre processus
MPSS	Multiple Point Single Strategy - Point de départ multiple et stratégie unique
MPDS	Multiple Point Different Strategy - Point de départ et stratégies multiples
OX	Order Cross-over - Combinaison génétique basée sur l'ordre des noeuds
OSI	Open Systems Interconnection - Modèle d'interconnexion des systèmes ouverts
RAM	Random Access Memory - Mémoire vive à accès aléatoire
SA	Simulated Annealing - Recuit simulé
SPSS	Single (Initial) Point Single Strategy - Point de départ et stratégie uniques
SPDS	Single Point Different Strategy - Point de départ unique et différentes stratégies

TS	Tabu Search - Recherche avec tabous
US	Unstringing-Stringing heuristic - Heuristique de détente-compression
UT	Unified Tabu - Tabou unifié
VNS	Variable Neighborhood Search - Recherche à voisinage variable
VRPTW	Vehicle Routing Problem with Time Windows
VRPTW	Problème de tournées de véhicules avec fenêtres de temps

(dédicace) À mes parents, à Florence.

REMERCIEMENTS

Je remercie mes directeurs, Teodor Gabriel Crainic et Peter Kropf, qui m'ont offert la chance de travailler avec eux, avec leur équipe et le plus beau cadeau qui peut se transmettre : le goût de la recherche. Je leur dois toute mon admiration et un profond respect pour la patience dont ils ont fait preuve et pour l'accompagnement fourni durant mon cheminement académique. Ce fut un privilège de travailler avec eux et j'espère pouvoir poursuivre cette collaboration. "Danke - Multumiri".

Le FQRNT, le CRSNG, le CIRRELT, l'ATC (Delcan), le CTRF (Bombardier) ainsi que l'Institut d'informatique de l'Université de Neuchâtel m'ont fourni leur soutien financier durant ces dernières années et je leur en suis reconnaissant. Le RQCHP, l'Université de Montréal, l'Université de Sherbrooke, l'Université de Paderborn et l'Université de Neuchâtel ont été d'une aide précieuse en fournissant un accès à leurs ressources informatiques. Je remercie Daniel Charbonneau, François Guertin, Gianni Gasparotto et Russell Standish.

Ma reconnaissance va à Jean-François Cordeau, Gilbert Laporte et Michel Gendreau qui ont permis l'utilisation des métaheuristiques « Unified Tabu » et « Taburoute » dans l'architecture parallèle coopérative et qui ont contribué à la qualité des résultats produits. Merci à Jean-Yves Potvin pour ses commentaires et à Éric Taillard pour ses travaux sur la mémoire adaptative.

Je tiens à remercier ma mère, mon père et ma famille pour le soutien moral qu'ils m'ont prodigué durant ces dernières années et qui m'a permis de poursuivre ma vocation. J'ai une pensée particulière pour ma conjointe Florence qui m'a fourni une aide précieuse dans la révision des textes français de cette recherche et les encouragements nécessaires pour les produire.

Je suis reconnaissant envers Marie-Claude pour sa compréhension et son aide qui m'ont permis d'avancer dans mon accomplissement académique et professionnel.

Il me reste bien sûr à remercier les personnes qui me sont proches, plus particulièrement Louis-Martin, Alexandre, Bernard, Ioan, Andreas, Benjamin et bien sûr Stéphane et Babak qui m'ont permis de me concentrer sur mes études.

CHAPITRE 1

INTRODUCTION

1.1 Problèmes de décisions : contexte et problématique

Cette thèse s'intéresse aux méthodes parallèles de résolutions efficaces et robustes pour des problèmes d'optimisation de grandes tailles.

Un problème de d'optimisation consiste à trouver les meilleures affectations de valeurs à un ensemble de variables. Dans ce contexte, on cherche à maximiser ou à minimiser l'objectif du problème, c'est-à-dire trouver la meilleure solution au problème créant le plus grand bénéfice ou le moindre coût selon le cas.

Certains problèmes d'optimisation sont difficiles à résoudre en raison de leur taille, leur formulation mathématique ou leur type de données. Deux techniques de résolutions sont utilisées : les techniques mathématiques et les métaheuristiques.

La solution optimale d'un programme mathématique formulant un problème de décision peut-être trouvée pour des problèmes de taille plus ou moins grande par des techniques mathématiques selon les problèmes.

Pour des problèmes de plus grandes tailles, l'exploration implicite effectuée par les techniques mathématiques de toutes les solutions n'est pas envisageable dans un temps limité. Les métaheuristiques sont alors utilisées.

Les métaheuristiques forment une famille d'algorithmes d'optimisation (également appelés algorithmes d'approximation) issue de la recherche opérationnelle visant à résoudre des problèmes de décisions difficiles de plus grande taille. Les métaheuristiques sont généralement des algorithmes stochastiques itératifs, qui progressent vers un optimum par échantillonnage d'une fonction objectif. Elles se comportent comme des algorithmes de recherche, tentant d'apprendre les caractéristiques d'un problème afin de trouver une approximation de la meilleure solution.

Il existe un grand nombre de métaheuristiques, allant de la simple recherche locale à des algorithmes de recherche globale. Ces méthodes utilisent un haut niveau

d'abstraction, leur permettant d'être adaptées à une large gamme de problèmes.

Nous pouvons premièrement constater que les techniques mathématiques ou les métaheuristiques résolvent efficacement certains problèmes de décisions de tailles plus ou moins grandes. Toutefois, plusieurs problèmes ne peuvent être résolus efficacement lorsque leur taille dépasse un certain seuil. Ils sont difficilement résolus avec un temps de calcul raisonnable et un espace mémoire limité.

Deuxièmement, une revue des résultats de littérature confirme qu'il n'existe pas d'instance de métaheuristiques permettant d'obtenir les meilleurs résultats sur tous ces problèmes.

Face à ces deux constats, qui constituent la problématique centrale de cette recherche, le parallélisme offre des perspectives intéressantes [43]. Il permet d'effectuer des résolutions plus rapides et plus robustes. Les principes du parallélisme sont détaillés dans la prochaine section.

1.2 Trois approches de parallélisation

Le parallélisme est une approche algorithmique qui décompose le travail sur un ensemble de processeurs. L'objectif de la parallélisation des méthodes de résolution est de trouver des solutions plus rapidement ou de résoudre des problèmes de plus grande taille ou plus complexes. Les approches parallèles se divisent en trois catégories : la décomposition de domaine de solution, la décomposition fonctionnelle et la recherche indépendante/coopérative.

La *décomposition de domaine de solution* consiste à diviser les tâches de résolution de sous-domaines de solutions sur p processeurs. Elle permet d'obtenir une réduction maximale théorique du temps de calcul réel qui est égale au nombre de processeurs p.

La *décomposition fonctionnelle* de l'espace divise l'espace de recherche en sous-problèmes. Chaque sous-problème est ainsi attribué à un processeur qui le résoud de manière individuelle. Les sous-solutions sont ensuite recombinées pour l'obtention de la solution au problème initial.

La *recherche indépendante* fournit le meilleur résultat produit par un ensemble de méthodes de résolutions (métaheuristiques ou méthodes exactes) exécutées en parallèle. Basée sur le même principe, la *recherche coopérative* enrichit l'exploration de l'espace de solutions par un partage de l'information recueillie durant la recherche. Cet échange d'information permet ainsi d'obtenir de meilleurs résultats que ceux produits par les méthodes indépendantes. Il en résulte généralement une recherche de qualité supérieure, plus rapide et plus robuste.

Le milieu scientifique reconnaît aujourd'hui les bénéfices de la recherche coopérative. De surcroît, les récentes études (Crainic et al. [27]) laissent entrevoir de nouvelles perspectives au sein de ce paradigme. Toutes les études ont démontré que les méthodes de recherche combinant les qualités de diverses méthodes sous-jacentes via la coopération permettent de produire rapidement et avec constance des solutions de qualité pour différents types de problèmes (conception et planification de réseau de transports (Crainic, Toulouse and Gendreau [30]), partitionnement de circuits (Aiex *et al.* [2]), problèmes de tournées de véhicules avec fenêtres de temps (VRPTW)(Le Bouthillier and Crainic [77], Le Bouthillier et al. [79]).

1.3 Défis et question de recherche

La mise en oeuvre d'un mécanisme de coopération ne garantit pas de facto une amélioration de la qualité des solutions ni de leur robustesse. Il ne s'agit pas, par exemple, d'augmenter le nombre de métaheuristiques pour accroître l'efficacité, ni d'échanger la meilleure solution trouvée entre les méthodes de recherches. Afin d'obtenir l'augmentation de performance escomptée, l'accent doit plutôt être mis sur la gestion de l'information, c'est à dire évaluer, fusionner, traiter et extraire l'information pertinente et la distribuer de manière efficace. Cette information doit être transmise de manière sélective, afin d'assurer la création de trajectoires de recherches qui ne se recoupent pas. Il est aussi nécessaire de faire dévier les recherches hors des espaces de solutions non prometteuses. Pour expliquer le comportement coopératif et en démontrer la faisabilité, il est nécessaire d'effectuer de nombreuses

expériences comparatives issues de méthodes avec et sans coopération. La reproductibilité est aussi un facteur important. La conception doit donc être orientée vers une méthode simple.

Notre question de recherche : Quelle est l'influence d'un contrôle global de la recherche de solutions sur 1) la qualité des solutions produites et 2) la robustesse de la recherche coopérative ?

1.4 Objectifs et réalisations

Notre objectif est d'explorer dans cette recherche les mécanismes permettant d'enrichir la coopération, plus particulièrement ceux qui s'appliquent à la gestion centralisée des informations obtenues au cours de la recherche et au guidage fixe et dynamique de la coopération. Ces axes ont été choisis en raison de leur complémentarité, de leur reproductibilité et des améliorations applicatives attendues.

La qualité des résultats obtenus sur le très populaire « problème de tournées de véhicules avec fenêtres de temps » (VRPTW) résulte de la recherche effectuée par de nombreuses équipes. Il s'en est suivi de nombreuses publications qui ont généralement amélioré la connaissance du problème et sa résolution. S'attaquer à cette thématique a paru comme une motivation additionnelle à la confection d'un algorithme parallèle. Notre objectif est de concevoir une architecture coopérative qui produit des résultats de qualité rapidement et avec robustesse.

Les réalisations principales de cette recherche sont :

- Une architecture de recherche coopérative possiblement applicable à tout un ensemble de problèmes combinatoires ;
- Une évaluation empirique détaillée du comportement de la coopération et des mécanismes de communications ;
- L'identification de patrons (éléments communs) d'éléments prometteurs ou non au sein des solutions explorées ;
- Une recherche guidée dynamiquement (par fixation/interdiction d'éléments de solutions) qui accentue l'orientation des recherches vers les espaces de

solutions prometteuses et évitant ceux qui le sont moins, améliorant ainsi de plus de 100% la qualité du guidage ;

- Une amélioration de la robustesse des résultats rapportés dans la littérature pour le problème du VRPTW : plusieurs nouveaux résultats et le deuxième meilleur algorithme[1] en terme de la moyenne des résultats sur des problèmes de 100 à 1000 noeuds.

1.5 Organisation de la thèse

Cette thèse est divisée en quatre chapitres principaux qui correspondent aux axes précités dont ont découlé des publications scientifiques. Chaque chapitre contient une introduction qui met en contexte et effectue un lien avec les autres axes de recherche. Suivent une recension de la littérature, la méthodologie de recherche ainsi que la copie de l'article dans son intégralité avec permission de l'éditeur. Finalement, une synthèse présente une discussion des résultats, une conclusion et une bibliographie.

Le premier chapitre est une introduction à la recherche coopérative. Nous y présentons le concept général de la coopération et une revue de littérature. L'article scientifique qui y est inclus détaille les méthodes de recherches impliquées dans la coopération, à savoir, les recherches avec tabous, les algorithmes évolutifs, ainsi que les méthodes de post-optimisation et de génération de solutions. Les travaux présentés dans ce chapitre sont une extension de ceux publiés dans (Le Bouthillier [75]) (LC02). Ils constituent la deuxième version (LC03) de notre architecture parallèle. Dans cette recherche, nous avons raffiné le mécanisme de coopération. De même, il nous a été possible d'améliorer cinq des meilleurs résultats connus du VRPTW et surtout d'offrir une plus grande robustesse aux calculs avec la meilleure moyenne de résultats pour les problèmes de 200 noeuds et avec les six meilleures moyennes pour des classes de problèmes variant de 100 à 1000 noeuds.

Le deuxième chapitre détaille le concept de recherche coopérative. Dans une

[1] SINTEF http ://www.sintef.no/static/am/opti/projects/top/

évolution naturelle de ce paradigme, nous passons de simples échanges de solutions via un entrepôt de données à une orientation de la coopération vers les endroits prometteurs de l'espace de solutions. La publication figurant dans ce chapitre propose un mécanisme de guidage basé sur la fixation et l'interdiction d'éléments qui sont communs au sein de solutions prometteuses ou non. Il devient dès lors possible de guider les méthodes de recherches en imposant ces patrons d'éléments prometteurs ou non. Par ailleurs, ce mécanisme intensifie ou diversifie les recherches par la variation de la longueur des patrons d'éléments imposés ou interdits. Par exemple, un patron fixant un nombre important d'éléments aura pour effet de concentrer (intensifier) la recherche sur une région particulière de l'espace de solutions. Ce mécanisme de guidage (LCK05) a permis l'amélioration des résultats des recherches antérieures (LCK02 et LCK03). Les perspectives soulignent qu'il serait intéressant d'employer ce mécanisme de guidage avec des méthodes parallèles non coopératives de même qu'avec des méthodes séquentielles qui disposent ou non de mémoire.

Le troisième chapitre traite de l'abstraction des méthodes de recherches par l'utilisation d'un générateur aléatoire de solutions. Ceci permet une analyse détaillée sur les mécanismes de guidage que peuvent offrir l'imposition et l'interdiction de patrons d'éléments de solutions. Avec les éléments d'informations obtenus dans cette analyse, nous avons pu proposer une méthode de guidage dynamique basée sur l'entropie des éléments de la population. Il devient donc possible de s'adapter à chaque type de problème sans spécifier à priori les phases d'intensification ou de diversification. La méthodologie proposée n'est pas propre à un problème particulier. Elle peut s'appliquer à tout un ensemble de problèmes combinatoires. Les tests effectués sur des problèmes de tournées de véhicules avec fenêtres de temps démontrent une amélioration substantielle grâce à l'utilisation de ce mécanisme de guidage dynamique.

Finalement, le dernier chapitre présente une discussion des contributions et des perspectives qui découlent de ce travail sur les recherches coopératives. L'échange d'informations statistiques sur l'espace de solutions exploré, qu'il soit constitué de solutions améliorantes ou non, offre une vue d'ensemble plus complète. Le méca-

nisme de guidage pourrait ainsi être adapté en fonction du comportement de chacune des méthodes. Cette personnalisation devrait pouvoir fournir une plus grande robustesse aux résultats issus de la recherche coopérative.

CHAPITRE 2

LES RECHERCHES COOPÉRATIVES

Au cours des dernières années, la puissance de calcul des ordinateurs a été combinée en grappes de plus en plus performantes. Ces grappes ont permis la résolution de problèmes combinatoires NP-difficiles d'une importance croissante en taille et en complexité. L'utilisation optimale de cette puissance de calcul grandissante est donc un sujet d'actualité. Les dernières avancées démontrent que les recherches coopératives offrent, dans ce sens, des perspectives intéressantes. Dans la volonté de participer à l'amélioration de cette méthode, nous y avons porté notre choix. Ce chapitre se veut une introduction à la recherche coopérative. Il détaille les méthodes de recherches sélectionnées ainsi que les mécanismes d'échange d'information. Une recension de la littérature situe la coopération au sein des métaheuristiques parallèles et son impact dans la résolution de problèmes combinatoires.

Notre premier article « A cooperative parallel meta-heuristic for the vehicle routing problem with time windows » (Le Bouthillier et al. [77]) est par la suite reproduit dans son intégralité. Il y est question de l'étude des mécanismes de coopération qui permettent la combinaison des avantages de diverses méthodes de recherches hétérogènes. Nous détaillons les différents aspects de la communication et des méthodes de gestion de l'information. Un cadre applicatif permet de valider l'utilisation d'une architecture coopérative pour la résolution d'un problème de VRPTW. Par l'expérimentation, nous démontrons une différence dans les trajectoires de recherches produites par les métaheuristiques lorsqu'elles coopèrent ou non. Il devient dès lors possible de présenter l'architecture coopérative comme une nouvelle métaheuristique en soi avec ses caractéristiques et ses comportements propres. L'architecture coopérative développée ne nécessite pas de calibration de paramètres et est assez générique pour pouvoir être déployée sur un parc de noeuds de calcul configuré en grappe, de même que sur un système hautement parallèle de type MIMD (Multiple Instruction - Multiple Data), une architecture parallèle

où plusieurs unités fonctionnelles effectuent différentes opérations sur diverses données. L'implémentation est effectuée en C++ avec la librairie MPICH pour les communications entre processus.

Les contributions proposées s'articulent autour de l'influence qu'ont l'échange d'information et la gestion de l'entrepôt de données sur le comportement global du système. Finalement, une synthèse permet une discussion des résultats et la présentation des perspectives.

2.1 Revue de la littérature

Les différents aspects qui ont contribué à l'élaboration des prémisses de cette thèse sont présentés dans cette revue de la littérature. Nous y présentons les recherches parallèles indépendantes (sans communications) ainsi que les types de parallélisme avec communications et plus particulièrement les méthodes coopératives. Nous effectuons une revue des différentes structures de contrôle utilisées, de l'influence des méthodes de recherche, ainsi que des modes de communication utilisés.

2.1.1 Méthodes indépendantes concurrentes

La nécessité de résoudre des problèmes de taille importante a poussé le développement de recherches parallèles performantes. Une des méthodes pour l'obtention de gains de performance dans la recherche de solutions est la parallélisation de métaheuristiques. Toulouse et al. [105] divisent la parallélisation de recherches locales en trois grandes catégories : parallélisation de bas niveau (i.e., maître-esclave), méthodes de partitionnement de l'espace des solutions et résolutions par des recherches concurrentes. La résolution concurrente existe sous forme de recherches indépendantes et de recherches coopératives.

La parallélisation par recherches indépendantes est une méthode simple qui peut utiliser une puissance de calcul distribuée, dans le but d'obtenir plus rapidement des résultats lorsque les paramètres ont une influence élevée sur l'exploration

de l'espace de solutions. Puisque les métaheuristiques nécessitent souvent un calibrage spécifique par type de problèmes, il devient intéressant d'effectuer différentes exécutions indépendantes avec des paramètres différents et de retenir la meilleure solution. L'utilisation de différentes solutions de départ pour chaque méthode de résolution permet d'augmenter les chances de trajectoires d'explorations différentes et ainsi de diminuer les risques de recoupement durant l'exploration. Cette approche a été utilisée pour la parallélisation des métaheuristiques itératives dans les recherches avec tabou pour la résolution des problèmes d'affectation quadratique [96], d'ordonnancement de tâches [97], ainsi que de location/allocation avec balancement de charges [31]. Une résolution indépendante en parallèle du même algorithme mais avec différents paramètres produit les même résultats que plusieurs résolutions séquentielles avec ces différents paramètres.

2.1.2 Type de parallélisme avec communication

La taxinomie définie par Crainic et al. [31] permet de couvrir trois dimensions essentielles à la caractérisation du type de parallélisme, soit :

- Cardinalité du contrôle
  - $1-control$
  - $p-control$
- Type de contrôle et de communication
  - Synchronisation rigide
  - Synchronisation de connaissance
  - Coopération totale
  - Recherche Coopérative
- Trajectoire de recherche
  - SPSS Single (Initial) Point Single Strategy
  - SPDS Single Point Different Strategy
  - MPSS Multiple Point Single Strategy
  - MPDS Multiple Point Different Strategy

2.1.2.1 Cardinalité du contrôle

Le contrôle des tâches s'effectue soit par un *processeur* qui divise les tâches en sous-problèmes et recombine ensuite les résultats, soit par $p - processeurs$ $(p \geq 1)$ qui se partagent la résolution du problème principal.

2.1.2.2 Type de contrôle et de communication

Le type de contrôle et de communication détermine le mode d'échange d'informations entre les méthodes. Dans le cadre de la synchronisation rigide, les métaheuristiques doivent arrêter leurs recherches à un moment précis et transmettre l'information requise à un maître. Notons qu'il n'y a aucun échange entre les métaheuristiques et il n'y a présence que d'un seul maître. Au sein de la synchronisation des connaissances, les méthodes de recherches hiérarchiques possèdent une mémoire locale et ont connaissance du problème global à résoudre. La coopération totale implique, pour sa part, différentes méthodes de résolutions communiquant entre elles de façon asynchrone par la diffusion des solutions améliorantes à l'ensemble des méthodes. Finalement, dans la recherche coopérative, l'information locale pertinente de chaque métaheuristique est envoyée à un entrepôt de données avant d'être traitée et mise à disposition de l'ensemble des méthodes. Ce traitement peut impliquer certaines déductions et permet d'une part la mise à jour des connaissances globales et d'autre part le transfert d'une information plus riche par la suite.

Le type de contrôle et de communication peut aussi être caractérisé par quatre facteurs distincts selon Toulouse et al. [105] :

- Type d'information communiquée et représentée entre les métaheuristiques ;
- Méthode de communication entre les métaheuristiques ;
- Topologie de distribution de l'information des résultats et des prévisions de recherche selon ce qui a déjà été fait ;
- Fréquence des communications entre les métaheuristiques.

Le type d'information que s'échangent les méthodes de recherche peut inclure les nouvelles solutions trouvées, les espaces de solutions à élaguer ainsi que les es-

paces de solutions prometteuses. Les méthodes de recherche peuvent communiquer directement entre elles ou via un entrepôt de données.

2.1.2.3 Trajectoire de recherche

La trajectoire de recherche est intimement liée au point de départ (solution initiale), ainsi qu'à la stratégie de parcours de l'espace de solutions. Le type de stratégie peut inclure différentes méthodes de recherche et/ou différents paramètres. Nous obtenons, soit une trajectoire de recherche à partir d'un seul point de départ et d'une seule stratégie, soit plusieurs trajectoires à partir de points de départ et de stratégies différents. Les trajectoires qui résultent de différents points de départ et de différentes stratégies ont moins tendance à souffrir du phénomène de cycle de recherches inter-méthodes (plusieurs méthodes effectuent les mêmes mouvements de recherche). Notons que l'utilisation de métaheuristiques différentes crée une variété plus marquée sur les trajectoires que l'utilisation de paramètres différents.

Dans le cas d'entrepôts de données centraux effectuant la collecte et la gestion de l'information, qui résulte des recherches effectuées par les métaheuristiques, il est primordial de définir des mécanismes qui permettent une répartition de ces informations selon leur diversité et leurs qualités respectives, afin d'assurer des trajectoires de recherches différentes et ainsi éviter une convergence prématurée de la résolution.

2.1.3 Topologies de communications

Les différents types de topologies de communications, tels les arbres, les maillages réguliers et irréguliers, les anneaux, les étoiles ainsi que la communication via un entrepôt de données (*blackboard*), vont influencer la propagation de l'information et ainsi les trajectoires de recherches de chaque métaheuristique impliquée. La fréquence d'échange de solutions entre les méthodes doit aussi être déterminée de façon à ne pas interrompre les recherches dans leurs explorations. Les meilleurs résultats sont obtenus par l'envoi asynchrone de solutions améliorantes. Nous obtenons ainsi

une exploration de l'espace de solutions sans interruption et nous limitons la focalisation de l'entrepôt vers un espace de solutions particulier, qui pourrait représenter un minimum local.

Toulouse et al. [105] indiquent aussi les éléments clefs à considérer dans la création d'une telle architecture. Ils évoquent trois gains possibles du partage de l'information entre les métaheuristiques, à savoir :

- Les mécanismes d'évaluation heuristique peuvent utiliser l'information de plusieurs recherches et ainsi être mieux informés ;
- Il est possible d'utiliser d'autres méthodes de recherches complémentaires, améliorant ainsi les méthodes actuelles, telles les recherches avec tabous ;
- L'information provenant de plusieurs recherches distinctes peut fournir certaines informations relatives à la composition de l'espace de solutions.

Pour la résolution de problèmes combinatoires, il y a émergence de méthodes coopératives, combinant différentes métaheuristiques qui utilisent des points de départ et des paramètres différents (MPMS). La gestion des communications par entrepôts de données permet d'obtenir une vision d'ensemble de l'espace de solutions. De cette façon, elle fournit aux métaheuristiques les informations nécessaires pour une exploration coordonnée. Il est préférable d'avoir des communications orientées par la structure de l'espace de solutions plutôt que par des logiques de synchronisation. Les communications asynchrones semblent donc favorisées.

2.1.4 Recherches coopératives

Selon Huberman [66], Toulouse [105], Crainic [25], Taillard [100] et Le Bouthillier [75], les recherches coopératives qui créent des relations entres les diverses méthodes impliquées offrent des gains de performance généralement substantiels par rapport aux méthodes indépendantes.

Les algorithmes coopératifs définis par Enslow [40] possèdent les quatre caractéristiques suivantes :

- L'exécution concurrente de méthodes de résolution (parallélisme à gros-grains) ;
- La coopération par échange de solutions ;

- Les communications hiérarchiques par l'utilisation d'un modèle client-serveur ;
- L'utilisation de mémoire partagée pour les communications.

Notons que les deux dernières caractéristiques ne sont pas nécessaires.

2.1.5 Mémoire adaptative : l'unification des métaheuristiques à mémoire

Déjà Rochat et Taillard [87] font état d'un concept de mémoire adaptative qui unifie plusieurs concepts de métaheuristiques à mémoire. Comme le mentionnent Taillard et al. [100], les algorithmes génétiques, les recherches par dispersion et les recherches avec tabous utilisent tous des formes de mémoires qui possèdent les particularités suivantes : Premièrement, ils mémorisent des solutions ou des caractéristiques des solutions visitées durant le processus de recherche ; deuxièmement, ils font usage d'une procédure créant une nouvelle solution à partir des informations mémorisées et troisièmement, ils sont dotés d'une recherche locale, que ce soit une méthode gloutonne, une recherche avec tabous élémentaires ou un recuit simulé. Le concept de mémoire adaptative unifie ces trois propriétés. Taillard et al. [100] utilisent la coopération en fournissant une mémoire commune où sont récoltées les sous-solutions fournies par les métaheuristiques. Il devient alors possible de reconstruire de nouvelles solutions et de les transmettre aux métaheuristiques.

2.1.6 Entrepôt de données

Nous utilisons le terme d'entrepôt de données qui reçoit, traite, enrichit et échange les informations reçues par les métaheuristiques impliquées dans la coopération. D'autres termes ont été précédemment employés, à savoir, mémoire centrale, blackboard ou mémoire adaptative. Cette nouvelle terminologie accentue la notion de coopération. Elle est une généralisation du concept qui facilite l'introduction de mécanismes additionnels.

Dans la figure 2.1, on distingue l'entrepôt de données (partie supérieure) ainsi qu'un ensemble de méthodes de résolutions (partie inférieure). La conception d'une

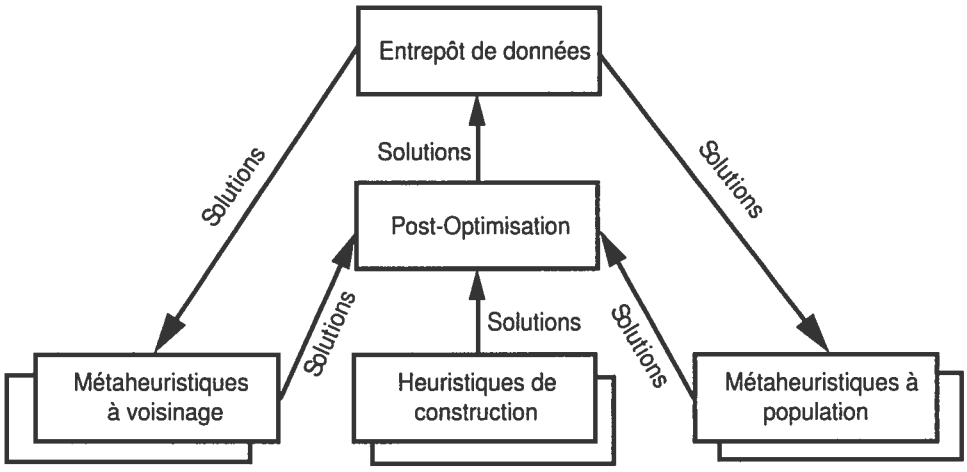

Figure 2.1 – Recherches coopératives

architecture coopérative nécessite, d'une part, la mise en place d'exécutions concurrentes de métaheuristiques et, d'autre part, un mécanisme de couplage permettant le partage d'informations entre ces méthodes de recherches. Les considérations dans la conception d'une telle architecture sont multiples. Elles influencent la qualité, la rapidité ainsi que la robustesse des résultats.

2.1.7 L'auto-organisation

Il est théoriquement possible d'effectuer une recherche indépendante avec un nombre illimité de métaheuristiques. Cependant, la difficulté inhérente aux métaheuristiques est de créer plusieurs stratégies différentes et efficaces pour un même problème. Plus le nombre de métaheuristiques croît, plus il y a redondance entre les trajectoires d'exploration, et plus il y a de stratégies d'exploration inefficaces qui vont utiliser la puissance de calcul en pure perte. S'assurer d'une exploration efficace devient particulièrement pertinent lorsque le nombre de métaheuristiques augmente.

Toulouse [104] a confirmé que l'échange des connaissances auto-organise l'exploration de l'espace de solutions. Ce comportement n'est pas nécessairement lié à une logique d'optimisation. Comme illustration, regardons le lien entre la distance globale de Hamming et l'auto-organisation. Rappelons que la distance de Hamming

$H(v_1, v_2)$ entre deux vecteurs binaires v_i et v_j de même taille correspond au nombre d'entrées qui n'ont pas le même statut dans les deux vecteurs. Par distance globale, nous entendons la somme de toutes les distances de Hamming entre les solutions d'une même population.

Comme le mentionne Toulouse [104], la distance globale de Hamming permet de mesurer le niveau d'organisation de l'espace de solutions exploré. Cette mesure est sensible à tout phénomène de structuration de l'exploration, phénomène qui se manifeste ici par l'exclusion de blocs dans les configurations. Il s'agit de la même auto-organisation qui apparaît dans la théorie de Wolfram [107] sur les automates cellulaires. Lorsqu'un plus grand nombre de blocs est exclu, les configurations tendent vers une uniformisation, ce qui a pour effet de réduire la distance globale de Hamming entre ces configurations. Ces configurations ont tendance à se concentrer sur un sous-ensemble de solutions en fonction de la structure de l'espace de solutions et de la stratégie d'exploration. Les résultats présentés sont assez convaincants pour montrer que l'échange d'information ne contribue pas automatiquement à améliorer l'exploration des métaheuristiques impliquées dans une coopération. Il devient alors pertinent d'identifier les zones où la recherche ne devrait pas avoir lieu, et ainsi d'élaborer des stratégies de coopération plus efficaces.

2.1.8 Modes de communication dans un système coopératif

Le mode d'interaction entre les métaheuristiques est au coeur de tout système coopératif. Les échanges directs entre métaheuristiques pourraient être une méthode efficace de transfert des informations si les coûts de communication étaient négligeables. Par contre, un partage total des informations sans aucun filtrage entraîne une convergence prématurée vers les minima locaux. Les coûts de communication sont relativement élevés comparativement aux coûts de calculs dans les systèmes distribués. Les méthodes de recherches ne peuvent donc pas obtenir une vue d'ensemble du processus de recherche et doivent partager seulement les informations clés. L'échange efficace d'informations entre méthodes de résolution n'est pas chose simple ; la topologie de communication et le type d'informations échangées l'est

encore moins. Le choix des méthodes de résolution qui coopèrent doit prendre en considération des facteurs tels la diversité, la qualité des solutions fournies, la rapidité et la fréquence de découverte de nouvelles solutions de qualité supérieure ainsi que la complémentarité des méthodes de recherche.

Toulouse et al. [105] ont étudié les trois aspects de cette interaction, à savoir les informations à échanger, le moment où effectuer les communications et les métaheuristiques qui doivent être impliquées. L'échange d'informations procure trois avantages :

- Les métaheuristiques ont une meilleure connaissance de l'espace de solutions par leur accès à d'autres mémoires ;
- Le couplage de métaheuristiques qui possèdent un comportement différent procure une complémentarité dans la recherche ;
- Certaines informations additionnelles sur l'espace de solutions peuvent être déduites par la combinaison d'informations échangées entre métaheuristiques.

Notons que p méthodes de recherches indépendantes ont chacune accumulé après t itérations l'information équivalente à au plus t itérations séquentielles. En présence de coopération, après t itérations, chaque méthode a accumulé de l'information équivalente à au plus $p*t$ itérations via l'entrepôt de données. Cette dernière assertion est valide lorsque les trajectoires des méthodes sont distinctes, ce qui est rarement le cas pour l'ensemble de la recherche. Deux facteurs additionnels doivent être considérés : les coûts de communications et la pertinence de l'information.

Le partage du contenu des mémoires locales nécessite des communications interprocessus fréquentes. Tel qu'identifié par Toulouse et al. [105], si it et ct sont les temps requis pour réaliser t itérations par p méthodes indépendantes et coopératives respectivement, il est possible d'évaluer la pertinence de la coopération selon la relation suivante :

$$it \leq ct \leq max\{it, \frac{\text{valeur de la meilleure solution indépendante}}{\text{valeur de la meilleure solution coopérative}} * it\} \qquad (2.1)$$

Dans cette équation, le temps pour effectuer t itérations coopératives est su-

périeur ou égale au temps pour effectuer t itérations séquentielles ; Ce temps est au plus soit it ou soit it multiplié par le ratio entre la valeur de la meilleure solution produite par la recherche indépendante sur celle produite par la recherche coopérative.

Ainsi, plus le partage d'information améliore la qualité des solutions, plus on peut tolérer un niveau élevé des coûts de communication. Il est question de trouver l'équilibre entre le coût de communication et le gain sur la qualité des solutions obtenues.

Huberman et al. [67] notent qu'une connaissance totale avec une mauvaise stratégie peut conduire à un comportement individualiste qui peut entraîner la convergence prématurée de l'espace de solutions. Ils définissent l'utilité d'une communication comme étant l'apport au système en information en fonction des coûts de communication. On observe une relation entre la densité de communication et son utilité. Avec une communication plus fréquente ou plus globale, les métaheuristiques obtiennent une vision plus cohérente des autres. La quantité d'information est réputée comme étant limitée dans la résolution de problèmes distribués. En supposant un coût de communication croissant de façon monotone et un nombre de communications augmentant selon une courbe logarithmique, l'utilité de l'information est maximale où l'écart entre les deux courbes est le plus grand. Avec cette définition d'utilité de communication, il est possible de maximiser le nombre de communications du système.

2.1.9 L'accès à l'information

Le couplage entre méthodes de résolution peut servir à prévenir les cycles de recherche inter-méthodes (plusieurs méthodes effectuent les mêmes mouvements). Par le partage d'informations, il est possible de synthétiser la connaissance de l'espace de solutions et ainsi limiter l'apparition de ces cycles.

Le partage ciblé d'informations, comme les optima locaux, peuvent être bénéfiques à d'autres méthodes. Au contraire, un accès illimité à l'entrepôt de données engendre la dominance des meilleurs individus au détriment de la diversité de popu-

lation. Les algorithmes génétiques font souvent appel à des sélections probabilistes afin d'assurer la diversité d'exploration.

Les informations partagées et non partagées par chaque méthode de résolution sont intimement liées et elles s'influencent mutuellement, phénomène nommé boucle de *rétroaction (feedback)*.

La communication synchrone ou asynchrone exerce une influence sur la boucle de *rétroaction*. Du point de vue expérimental, des études [105] révèlent que les échanges asynchrones d'informations améliorent les performances des recherches coopératives. À l'opposé, l'échange synchrone d'informations dégrade la qualité des solutions obtenues.

Deux hypothèses fondées sur la relation entre la structure de communication et la boucle de *rétroaction* sont émises afin d'expliquer ces résultats. La première hypothèse est basée sur la capacité d'auto-organisation de la boucle de *rétroaction*. La communication asynchrone est préférable aux recherches indépendantes puisqu'elle permet d'adapter l'exploration en fonction de l'espace de solutions. La recherche est ainsi orientée par la structure de l'espace de solutions plutôt que par la logique de synchronisation. La deuxième hypothèse assume que les recherches coopératives qui combinent des méthodes de recherches différentes (i.e., recherches avec tabous et des algorithmes évolutifs) fournissent des résultats de qualité supérieure à celles qui n'utilisent qu'une seule méthode. À noter que ces résultats sont obtenus pour une puissance de calcul équivalente.

2.1.10 Modèle de communication

Différents paramètres influencent les coûts de communications. Selon le tableau 2.1 proposé par Narazaki [84], ils peuvent être classés en trois catégories :

	Dimension	Paramètres de contrôle
1	Espace	Nombre de méthodes de recherche
2	Temps	Fréquence des communications
3	Information	Abstraction et quantité d'informations à envoyer

Tableau 2.1 – Paramètres influençant les coûts totaux de communications

le nombre de méthodes, la fréquence des communications et la pertinence de l'information. Le nombre de méthodes et la fréquence des communications influencent directement le nombre total de communications. La troisième dimension définit le type d'informations pertinentes à transmettre. Elle a été étudiée par de nombreux chercheurs [38] [80] [92], mais rares sont ceux ayant étudié le changement dynamique de la coopération [84].

De nombreux auteurs [66] [1] ont effectué des analyses théoriques montrant le gain de performance de la coopération. Pour leur part, Hogg et al. [62] ont regardé la coopération au sein d'un problème de graphe. Dans la recherche distribuée, Sycara et al. [95] ont étudié la sélection et la transmission d'informations dans une topologie fixe.

Durfee et al. [39] ont proposé une méthode de coopération dans la surveillance distribuée de véhicules. Dans leur modèle, l'espace de solutions est divisé selon les vues de chaque méthode qui influencera la recherche en fonction du niveau d'abstraction effectué. Les tâches sont divisées a priori au sein d'une structure de communication fixe. Bien que le type d'informations échangées soit très pertinent pour l'obtention de résultats de qualité, il demeure spécifique au problème traité.

Narazaki et al. [84] définissent une topologie dynamique de communication entre méthodes de recherches homogènes qui effectuent une résolution de sous-problèmes. Cette topologie évoluera d'une communication sans hiérarchie à une communication hiérarchisée, en fonction de la connaissance du système que possèdent les méthodes. Même si elles ont une vue limitée des autres et ainsi une appréciation incomplète du problème, les méthodes peuvent estimer l'état du système et de leurs environnements respectifs. Cette estimation est basée sur l'homogénéité des méthodes et sur les données historiques. Il est possible de fonctionner avec une connaissance partielle du modèle et des autres métaheuristiques. Les auteurs assument que, en présence d'une homogénéité des méthodes de recherche et d'une continuité dans l'efficacité de calcul, le taux de renouvellement de l'information est sensiblement le même pour toutes les méthodes. Notons qu'en présence de méthodes hétérogènes, cette hypothèse ne s'applique pas. Dans ce cas, la topologie des communications

sera basée uniquement sur leur historique.

La séparation entre les méthodes de recherches et de communications (Narazaki [84]) résulte d'une gestion par modules (recherche, communication et coopération). Chaque méthode de résolution dispose de son propre module de recherche qui effectue une exploration de l'espace des solutions. Les méthodes utilisent le module de communication pour envoyer leurs informations à l'entrepôt de données. Pour sa part, le module de coopération contrôle de façon dynamique la topologie de communication entre les méthodes de façon à maximiser l'efficacité computationnelle.

2.1.11 Temps d'exécution

Regardons le modèle de coopération simplifiée, tel que défini par Narazaki [84], où les méthodes de recherche explorent avec une méthode de type *branch-and-bound* les noeuds d'un sous-problème dans un réseau à état et échangent de l'information au besoin. L'espace de solutions non exploré $S_{(t)}$ à l'étape t est défini récursivement par : $S_{(t)} = (1 - k(n))(S_{(t-1)} - N * p)$ où N est le nombre de méthodes, p est la fraction de l'espace des solutions explorées par la méthode à une étape, $k(n)$ est la coupe de l'espace de solutions engendrée par l'échange des meilleures informations entre N méthodes. À l'état initial, $S_{(0)}$ est défini à 1.

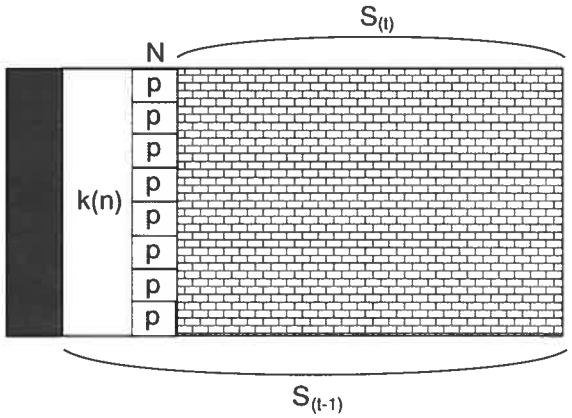

Figure 2.2 – Espace non exploré à l'étape t et $t-1$

La figure 2.2 montre l'espace qui reste à explorer à l'étape $S_{(t-1)}$ en blanc et

carrelée. Elle montre aussi l'espace qui reste à explorer à l'étape $S_{(t)}$ soit la surface carrelée. La surface en noir représente l'espace des solutions explorées avant $S_{(t-1)}$.

En présence de méthodes homogènes, la vitesse moyenne d'exploration de p est identique. La recherche exhaustive termine à l'étape t^* tel que $S_{(t^*)} = 0$. $S_{(t)}$ étant une série géométrique on trouve que t^* devient :

$$t^* = \frac{1}{log(1-k(n))} log(\frac{(1-k(n))N*p}{1-(1-N*p)(1-k(n))}). \tag{2.2}$$

Puisque le temps d'exécution d'une étape est la somme des exécutions des méthodes de résolution et que le temps de communication des n membres est $T(n)$, le temps total d'exécution T est défini par :

$$T = \frac{1+T(n)}{log(1-k(n))} log(\frac{(1-k(n))N*p}{1-(1-N*p)(1-k(n))}), \tag{2.3}$$

en utilisant le temps d'exécution d'une étape comme unité de temps.

En assumant $N*p \ll 1$ et $k(n) \ll 1$ ce qui indique une recherche assez longue, le temps total d'exécution T devient :

$$T \approx \frac{1+T(n)}{N*p+k(n)} \tag{2.4}$$

Nous pouvons diminuer le temps des communications en réduisant le nombre de méthodes impliquées dans les échanges. Le nombre idéal de communications entre méthodes varie en fonction du type de topologie. En effet, en comparant les communications à un seul niveau décisionnel et les communications hiérarchiques, il est possible d'effectuer une estimation de premier ordre du nombre de méthodes requises, de façon à maximiser l'utilité de la communication. Sachant que la valeur espérée de la quantité d'informations utiles obtenue par N méthodes est bornée supérieurement, il existe un optimum global quant au nombre de méthodes à utiliser. Cet optimum existe en autant que le coût de la communication suive une fonction monotone croissante.

2.2 Conclusion

Les limitations des modèles proposés par Narazaki et al. [84] et [66] [1] nécessitent, *a priori*, la connaissance du coût des communications, ce qui est difficile à obtenir dans un système hiérarchique ou un système à grande échelle. De plus, les stratégies de communication sont basées sur l'homogénéité des méthodes et une décomposition du problème.

Nous avons soulevé quelques questions dans cette brève revue des méthodes de recherches coopératives. Par exemple, Narazaki et al. [84] ont défini la notion d'utilité et le nombre de métaheuristiques optimal dans une coopération entre méthodes homogènes. Ils assument que, en présence d'une homogénéité des méthodes et d'une continuité dans l'efficacité de calcul, le taux de renouvellement de l'information d'une méthode est sensiblement le même pour toutes. Il est à noter toutefois, qu'avec des méthodes hétérogènes cette hypothèse ne tient plus. Sur ces considérations, nous avons voulu explorer les performances de méthodes coopératives hétérogènes appliquées à des problèmes NP-difficiles.

Nous détaillons dans la prochaine section la publication [77] qui présente le concept général de la coopération entre méthodes hétérogènes et son application au VRPTW. Les prochains chapitres, pour leur part, sont consacrés à l'identification des espaces de solutions prometteuses, au mécanisme de guidage dynamique et finalement à l'échange d'informations locales issues des méthodes de recherches vers un entrepôt de données.

2.3 Méthodes de résolutions pour le problème de confection de tournées de véhicules

La présentation exhaustive de méthodes de résolution pour le VRPTW ne relève pas des objectifs de cette recherche. Nous référons le lecteur aux revues récentes sur les méthodes parallèles et séquentielles [21][102][72][73][58] [51][37][22][21][9][8].

Dans Solomon et Desrosiers [94], le VRPTW est défini comme suit

$$x_{ij}^v = \begin{cases} 1, & \text{si le vehicule } v \in V \text{ voyage directement du client } i \text{ au client } j \\ 0, & \text{otherwise.} \end{cases}$$

$$min \sum_{v \in V} [\sum_{j \in C} x_{0j}^v + \sum_{i \in N} \sum_{j \in N} c_{ij} x_{ij}^v] \qquad \text{s.a.} \tag{2.5}$$

$$\sum_{v \in V} \sum_{j \in N} x_{ij}^v = 1 \qquad \forall i \in C \tag{2.6}$$

$$\sum_{i \in C} d_i \sum_{j \in N} x_{ij}^v \leq q \tag{2.7}$$

$$\sum_{j \in N} x_{0j}^v = 1 \qquad \forall v \in V \tag{2.8}$$

$$\sum_{j \in N} x_{ij}^v - \sum_{j \in N} x_{ji}^v = 0 \qquad \forall i \in C, v \in V \tag{2.9}$$

$$\sum_{j \in N} x_{jn+1}^v = 1 \qquad \forall v \in V \tag{2.10}$$

$$x_{ij}^v (b_i + t_{ij} - b_j) \leq 0 \forall ij \in C, v \in V \tag{2.11}$$

$$e_i \leq b_i \leq l_i \qquad \forall i \in C \tag{2.12}$$

$$x_{ij}^v \in 0, 1 \qquad \forall i, j \in N, v \in V \tag{2.13}$$

La fonction objectif (2.5) minimise la distance totale parcourue avec un nombre minimal de véhicules. Notons que le coût de chaque véhicule v est suffisamment élevé de façon à minimiser d'abord le nombre de véhicules et ensuite la distance totale parcourue. La contrainte (2.6) garantit que chaque client est assigné à un

seul véhicule. La contrainte (2.7) indique que la capacité des véhicules ne sera pas excédée. (2.8), (2.9), (2.10) représentent les contraintes de flots qui requièrent que chaque véhicule v quitte le dépôt une seule fois, qu'il quitte le noeud $i \in C$ si et seulement s"il a visité ce noeud et qu'il retourne au dépôt (contrainte redondante conservée ici pour mettre l'emphase sur la structure réseau du problème). La contrainte (2.11) indique que le véhicule $v \in V$ ne peut pas arriver à j avant $b_i + t_{ij}$ s'il voyage de i à j. La contrainte (2.12) garantit que chaque client est desservi durant les heures d'ouvertures (fenêtre de temps) et la contrainte (2.13) représente l'ensemble des contraintes d'intégralité. Il est pertinent de mentionner que les clients ayant des demandes plus grandes qu'un seul véhicule peuvent être divisés en deux ou plusieurs clients identiques ayant les mêmes positions et fenêtres de temps. Il est clair que la façon de diviser les quantités demandées influence la solution. Dans ce modèle avec variables temporelles, il est possible de formuler le VRPTW sans contraintes d'éliminations de sous-tour.

Le VRPTW est NP-difficile puisqu'il peut être réduit à un VRP si la taille des fenêtres de temps est mise à l'infini et à un TSP (travelling salesman problem) si et seulement si un seul véhicule est alloué. Ce dernier problème a été démontré comme étant NP-difficile par Beame and al. [4].

L'ajout des fenêtres de temps ajoute une complexité supplémentaire au problème, au sens où une modification à un endroit de la tournée peut se répercuter sur les temps d'arrivée de tous les arcs subséquents, et de se fait rendre la tournée irréalisable. Contrairement à la contrainte de capacité, la faisabilité d'une tournée en temps dépend de l'ordonnancement des arcs.

En regardant les différents algorithmes qui ont été utilisés pour la résolution du VRPTW, nous remarquons une certaine évolution. Des heuristiques séquentielles aux métaheuristiques parallèles, il y a une panoplie de variantes intéressantes qui ont exploré différentes avenues de résolution. Nous retrouvons, en autre, des méthodes de résolution par constructions, améliorations, programmation par contraintes, programmation mathématique, combinaisons d'heuristiques et métaheuristiques tel les recherches avec tabous et algorithmes évolutifs.

2.4 Recherches coopératives appliquées à un problème de VRPTW

Cet article présente une méthode parallèle coopérative pour la résolution de problèmes combinatoires. Plusieurs métaheuristiques (recherches avec tabous et recherches génétiques) coopèrent de façon asynchrone en échangeant les meilleures solutions identifiées. Les échanges sont effectués via un entrepôt de données qui gère les solutions. Ce type d'échange permet de conserver l'indépendance des métaheuristiques tout en influençant leurs trajectoires de recherches. Les résultats présentés sur un problème NP-difficile, le VRPTW, montrent une accélération linéaire et permettent l'amélioration de la qualité des meilleurs résultats connus.

L'architecture coopérative générique sépare les mécanismes de communication et de résolution. Elle peut être alors utilisée pour la résolution de divers problèmes. La recherche coopérative est une métaheuristique possédant ses propres caractéristiques qui permet aux métaheuristiques impliquées l'obtention de meilleures trajectoires de recherches distinctes.

Nous nous inspirons du modèle de couche OSI pour situer notre modèle de recherche coopérative. La figure 2.3 illustre les couches sous-jacentes à notre cadre de recherche qui se situe au niveau de la septième couche. Les couches inférieures sont requises à la recherche coopérative. Dans la première et deuxième couches, nous retrouvons les composantes matérielles définissant le type d'unité centrale de traitement, ainsi que la topologie de communications utilisée. Notre méthode peut être utilisée sur une combinaison de systèmes SISD (Single Instruction Single Data) avec mémoire distribuée sous forme de grappe (cluster) ou sur un système hautement parallèle à mémoire partagée MIMD (Multiple Instruction Multiple Data). La topologie d'interconnexion physique des processeurs varie d'interconnexions complètes, dans le cas de machines hautement parallèles ou de systèmes en grappe, jusqu'à l'interconnexion de réseaux par large bande entre sous-réseaux, dans le cas d'une grille de calcul. La troisième couche, le système d'exploitation, définit les primitives de processus et de communications distribuées qui sont nécessaires au compilateur et aux librairies des niveaux supérieurs quatre et cinq. Ceci permet,

7	Architecture coopérative	Échange et traitement d'informations
6	Métaheuristiques	Entité calcul / communication
5	Librairie parallèle	Gestion des processus / communications
4	Compilateur	Compilation parallèle
3	Système d'exploitation	Primitive de calculs - communications
2	Réseau	Degré d'interconnexion des systèmes
1	Système informatique	Type de processeur / mémoire

Figure 2.3 – Couches sous-jacentes à la recherche coopérative

entre autres, la communication, la division des tâches et la gestion des processus. La sixième couche définit les métaheuristiques et autres méthodes de recherches utilisées. Finalement, la septième couche utilise la notion de recherche coopérative, processus autonome de décisions et de recherches, gérant des quantités d'informations échangées aux métaheuristiques. Elle encapsule les méthodes de contrôle et de gestion de l'information.

Les liens de communication entre métaheuristiques peuvent s'effectuer par un entrepôt de données de type *blackboard*, où les diverses métaheuristiques impliquées peuvent échanger leurs informations par une topologie en arbre ou en maillage irrégulier.

Bien que de nombreuses questions de recherche subsistent dans les couches inférieures, le présent travail porte sur la sixième et la septième couche qui représentent respectivement les métaheuristiques et l'architecture coopérative (c.f. Figure 2.3). Les relations avec les couches inférieures seront définies indépendamment de leur contenu.

2.4.1 Problématique

Les travaux antérieurs connus de la littérature ont permis de regarder et de comparer les méthodes de communication entre les diverses méthodes de résolution. On constate que la recherche coopérative influence les trajectoires de recherche des méthodes de résolution. Comme il a été souligné par Toulouse, Crainic et Gendreau, [105], le comportement coopératif ne respecte pas la logique d'optimisation des méthodes de résolution individuelles. Il devient donc très important que les mécanismes de contrôle des métaheuristiques de résolution imposent une logique d'optimisation globale sur la logique d'optimisation locale.

2.4.2 Méthodologie

Afin d'offrir une meilleure compréhension générale de la coopération, nous analysons son comportement en présence d'un nouveau problème NP-difficile, celui du VRPTW. Nous explorons aussi l'impact de la qualité des métaheuristiques individuelles sur la performance globale du système.

L'architecture coopérative a été conçue pour être indépendante des couches 1 à 5, à savoir indépendante du matériel, des communications, de sa topologie, du système d'exploitation, ainsi que des compilateurs et des librairies parallèles. La couche sept, que constituent les mécanismes de gestion et de coopération, reste aussi générique à tout problème combinatoire. Seule la couche six, qui implante des méthodes de recherches adaptées, doit être remplacée selon le problème résolu. L'architecture coopérative est donc portable et théoriquement indépendante du problème.

Dans le cadre expérimental, nous avons ajouté à la couche six une méthode de post-optimisation basée sur les chaînes d'éjections [17]. Nous avons aussi remplacé une des deux méthodes Taburoute [49] par une méthode de recherche avec tabous unifiés [23].

La publication est structurée de la façon suivante :

1. La motivation de la recherche et l'identification des contributions scientifiques

proposées ;

2. Le problème combinatoire choisi : le VRPTW, ainsi que les méthodes de résolutions habituellement utilisées. Nous discutons des recherches avec tabous, des algorithmes génétiques, des recuits simulés, des LNS et VNS, de différents mécanismes de post-optimisation, de croisements et de la mémoire adaptative ;
3. Le concept de recherche coopérative appliqué au VRPTW, le design général ainsi que les méthodes coopératives utilisées : Taburoute, le tabou unifié ainsi que deux algorithmes génétiques basés sur des opérateurs de croisements simples OX et ER. Nous présentons aussi une variation des chaînes d'éjections, des méthodes de post-optimisation 2-opt, 3-opt, OR-opt ainsi que des méthodes de génération de solutions initiales adaptées pour le VRPTW ;
4. La partie expérimentale : Cette section nous permet de comparer la recherche coopérative appliquée au VRPTW aux meilleures méthodes figurant dans la littérature et d'évaluer le rôle de la coopération ainsi que les contributions des méthodes individuelles à la performance de la recherche globale.

2.5 Permission de l'éditeur

3 December 2003

Our Ref: HW/jj/Dec03/J046
Your Ref:

Alexandre Le Bouthillier
Via email alexleb@crt.umontreal.ca

Dear Alexandre Le Bouthillier

COMPUTERS AND OPERATION RESEARCH, (in press), Bouthillier: "A cooperative parallel meta-heuristic for..."

As per your letter dated 28 November 2003, we hereby grant you permission to reprint the aforementioned material at no charge **in your thesis** subject to the following conditions:

1. If any part of the material to be used (for example, figures) has appeared in our publication with credit or acknowledgement to another source, permission must also be sought from that source. If such permission is not obtained then that material may not be included in your publication/copies.

2. Suitable acknowledgment to the source must be made, either as a footnote or in a reference list at the end of your publication, as follows: Repri nted from Publication title, Vol number, Author(s), Title of article, Pages No., Copyright (Year), with permission from Elsevier .

3. Reproduction of this material is confined to the purpose for which permission is hereby given.

4. This permission is granted for non-exclusive world **English** rights only. For other languages please reapply separately for each one required. Permission excludes use in an electronic form. Should you have a specific electronic project in mind please reapply for permission.

5. This includes permission for the National Library of Canada to supply single copies, on demand, of the complete thesis. Should your thesis be published commercially, please reapply for permission.

Yours sincerely

Helen Wilson
Rights Manager

Your future requests will be handled more quickly if you complete the online form at
www.elsevier.com/locate/permissions

Figure 2.4 – Permission de l'éditeur

2.6 Publication 1: "A cooperative parallel meta-heuristic for the vehicle routing problem with time windows"

Alexandre Le Bouthillier
Département d'informatique et de recherche opérationnelle and
Centre de recherche sur les transports
Université de Montréal
C.P.6128, succursale Centre-ville, Montréal, QC, CANADA H3C 3J7
alexleb@crt.umontreal.ca

Teodor Gabriel Crainic
Département de management et technologie
Université du Québec à Montréal
C.P. 8888, Succursale Centre-Ville, Montréal, QC, CANADA H3C 3P8 and
Centre de recherche sur les transports,
Université de Montréal
C.P.6128, succursale Centre-ville, Montréal, QC, CANADA H3C 3J7
theo@crt.umontreal.ca

Reprinted from Computers & Operations Research, Vol 32, Le Bouthillier, A., Crainic, T.G., "A Cooperative Parallel Meta-Heuristic for the Vehicle Routing Problem with Time Windows", 1685-1708 ©(2005), with permission from Elsevier.

2.7 Abstract

This paper presents a parallel cooperative multi-search method for the vehicle routing problem with time windows. It is based on the solution warehouse strategy, in which several search threads cooperate by asynchronously exchanging information on the best solutions identified. The exchanges are performed through a mechanism, called solution warehouse, which holds and manages a pool of solutions. This enforces the asynchronous strategy of information exchanges and ensures the independence of the individual search processes. Each of these independent processes implements a different meta-heuristic, an evolutionary algorithm or a tabu search procedure. No attempt has been made to calibrate the individual procedures or the parallel cooperative method. The results obtained on an extended set of test problems show that the parallel procedure achieves linear accelerations and identifies solutions of comparable quality to those obtained by the best methods in the literature.

Keywords : vehicle routing with time windows, parallel meta-heuristics, cooperative search, solution warehouse strategy

2.8 Introduction

Parallel resolution methods offer the possibility of accelerating computations and, as such, these methods constitute an interesting field of research for combinatorial optimization. However, classical parallel approaches, based on functional or data decomposition, do not significantly modify the search trajectories of meta-heuristics. Thus, they cannot improve the quality of the solution, nor do they enhance the robustness of the search when faced with different problem instances than those which were originally calibrated and applied. Consequently, in recent years, *multi-search* (or *multi-thread*) *meta-heuristics*, with varying degrees of *cooperation*, have increasingly been used for difficult combinatorial problems and have been shown to both speed up the search and dramatically improve the robustness and the quality of the solutions obtained (e.g., Crainic [24], Crainic and Gendreau [25, 26], Crainic and Toulouse [28, 29], Crainic, Toulouse, and Gendreau [30], Cung *et al.* [32], Garcia, Potvin, and Rousseau [45], Gambarella, Taillard, and Agazzi [44], Homberger and Gehring [64], Gehring and Homberger [47], Schulze and Fahle [90], Taillard *et al.* [100], Taillard [98], Toulouse *et al.* [106], etc.).

The *vehicle routing problem with time windows (VRPTW)* is one of the central problems in operations research and combinatorial optimization with numerous practical applications (Cordeau *et al.* [21]). The problem is NP-Hard and thus, not surprisingly, classical and modern heuristics have been extensively studied, with meta-heuristics generally offering the best performances (e.g., Bräysy and Gendreau [14], Cordeau *et al.* [21, 22], Christofides, Mingozzi, and Toth [19], Desrosiers *et al.* [36], Desaulniers *et al.* [35], Fisher [42], Gendreau, Laporte, and Potvin [51], Golden and Assad [57], Golden *et al.* [58], Laporte [71], Laporte and Semet [73], Laporte *et al.* [72], Toth and Vigo [101], [102] and references therein).

The parallel cooperative multi-search method based on the *solution warehouse* strategy has been successfully applied to a number of difficult combinatorial problems (e.g., Crainic, Toulouse, and Gendreau [30], Crainic and Gendreau [26], Aiex *et al.* [2]) but not, at least to the authors' knowledge, to the VRPTW. This par-

allel approach, based on several search threads that cooperate by asynchronously exchanging information about the best solutions identified so far, offers several advantages: simplicity of design, increased search robustness, the possibility to enhance performance through enhancement of some of the individual search threads. The objective of this paper is to explore the applicability of the solution warehouse-based cooperative multi-search method to VRPTW and determine its competitiveness compared to the current state-of-the-art.

The contributions of this paper are as follows. We present an easy-to-build and efficient parallel solution method for the single-depot version of the VRPTW. The method makes use of existing search methods without any particular calibration for the parallel cooperative method or the individual procedures (we use the parameter settings proposed for the sequential methods by the original authors). Experiments using both the standard benchmark test problems of Solomon [93] and the more recent one proposed by Homberger and Gehring [64] show that the parallel procedure achieves solutions of quality comparable to that obtained by the best methods of the literature. The proposed cooperative search identifies several new best solutions and finds new best cumulative numbers of vehicles for several problems ranging from 200 to 1000 customers. From a parallel computation point of view, the proposed method is efficient as it displays linear accelerations compared to the sequential methods. The paper also contributes to the general understanding of cooperative parallel methods by analyzing its behavior when applied to a new problem class and by exploring the impact of the quality of the individual search threads on the performance of the parallel search.

The paper is organized as follows. Section 2.9 briefly reviews the main sequential and parallel methods proposed to solve the VRPTW. Section 2.10 presents the cooperative search procedure, its general design and the methods selected for cooperation. Section 2.11 presents the computational results and analyzes them both from the point of view of solving the VRPTW and from that of the performance of the parallel strategy. Conclusions and perspectives are the subject of the last section.

2.9 The VRPTW and Main Solution Methods

In this paper, we address the single depot VRPTW. One is given a set of customers with known positive demands and specific time intervals when service can be provided. A fleet of homogeneous vehicles of known capacity is available at a given depot to perform this service. The objective is to find a set of closed routes (or tours) that start and end at the depot within its opening hours, such that the total cost of performing the service is minimized, customers are visited and served during the specified time windows, and vehicles are not overloaded. In the problem version we address, cost is a combination of two factors: the number of vehicles (routes) used and the total distance traveled. A high cost is associated with vehicle utilization to enforce the search towards solutions with a reduced number of vehicles. Each customer is visited only once. A vehicle cannot arrive later than the customer's closing time, but is allowed to arrive before the associated opening time, in which case it waits, without explicit penalty, until the customer is ready. Once the service starts, it is carried on until completion, even if the service ending time might be later than the expiration of the time window. The recent review by Cordeau *et al.* [21], as well as the other reviews indicated in the Introduction, present mathematical formulations, variants, and solution methods for the problem.

Presenting a comprehensive review of solution methods for the VRPTW is beyond the scope of this paper, especially since several reviews have appeared recently. In the following, we briefly review a number of "successful" sequential and parallel solution approaches that obtained best-known results for the classical Solomon problem set and on the 200-1000-customer problems. Methods that are incorporated in the proposed cooperative search are presented in somewhat more details in Section 2.10.2.

Efforts are being dedicated to the development of exact algorithms and most 100-customer VRPTW problems with a distance-minimization objective have been solved (Chabrier [18], Cook and Rich [20], Kallehauge, Larsen, and Madsen [68], Larsen [74], Kohl *et al.* [70]). For most problem instances, heuristics and meta-

heuristics are required, however. Most such efforts have addressed the combined objective of minimizing first the number of vehicles and then the total distance.

Tabu search (Glover [53], [54], Glover and Laguna [56]) has often been used successfully to address vehicle routing problems including the VRPTW. *Taburoute* (Gendreau, Hertz, and Laporte [49]) and the *Unified Tabu Search* (Cordeau, Laporte, and Mercier [23]) stand out by the quality of the results. These two methods are used in the proposed framework and are reviewed in Section 2.10.2.

Hybrids that combine elements of different methodologies appear very promising. Rousseau, Gendreau, and Pesant [88] combined constraint programming and variable neighborhood search (VNS; Mladenović and Hansen [83], Hansen and Mladenović [60], [61]). The pruning operators and propagation techniques of constraint programming permit the search of large neighborhoods. This increases the probability of finding a good solution at each iteration and avoids the need for specialized neighborhood descent procedures. The utilization of constraint programming also allows additional constraints to be added easily without fundamentally modifying the methodology.

Bräysy [12] proposed the *reactive variable neighborhood search*, a variation on the VNS based on a four-phase method: initialization, route elimination, total distance minimization by using four new local search heuristics, and a modification of the objective function to allow escaping from local minima. Bräysy et al. [11] proposed a multi-start local search method that proceeds in two phases: 1) production of initial solutions using a fast construction heuristic and an ejection chain-based heuristic to reduce the number of total routes; 2) two improvement heuristics based on CROSS-Exchanges (Taillard *et al.* [99]) to reduce the total distance. Bräysy and Dullaert [13] proposed the *fast evolutionary metaheuristic*, an hybrid procedure based on evolutionary principles.

Bent and Van Hentenryck [5] presented a two phase method that combines simulated annealing (SA) and large neighborhood search (LNS). The first phase (SA) utilizes a lexicographic evaluation function to minimize the number of routes and the delay of the entire solution. The SA explores the neighborhood with tradi-

tional move operators: 2-exchange, Or-exchange, and crossover. It also includes some components often seen in tabu search, such as an aspiration criteria and a bias towards good solution in the random selection process. The second phase (LNS, Shaw [91]) aims to minimize the total travel cost by using a branch and bound algorithm.

Many of the most successful meta-heuristics for the large VRPTW instances are based on some form of parallel computation. Two methodological approaches stand out: the *adaptive memory* strategy (Rochat and Taillard [87], Taillard *et al.* [99, 100], Badeau *et al.* [3]) and the combination of tabu search and evolutionary algorithms proposed by Gehring and Homberger [47], Homberger and Gehring [64].

Rochat and Taillard [87] proposed what may be considered as the first fully developed adaptive memory-based approach for the VRPTW. It is a cooperative multi-thread method, where the adaptive memory contains tours of good solutions identified by the tabu search process. The tours are ranked according to attribute values, including the objective values of their respective solutions. Each tabu search process then probabilistically selects tours in the memory, constructs an initial solution, improves it, and returns the corresponding tours to the adaptive memory. Despite the fact that it used a rather simple tabu search, this method produced many new best results at publication time. Taillard *et al.* [99] adapted this method for the vehicle routing problem with soft time windows. Late arrival at a customer is allowed at a penalty. By adjusting the penalty, hard time window cases may also be addressed. The authors significantly refined the tabu search by enriching the neighborhood and the intensification phase and by adding a post-optimization procedure. Badeau *et al.* [3] report the first "true" parallel implementation of this approach for the VRPTW with soft time windows. Another variant on the adaptive-memory idea may be found in the work of Schulze and Fahle [90]. Here, the pool of partial solutions is distributed among processes to eliminate the need for a "master". The elements of the best solutions found by each thread are broadcast to ensure that each search has still access to all the information when building new solutions. Implemented on a limited number of processors (eight), the method

performed well.

Gehring and Homberger [47] introduced a cooperative parallel strategy where concurrent searches are performed with differently configured two-phase meta-heuristics. The first phase tries to minimize the number of vehicles by using an evolutionary meta-heuristic, while the second phase aims to minimize the total traveled distance by means of a tabu search. The evolution strategies are based on the previous work by Homberger and Gehring [64] and Homberger [63] with much emphasis on the minimization of the number of vehicles in the evolution strategy. The parallel meta-heuristic is initiated on different threads with different starting points and values for the time allocated to each search phase. Threads cooperate by exchanging solutions asynchronously through a master process. For the first time, results were also presented on new and larger problem instances, generated similarly to the original Solomon instances, but varying in size from 200 to 1000 customers.

Berger *et al.* [7] introduced a parallel genetic method where two populations co-evolve, each with a distinct objective. The first population aims to minimize the total distance, while the second focuses on minimizing the temporal constraints violation to generate feasible solutions. New feasible solutions are exchanged between populations. New genetic operators inspired by an insertion heuristic, a large neighborhood search method and an ant colony system are also used.

To conclude, it is worth noticing that while several solution methods have successfully addressed several classes of VRPTW problem instances, none can claim to dominate all the others. On the other hand, in general, the most successful methods combine several methodologies. The method we propose is also based on combining the search efforts of different methods, but it follows a principle not found among the methods reviewed that offers a general framework for algorithmic development for the VRPTW.

2.10 Cooperative Meta-heuristics for the VRPTW

The multi-thread cooperative parallel search procedure we propose for the VRPTW is based on the solution warehouse approach. This section describes the general principles and main components of this solution strategy.

2.10.1 General Design

A number of independent processes (threads) are defined and each executes a complete search through the solution space of the VRPTW (one of them may, as in the present implementation construct and improve solutions based on classical heuristics). When the same meta-heuristic is used by several search threads, the initial solution and particular setting of a number of important search parameters differentiates each search thread from the others. Thus, each thread follows a different *search strategy*. Full implementation details for the strategies used in the present method are given in the following sub-sections. Both an *independent search* method and a *cooperative search* algorithm are implemented. In the former, all searches proceed independently and the best solution is collected at the end. In the latter, it is hoped that by exchanging information among the search threads the efficiency of the global search will increase to yield a higher-quality final solution.

The cooperation aspect of the parallelization scheme is achieved through asynchronous exchanges of information. Information is shared through a *solution warehouse* or *pool of solutions*. In this scheme, whenever a thread desires to send out information, it sends it to the pool. Similarly, when a thread accesses outside information, it reaches out and takes it from the pool. Communications are initiated exclusively by the individual threads, irrespective of their role as senders or receivers of information. No broadcasting is taking place and there is no need for complex mechanisms to select the threads that will receive or send information and to control the cooperation. The solution warehouse is thus an efficient implementation device that allows for a strict asynchronous mode of exchange, with no pre-determined connection pattern, where no process is interrupted by another for

communication purposes, but where any thread may access at all times the data previously sent out by any other search thread. The solution warehouse keeps the information in an order appropriate for the exchange mechanism considered.

To fully characterize the cooperation process, one has to specify *(i)* the information which is to be shared; *(ii)* the particular methods that make up the cooperative search; *(iii)* the time when communications occur; *(iv)* the utilization each thread makes of the imported information (Toulouse, Crainic, and Gendreau [105], Crainic and Toulouse [28, 29], Crainic [24]).

The information exchanged among cooperating procedures has to be meaningful, in the sense that it has to be useful for the decision process of the receiving threads. Information that gives the current status of the global search or, at least, of some other searches is, in this sense, meaningful. In the implementations described in this paper, threads share information about their respective good solutions identified so far. When a thread improves the imported solution or when it identifies a new best solution, it sends out the solution. This scheme is intuitive and simple, and it satisfies the meaningfulness requirement.

The selection of the methods involved in cooperation was mainly oriented toward obtaining: *(i)* Good quality solutions; *(ii)* A broad diversity of solutions to facilitate the discovery of promising regions; *(iii)* The rapid production of intermediate solutions to feed the information exchange mechanisms. Tabu search methods and, recently, evolutionary algorithms have provided some of the best solutions found so far in the literature. Moreover, evolutionary algorithms may contribute toward increasing the diversity of solutions exchanged among cooperating methods. It was thus decided to include tabu search and evolutionary algorithms in the cooperative framework.

Taburoute (Gendreau, Hertz, and Laporte [49]) and unified tabu search (Cordeau, Laporte, and Mercier [23]) were the two tabu search-based methods considered. Two evolutionary algorithms were also included with simple but different, crossover mechanisms. Construction and improvement heuristics were also included to generate an initial population (pool of solutions) and to perform post-optimization.

These methods are described in the following sub-sections, together with the corresponding mechanisms for exchanging information with the solution warehouse and for processing this data.

2.10.2 The Cooperating Methods

2.10.2.1 Taburoute

The Taburoute tabu search method (Gendreau, Hertz, and Laporte [49]) was originally proposed for the vehicle routing problem with capacities and route length restrictions. In this algorithm, the neighborhood of a solution is defined by considering a sequence of adjacent solutions obtained by repeatedly removing a node from its current route and reinserting it into another route that contains one of its p nearest neighbors. Re-insertion is done by means of the *generalized insertion (GENI)* procedure. The post-optimization procedure *unstringing-stringing (US)* attempts to improve the solution by trying to remove and reinsert each node of a route. The two procedures make up the *GENIUS* heuristic (Gendreau, Hertz, and Laporte [48], Gendreau *et al.* [50]). When a node is moved, it is tagged as "Tabu", to prevent it from moving back to its original route, for a number of iterations uniformly drawn from a $[inf, sup]$ interval. Moving to infeasible solutions is allowed during the course of the algorithm, in order to decrease the likelihood of getting trapped in local optima, but a penalty term in the objective function is increased when solutions obtained in the last 10 iterations are infeasible. The diversification strategy penalizes vertices that have been moved frequently in order to avoid considering similar solutions. A false start is performed at the beginning by performing a local search on a (usually random) solution to find a good initialization.

In the time window versions of Taburoute and GENIUS developed for the cooperative parallel method, insertion is only allowed when no time window constraints are violated. The method will send improving solutions to the central warehouse and require solutions from the warehouse before diversification. This imported solution will be accepted if it's better than the current one. Otherwise, the algorithm

will proceed from its own solution.

2.10.2.2 Unified tabu

The Unified tabu Search heuristic proposed by Cordeau, Laporte, and Mercier [23] is one of the best tabu search-based methods for the VRPTW. The unified tabu also solves two important generalizations of the VRPTW: the periodic and the multi-depot vehicle routing problems with time windows. The major benefits of the approach are its speed, simplicity, flexibility, as well as the quality of the produced results.

Unified tabu uses several of the features of Taburoute, namely the same neighborhood structure, GENI insertions, long-term frequency-based penalties. It also differs from Taburoute in a number of ways. The search is applied to a single initial solution, fixed tabu durations are used and no intensification phase is included. The methods allows intermediate infeasible solutions (similarly to Taburoute), but modifies its search parameters at each iteration according to whether the previous solution was feasible or not with respect to capacity or route duration. The tabu mechanism operates on an attribute set :

$$B(x) = \{(i,k) : \text{ customer } v_i \text{ is visited by vehicle } k \text{ in solution } x.\} \qquad (2.14)$$

Neighbor solutions are obtained by removing (i,k) from $B(x)$ and replacing it with (i,k'), where $k' \neq k$. Attribute (i,k) is then declared tabu for a number of iterations, and the aspiration criterion is defined relative to attribute (i,k). At the end of n iterations, a GENIUS-based (Gendreau *et al.* [50]) post-optimization is applied to each route of the best feasible solution found.

To integrate this method into a cooperative mechanism, one has to create moments when communications are possible (this step is required since neither intensification nor diversification are in the original design). We decided to stop the method after w iterations and import a solution from the central warehouse. This solution is then used to re-start the search. On the other hand, improving feasible

solutions are sent to the warehouse.

2.10.2.3 Evolutionary Algorithms

Two evolutionary algorithms also participate in the cooperative global search. Both methods use the solution warehouse as population. Two parent solutions are chosen randomly and a probabilistic mutation is performed on a copy of each parent (for each arc, a 1% probability to replace it with a randomly-chosen one). The algorithms differ by the crossover operator used, either an *order* crossover *(OX)* or an *edge recombination (ER)* crossover. When required, a repair procedure restores the feasibility of the solution by re-ordering or re-routing nodes reached after their closing time windows. The offspring is sent to the solution warehouse.

The OX operator is based on a path representation on each route and attempts to preserve the relative order of the customers in the second parent. It copies a sub-chain of the first parent to the offspring; the remaining positions are filled by customers that are no yet in the offspring, in the order that they appear in the second parent. The ER crossover operator aims to preserve the maximum number of arcs from the parent and to introduce a minimum number of new arcs. An adjacency table is constructed to represent, for each node, the nodes that are adjacent to it in each parent. The construction of the offspring is performed by selecting paths with a minimum number of arcs between two adjacent nodes that are not yet in the offspring. A back-track scheme is performed when nodes are left with no active arcs.

2.10.2.4 Post-Optimization

The *ejection chain* principle is mainly used to explore large and complex neighborhoods in tabu Search and corresponds in general to chained-movements of solution elements among several solutions. Two types of Ejection Chains are used in the VRP literature (Glover [52], Glover [55], Rego [85], Rego [86], Rousseau, Gendreau, and Pesant [88], Caseau and Laburthe [17]). The first type is called

a multi-node exchange process and involves only one route at a time. All node exchanges are performed within the same route and the last node of the chain takes the position of the first node. The second type is called a multi-node insert process and involves several routes with exchanges being performed on two routes at a time. A node is ejected from the first route and inserted into a second one, from which another node is eventually ejected to be inserted into another route, and so on and so forth until a certain number of exchanges is attained.

The proposed Ejection Search procedure is based on the multi-node process and tries to empty a route, called the origin route, by sequentially ejecting its nodes and inserting them into other routes. An ejected node does not, however, replace an existing node. Rather, it is simply inserted in a destination route between two existing nodes. If the insertion of this node causes the receiving route to become infeasible, other nodes of this route are ejected until feasibility is regained. This process is repeated for subsequent destination routes until the maximum number of exchanges is attained, at which point the process backtracks to choose alternative destination routes if the route-feasibility criterion is still not met.

The origin route is chosen according to the shortest tour in distance, the tour with the smallest number of customers, the tour with the highest residual capacity, or the tour with the smallest sum of distance of each customer to its closest neighbor. The nodes in the chosen origin tour are ejected one at a time to maximize the total saving in distance (the saving in the origin tour minus the added detour in the destination tour) and the slack in the available capacity of the destination route.

The proposed ejection-chain search and classical 2-Opt, 3-Opt, Or-Opt heuristics (Kindervater and Savelsbergh [69], De Backer *et al.* [33]) make up the post-optimization procedure. They are executed one after the other (ejection search followed by 2-opt, 3-opt and finally Or-opt) to improve the solutions received by the solution warehouse as described in Section 2.10.3.

2.10.2.5 Initial Solutions

A number of classical heuristics have also been included to generate initial solutions and help diversify the pool. Four simple construction heuristics from Bentley [6] were adapted for the VRPTW: *(i)* least successor; *(ii)* double-ended nearest neighbor, which starts from a seed node for each tour and sequentially adds the nearest feasible node on each side of the current tour until the tour is full; *(iii)* multiple fragments, which sequentially adds the shortest arcs to form feasible tours; and *(iv)* shortest arcs hybridizing, which is similar to the previous heuristic but adds arcs probabilistically according to their length (shorter length arcs have a greater probability of being chosen). When required, the repair procedure is applied to the resulting solutions.

2.10.3 The Cooperation Mechanism

The solution warehouse is the core of the cooperation mechanism. It keeps good, feasible solutions and is dynamically updated by the independent search processes. The pool of solutions forms therefore an elite population from which the independent procedures will require solutions at various stages of execution. Path representation is used to keep solutions in solution warehouse. These solutions are ordered first by the number of vehicles (in increasing order) and then according to the weighted sum of the corresponding travel time, total distance, waiting time and residual time for all customers (see Section 2.11).

The solution warehouse is divided into two sub-populations: *in-training* and *adult*. All solutions received from the independent processes are placed in the in-training part. The post-optimization procedure is then applied and the resulting solution is moved to the adult sub-population. Duplicate solutions are eliminated. All requests for solutions initiated by the independent processes are satisfied by the adult sub-population. Solutions are selected randomly according to probabilities biased toward the best based on the same function used to order solutions in the solution warehouse.

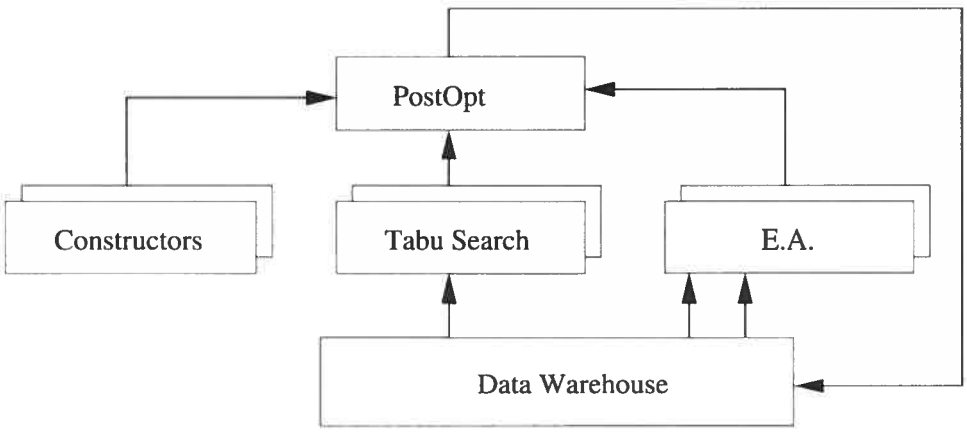

Figure 2.5: Collaboration diagram

In the collaboration diagram depicted in Figure 2.5, each independent method (the construction heuristics are grouped together) is encapsulated in an MPI process. An MPI process contains the particular algorithm, a communication mechanism for exchanging solutions, and a control mechanism for parameter setting, initialization, termination, and solution acceptance. The cooperative method works through the solution warehouse, which is the central point of communication. It provides starting and diversification solutions to tabu search procedures and parents to the evolutionary algorithms (E.A). The population size in the solution warehouse is fixed to a value related to the problem size (10*size) and the worst results are eliminated as needed. No direct communications take place between processes thus enforcing their independence and the asynchronous mode of exchange. This scheme makes the cooperation design simpler and, eventually, allows easy modification of the parallel system by adding new methods or dropping inefficient ones.

2.11 Computational Experiments

The experimental study had a dual objective: *(i)* to compare the proposed warehouse cooperative parallel meta-heuristic to the best performing methods proposed in the literature for the VRPTW and, thus, to validate our claim that the proposed method offers comparable performance in terms of both solution quality and

computation effort; *(ii)* to evaluate the role of cooperation and the contribution of the individual methods involved in the cooperation, in particular the impact of the quality of individual methods on the performance of the global search. The ultimate objective of the cooperative search is to show that by combining off-the-shelf methods robustness can be attain with linear speedup and no calibration.

We start by indicating the problem data sets used, the settings of the individual and global searches, and the computing environment. Results are then presented, compared, and analyzed. Finally, we present a brief analysis of the behavior of the parallel cooperative search.

2.11.1 Experimental setting

A different search method is run on each of four processors (two for tabu search and two for evolutionary algorithm). The solution warehouse, the post-optimization, and the construction methods are run on another processor for a total of five processors in this study.

Experimentations were performed on two versions of the cooperative search to study the influence of the individual methods on the global performance and the interaction between the methods when used in cooperation. The first experimentation (identified **LC02** in the following) consists of two Taburoute methods (**TS1** and **TS2**) and the two evolutionary algorithms (**OX** and **ER**). In the second experiment (**LC03**), the unified tabu (**UT1**) replaces the second Taburoute method. The evolutionary algorithms do not change.

Two sets of parameters were defined for the Taburoute search threads based on those indicated in the original paper for the VRP (Gendreau, Hertz, and Laporte [49]. Both sets used a tabu tag length of [5,10] iterations. Each set used a different seed as well as different p-neighborhood dimensions (15% and 20% of the nodes are evaluated in the first and second Tabu search, respectively), initial solutions (the best solution out of 15 and 20 false starts), and penalty factors for frequently moved vertices (1 and 0.5). The parameters for the value function presented in the initial article by Cordeau, Laporte, and Mercier [23] : $\alpha = 1, \beta = 1, \gamma = 1$) were

used for the unified tabu search thread. Finally, an arc mutation probability of 1/100 was used on temporary copies of the parents for the crossovers used by the evolutionary algorithms.

Tests have been carried out on the standard set of test problems proposed by Solomon [93] and used by all authors. The set contains 56 problems of 100 customers each. We also used the extended set produced by Homberger and Gehrin [64] with 300 problem instances that vary from 200 to 1000 customers.

The Solomon standard and extended problems are divided into six categories, named C1, C2, R1, R2, RC1, and RC2. For all problem instances, customers are distributed in a [0,100] square unit. The customers in sets C are clustered together, while those in sets R are distributed randomly. Problems in sets RC combine the two characteristics. Time windows at the depot are relatively small for problems of type 1, to allow less customers to be served by each route; time windows are larger for problems of type 2. The service time is of 10 units by customer for problems of type R and RC, and of 90 units for class C.

Solutions in the adult sub-population of the solution warehouse are sorted, first by the number of vehicles, second by a weighted sum, $C(p)$, of attributes: the total time required to serve all customers, the associated total distance and total waiting time at customers, and the sum of the slack left in each time window:

$$C(p) = W_1 * totalTime + W_2 * totalDistance + W_3 * totalWait + W_4 * totalSlack$$

W_1 through W_4 where set to 1 in all the reported experiments. This measure combined with the number of vehicles gives us an overall idea of the solution quality (*totalTime* and *totalDistance*) and flexibility (*totalWait* and *totalSlack*). The two last measures indicate how much slack there exists in the solution and how easily feasible neighboring solutions may be explored.

Runs of 12 minutes wall clock time are performed by the cooperative meta-heuristic for each of the 100 city problems, while 50 minutes are given to the sequential runs of the independent algorithms. Longer running times, equal to

those reported by Homberger and Gehring [64] were allowed for the larger problem instances. These times go up to 50 minutes wall clock time for the 1000 city problem.

Clusters of five Pentium III computers were used for both versions of the cooperative search. The first version (**LC02**) was run on 500MHz CPUs with 128MB of RAM under Windows. The second version (**LC03**) was run on 850MHz CPUs with 512MB of RAM under Linux. Communications were managed by Voyager and MPICH for version **LC02** and **LC03**, respectively. Computations of distances have been carried out in double precision. The second implementation (**LC03**) is machine independent and can be run with MPICH on Unix, Windows, or Linux.

2.11.2 Computational results

Average results for each set of problem instances are given in this section. Complete results are presented in the Appendix for **LC03** and in Le Bouthillier and Crainic [76] for **LC02**. Tables display average values for the total number of vehicles and the total distance (just below the number of vehicles) for each problem class and, globally, for each meta-heuristic. Best results are indicated in bold characters. Tables 2.4 to 2.11 also display the cumulative number of vehicles (CNV) and cumulative total distance (CTD) for each procedure. Unless otherwise indicated, results were obtained on the classical Solomon data set with 100 customers.

First, a comparison is made between the sequential methods involved in the cooperative search. Results are displayed in Table 2.2. The five meta-heuristics equally solve the problems of type C that have clustered customers. This is not surprising since it is easy to assign distant customers to distinct routes which belong to the optimal solution. Problems with randomly positioned customers and large time windows are more difficult to solve. There is no clear winner, although the unified tabu search slightly outperforms our version of Taburoute and the evolutionary algorithms.

Table 2.3 displays results for the cooperative (**LC02 and LC03**) and independent (**LCIND**) parallel methods using **TSI**, **UT**, **OX** and **ER**. In the independent

Class	**UT1**	**TS1**	**TS2**	**ER**	**OX**
R1	**12.08**	12.17	12.17	12.25	12.49
	1210.14	1211.10	1215.45	1211.38	1218.79
C1	**10.00**	**10.00**	**10.00**	**10.00**	**10.00**
	828.38	828.38	828.38	828.38	828.38
RC1	11.50	**11.50**	11.67	11.88	11.74
	1389.78	1388.24	1384.65	1394.28	1380.29
R2	**2.73**	2.94	2.96	2.94	3.02
	969.57	955.18	958.95	966.78	954.32
C2	**3.00**	**3.00**	**3.00**	**3.00**	**3.00**
	589.86	**589.86**	**589.86**	**589.86**	**589.86**
RC2	**3.25**	3.25	3.25	3.38	3.31
	1134.52	1145.29	1147.00	1118.63	1144.13

Table 2.2: Internal Comparisons

runs, there is no communication between the individual searches and the solution warehouse serves only to report the best solution of all the four methods (UT1, TS1, ER, OX). It is worth recalling that each independent method runs for 50 minutes and the parallel method runs for 12 minutes for each of the five processors (one for each method and one for the solution warehouse). There is a speedup of 1.2 when comparing the total clock time of each cooperative method (**LC02 and LC03**) to the sequential run (**LCIND**) as seen in figure 2.7 and 2.8. In both cases, the best solution is found in less than 600 seconds for both cooperative methods and in 3600 seconds for sequential methods.

The results indicate that cooperation increases the robustness of the search, without an increase in computation cost. The method **LCIND** is slightly better than **LC02** due to the use of a better tabu search (**UT**). The cooperative search is able to produce better results than the best independent method. Furthermore, even accounting for a significant speedup, the cumulative number of vehicles is reduced by two for **LC03** when compared to the independent method. With only one fifth of the time available when running in sequential mode, individual methods benefit from the cooperation and are thus able to obtain the same or better solution quality than in sequential (with only one exception out of 356 problems –

R205 – with 0.49% more distance for **LC03**). On average the cooperative method shows better total distance for each class of problems compared to the independent approach.

To assess the performance of the cooperative parallel meta-heuristic, we compared the results of the two variants (**LC02** and (**LC03**) to those of the best methods (published or not) for the VRPTW, reviewed in Section 2.9: Rochat and Taillard ([87], denoted **RT**), Homberger and Gehring ([64], denoted **GH99**), Taillard *et al.* ([99], denoted **TB**), Rousseau, Gendreau, and Pesant ([88], denoted **RGP**), Cordeau, Laporte and Mercier ([23], denoted **CLM**), Bräysy, Hasle, and Dullaert ([11] - **BHD**), Homberger ([63], denoted **H99**), Bräysy and Dullaert ([13], denoted **EA2**), Gehring and Homberger (2001[47], denoted **GH01**), Bräysy ([12], denoted **B01**), and Bent and Hentenryck ([5], denoted **BVH**). Table 2.4 displays the results on the standard 100-customer problem sets (the only set addressed by all). The table also presents the CNV, the CTD for each method, and the experiment settings (CPU time, number of runs and minutes for each run).

The performance of the two versions of the proposed cooperative are competitive even if no particular calibration has been performed. The solution quality is comparable to the best of the sequential and parallel methods (**LC03** is in 7th position with regard to the CNV), while computation times are reasonable, even when we take into account the five processors used.

We also compared our results to those in the literature on the extended data sets provided by Homberger and Gehring [64]. Table 2.11 shows the total CNV and CTD for all 300 problems of the extended set. Average results by problem size and class can be found in Tables 2.6 to Tables 2.10. Detailed results by problem and class can be found in Tables 2.14 to Tables 2.18 in the Annex. For each problem size and class, each table displays the CNV, the CTD, and the CPU time for each method.

Results are again very satisfying. The quality of results and the computation effort are on a par with the best methods available. Five new best solutions are found and shown in Table 2.5.

Class	**LCIND**	**LC02**	**LC03**
R1	12.08	12.17	**12.08**
	1210.14	1209.27	1209.19
C1	**10.00**	**10.00**	**10.00**
	828.38	828.38	828.38
RC1	11.50	11.50	**11.50**
	1388.24	1389.22	1386.38
R2	2.73	2.82	**2.73**
	969.57	965.91	960.95
C2	**3.00**	**3.00**	**3.00**
	589.86	589.86	589.86
RC2	3.25	3.25	**3.25**
	1134.52	1143.70	1133.30

Table 2.3: Cooperative versus Independent Methods

Author	R1	R2	C1	C2	RC1	RC2	CNV/CTD	Experiment
RT	12.25	2.91	10.00	3.00	11.88	3.38	415	SG. 100MHz
	1 208.50	961.72	828.38	589.86	1 377.39	1 117.44	57 231	1 run, 92.2 Min
GH99	12.42	2.82	10.00	3.00	11.88	3.25	415	4 P200MHz,
	1 198	947	829	590	1 356	1 144	56 946	1 run, 5 Min
TB	12.17	2.82	10.00	3.00	11.50	3.38	410	Sun 10
	1 209.27	980.27	828.385	589.86	1 388.24	1 140.42	57 521	1 run, 248 Min
LC02	12.17	2.82	10.00	3.00	11.50	3.25	409	5 P500Mhz
	1 209.27	965.91	828.38	589.86	1 389.22	1 143.70	57 573	1 run, 12 Min
RGP	12.00	2.82	10.00	3.00	11.50	3.38	408	Sun U10
	1 203.38	950.77	828.38	589.86	1 366.45	1 093.46	56 752	n/a
CLM	12.08	2.73	10.00	3.00	11.50	3.25	407	n/a
	1 210.14	969.57	828.38	589.86	1 389.78	1 134.52	57 555	Sun U2 300MHz
LC03	12.08	2.73	10.00	3.00	**11.50**	3.25	407	5xP850 MHz,
	1 209.19	963.62	828.38	589.86	1 389.22	1143.70	57412	1 run, 12 Min
BHD	12.00	2.73	10.00	3.00	11.50	3.25	407	AMD 700 MHz,
	1 239.77	1 016.86	832.06	590.68	1 417.79	1 199.95	59 210	1 run, 2.6 Min
H99	11.92	2.73	10.00	3.00	11.63	3.25	406	P200 MHz,
	1 228.06	969.95	828.38	589.86	1 392.57	1 144.43	57 876	10 runs, 13 Min
EA2	12.00	2.73	10.00	3.00	11.50	3.25	406	AMD 700Mhz
	1 220.14	977.57	828.38	589.86	1 397.44	1 140.06	57 870	3 runs, 9.1 Mins
GH01	12.00	2.73	10.00	3.00	11.50	3.25	406	4 P400 MHz,
	1 217.57	961.29	828.63	590.33	1 395.13	1 139.37	57 641	5 runs, 13.5 Min
B01	11.92	2.73	10.00	3.00	11.50	3.25	405	P200 MHz,
	1 222.12	975.12	828.38	589.86	1 389.58	1 128.39	57 710	1 run, 82.5 Min
BVH	12.18	2.73	10.00	3.00	11.50	3.25	405	Sun U10 440Mhz
	1 231,08	954.18	828.38	589.86	1 384.17	1 124,47	57 272	5 run, 120 Min

Table 2.4: Comparison of average results on 100-customer problems

New best known CNV and CTD measures for the 200-customer problem set and six new best averages for RC1-100, R2-200, RC2-200, RC2-400, RC2-600, and RC2-1000 customer problem classes are also reported in bold in Tables 2.6 to 2.10. These results are a clear indicator of the robustness of the proposed cooperative search, which also seems to perform better than the others on the RC2 problem class.

2.11.3 Solution warehouse evolution

The results presented in the preceding sub-section emphasize the interest of cooperation. We now turn to the issue of the behavior of the cooperation.

Table 2.12 displays the distribution of the number of solutions sent to the solution warehouse by each individual method for problem instance RC204 for methods **LC02** and **LC03**, respectively. For the first version, the Taburoute methods identify the largest number of good solutions, with the first thread, which has the set of parameters recommended by the authors, contributing the largest number. The unified tabu search contributes the largest number of best solutions in the second version, the Taburoute thread taking second place. It is noteworthy that for both versions, all processes contribute solutions to the warehouse and, as the comparison to the independent search method shows (Table 2.3), these solutions are important for the performance of the cooperative method in terms of solution quality.

This insight is also supported by the evolution of the cooperative meta-heuristic, as illustrated by the evolution of the best solution kept in the solution warehouse, and its comparison to the evolution of the independent search method. Figures 2.6, 2.7, and 2.8 display this behavior for a typical problem (RC204) for the Independent, **LC02**, and **LC03** search methods, respectively. In these figures, the best value of the objective function is plotted in time (in seconds) and the individual method that generated each solution is identified. The best sequential method value indicated in Figures 2.7 and 2.8 corresponds to the last solution identified by thread **TB1** in Figure 2.6. These figures clearly indicate that, after a warm up period, all methods contribute to the improvement of the solution. One also

No	Instance	Number of vehicles	Distance
1	R1-2-3	18	3164.41
2	RC2-2-6	4	3086.76
3	C1-4-7	39	7668.33
4	RC2-8-6	15	19744.73
5	RC2-10-5	18	29352.08

Table 2.5: New best solutions

Class	GH99	GH01	BVH	GES	LC02	BHD	LC03
R1	18.20	18.20	18.20	**18.20**	18.40	18.20	18.20
	3 705	3 855.03	3 677.96	3618.68	3 716.36	3 718.30	3 676.95
R2	4.00	4.00	4.10	4.00	4.00	4.00	**4.00**
	3 055	3 032.49	3 023.62	2942.92	2 986.01	3 014.28	2 986.01
C1	18.90	18.90	18.90	**18.80**	19.30	18.90	18.90
	2 782	2 842.08	2 726.63	2717.21	2 757.36	2 749.83	2 743.66
C2	6.00	6.00	6.00	**6.00**	6.00	6.00	6.00
	1 846	1 856.99	1 860.17	1833.57	1 837.02	1 842.65	1 836.10
RC1	18.00	18.10	**18.00**	18.00	18.00	18.00	18.00
	3 555	3 674.91	3 279.99	3221.34	3 449.71	3 329.62	3 449.71
RC2	4.30	4.40	4.50	4.40	4.30	4.40	**4.30**
	2 675	2 671.34	2 603.08	2519.79	2 627.31	2 585.89	2 613.75
CNV	694	696	697	694	700	695	**694**
CTD	176 180	179 328	171 715	168 573	173 738	172 406	173 061
Computer	P200	P400	Sun U10	P2000	P500	AMD700	P850
CPU Min	4x10	4x2.1	n/a	1x8	5x10	2.4	5x10

Table 2.6: Average results 200-customers

Class	GH99	GH01	BVH	GES	LC02	BHD	LC03
R1	36.40	36.40	36.40	**36.30**	36.70	36.40	36.50
	8 925	9 478.22	8 713.37	8 530.03.05	8 851.55	8 692.17	8 839.28
R2	8.00	8.00	8.00	**8.00**	8.20	8.00	8.00
	6 502	6 650.28	6 959.75	6 209.94	6 249.05	6 382.63	6 437.68
C1	38.00	38.00	38.00	**37.90**	38.00	37.90	37.90
	7 584	7 855.82	7 220.96	7 148.27	7 651.18	7 230.48	7 447.09
C2	12.00	12.00	12.00	**12.00**	12.30	12.00	12.00
	3 935	3 940.19	4 154.40	3 840,85	3 890.24	3 894.48	3 940.87
RC1	36.10	36.10	36.10	**36.00**	36.00	36.00	36.00
	8 763	9 294.99	8 330.98	8 066.44	8 704.82	8 305.55	8 652.01
RC2	8.60	8.80	8.90	8.80	8.70	8.90	**8.60**
	5 518	5 629.43	5 631.70	5 243.06	5 447.28	5 407.87	5 511.22
CNV	1 390	1 392	1 393	**1389**	1 398	1 391	1 390
CTD	412 270	428 489	410 112	390 386	409 741	399 132	408 281
Computer	P200	P400	Sun U10	P2000	P500	AMD700	P850
CPU Min	4x20	4x7.1	n/a	1x17	5x20	7.9	5x20

Table 2.7: Average results 400-customers

Class	GH99	GH01	BVH	GES	LC02	BHD	LC03
R1	54.50	54.50	55.00	**54.50**	54.80	54.50	54.80
	20 854	21 864.47	19 308.62	18 358.68	20 624.85	19 081.18	19 869.82
R2	11.00	11.00	11.00	**11.00**	11.20	11.00	11.20
	13 335	13 656.15	14 855.43	12 703.52	13 097.70	13 054.83	13 093.97
C1	57.90	**57.70**	57.80	57.80	58.00	57.80	57.90
	14 792	14 817.25	14357.11	14 003.09	14 523.65	14 165.90	14 205.58
C2	17.90	**17.80**	17.80	17.80	18.00	18.00	17.90
	7 787	7 889.96	8 259.04	7 455.83	7 704.85	7 528.73	7 743.92
RC1	55.10	55.00	55.10	**55.00**	55.20	55.00	55.20
	18 411	19 114.02	17 035.91	16 418.63	18 012.52	16 994.22	17 678.13
RC2	11.80	11.90	12.40	12.10	11.80	12.10	**11.80**
	11 522	11 670.29	11 987.89	10 677.46	11 189.75	11 212.36	11 034.71
CNV	2082	**2079**	2091	2082	2090	2084	2088
CTD	867 010	890 121	808 646	796 172	851 553	820 372	836 261
Computer	P200	P400	Sun U10	P2000	P500	AMD700	P850
CPU Min	4x30	4x12.9	n/a	1x40	5x30	16.2	5x30

Table 2.8: Average results 600-customers

Class	GH99	GH01	BVH	GES	LC02	BHD	LC03
R1	72.80	72.80	72.70	**72.80**	73.10	72.80	73.10
	34 586	34 653.88	33 337.91	31 918.47	33 552.40	32 748.06	33552.40
R2	15.00	15.00	15.00	**15.00**	15.20	15.00	15.00
	21 697	21 672.85	24 554.63	20 295.28	20 158.94	21 170.15	21 157.56
C1	76.70	76.10	**76.10**	76.20	76.30	76.30	76.30
	26 528	26 936.68	25 391.67	25 132.27	26 278.74	25 170.88	25 668.82
C2	24.00	23.70	24.40	**23.70**	24.20	24.20	24.10
	12 451	11 847.92	14 253.83	11 352.29	12 056.74	11 648.92	11 985.11
RC1	72.40	72.30	73.00	73.00	**72.30**	73.00	72.30
	38 509	40 532.35	30 500.15	30 731.07	37 722.62	30 005.95	37 722.62
RC2	16.10	16.10	16.60	15.80	**15.80**	16.30	15.80
	17 741	17 941.23	18 940.84	16 729.18	17 464.30	17 686.65	17 441.60
CNV	2770	**2760**	2778	2768	2769	2776	2766
CTD	1 515 120	1 535 849	1 469 790	1 373 662	1 472 337	1 384 306	1 475 281
Computer	P200	P400	Sun U10	P1500	P500	AMD700	P850
CPU Min	4x40	4x23.2	n/a	367.1	5x40	26.2	5x40

Table 2.9: Average results 800-customers

Class	GH99	GH01	BVH	GES	LC02	BHD	LC03
R1	**91.90**	91.90	92.80	92.10	92.20	92.10	92.20
	57 186	58 069.61	51 193.47	49 281.48	55 280.49	50 025.64	55 176.95
R2	19.00	19.00	19.00	**19.00**	19.20	19.00	19.20
	31 930	31 873.62	36 736.91	29 860.32	30 951.07	31 458.23	30 919.77
C1	96.00	95.40	95.10	**95.10**	95.30	95.80	95.30
	43 273	43 392.59	42 505.35	41 569.67	44 061.43	42 086.77	43 283.92
C2	30.20	29.70	30.30	bf29.70	30.10	30.60	29.90
	17 570	17 574.72	18 546.13	16 639.54	17 365.26	17 035.88	17 443.50
RC1	90.00	90.10	90.20	**90.00**	90.00	90.00	90.00
	50 668	50950.14	48 634.15	45 396.41	49 711.36	46 736.92	49 711.36
RC2	19.00	18.50	19.40	18.70	18.50	19.00	**18.50**
	27 012	27 175.98	29 079.78	25 063.51	26 309.10	25 994.12	26 001.11
CNV	3461	**3446**	3468	3446	3453	3465	3451
CTD	2 276 390	2 290 367	2 266 959	2 078 110	2 236 787	2 133 376	2 225 366
Computer	P200	P400	Sun U10	P2000	P500	AMD700	P850
CPU Min	4x50	4x30.1	n/a	1x600	5x50	39.6	5x50

Table 2.10: Average results 1000-customers

	GH99	GH01	BVH	GES	LC02	BHD	LC03
CNV	10397	**10373**	10427	10383	10410	10411	10 389
CTD	5 246 970	5 324 154	5 176 616	4 851 217	5 144 136	4 909 592	5 118 250

Table 2.11: Cumulative results 200-1000-customers

Method	LC02	LC03
Construction	24	0
TB1	44	11
TB2	21	
UT		24
ER	9	4
OX	3	2

Table 2.12: Distribution of improved solutions found by methods LC02 and LC03 for RC204

notices that the cooperative search finds better solutions faster than the independent search, and that it does so in the early stages of the resolution. This indicates that cooperation impacts the search early on in the process. Even in the case were a method like the unified tabu search finds better solutions more often, the other methods have a positive influence and can lead to better solutions as shown in Figure 2.8: while the unified tabu search was not able to find this best solution when run independently, it yielded better solutions at the beginning of the cooperative search, which significantly reduced the warm-up period. Figures 2.7 and 2.8 also illustrate how cooperation may allow to get out of a local minimum and continue the search toward a significantly improved solution.

On a general note, one unified tabu search cooperating with one Taburoute were able to perform better than two Taburoute versions, mainly due to the quality of solution produced by the unified Tabu search. This indicates that the quality of the individual methods in cooperation has an impact on the global performance of the method. However, as illustrated by the performance of **LC02**, the cooperative search will produce good results for as long as the individual threads produce reasonably good solutions.

The results also indicate that cooperation changes the behavior of each method by inducing new information. This property strongly suggests that cooperative search should be considered a method by itself and be studied as such. With respect to the VRPTW in particular, the solution warehouse cooperation between the Tabu search and the evolutionary mechanisms allows excellent quality solutions in very reasonable computing times.

2.12 Conclusions

We have presented a new parallel cooperative meta-heuristic for the VRPTW. The proposed meta-heuristic displays very good performance in terms of solution quality, computational effort, and robustness over the broad range of problem instances. Linear speedups have been attained and results are comparable to the

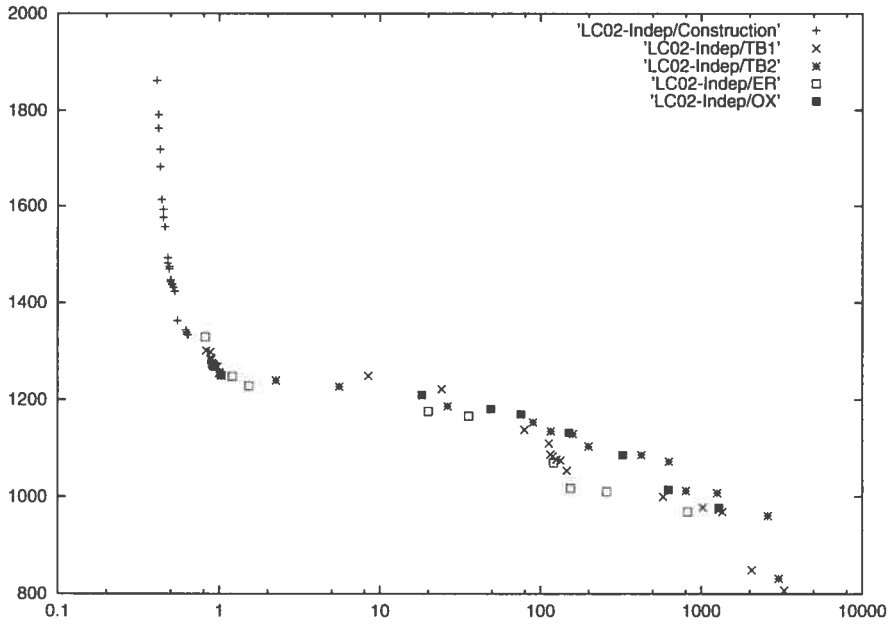

Figure 2.6: Evolution of best solution value in time (s) for RC204 by independent search

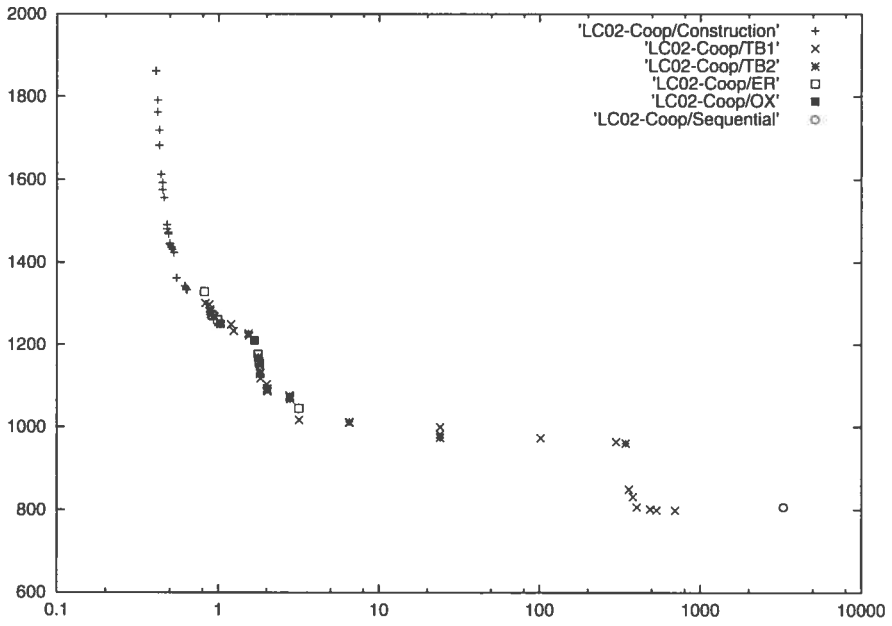

Figure 2.7: Evolution of best solution values in time (s) for RC204 by LC02

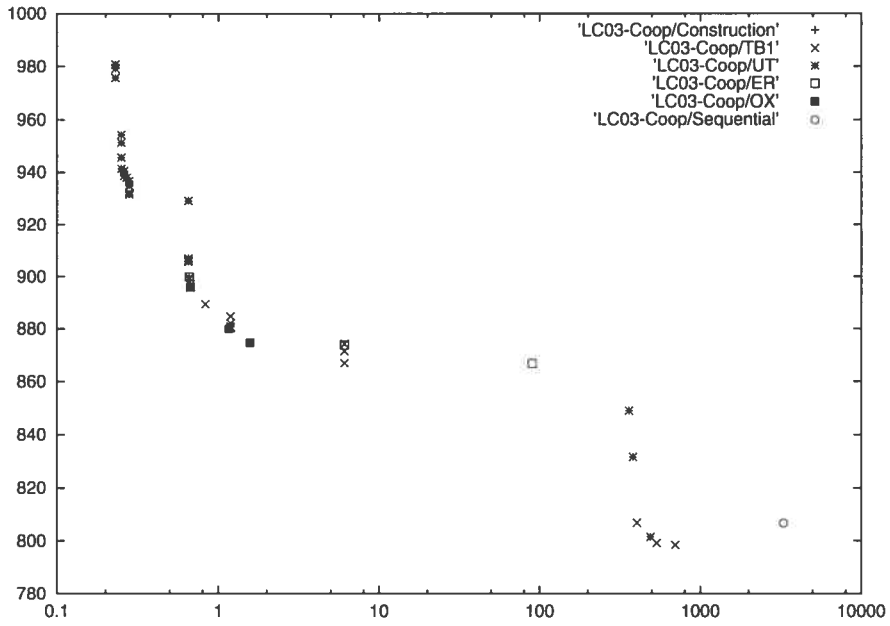

Figure 2.8: Evolution of best solution values in time (s) for RC204 by LC03

best in the literature. The cooperative framework is simple to implement and expand, and shows great promise in addressing difficult combinatorial problems, the VRPTW in particular.

2.13 Acknowledgments

Many thanks are given to Olli Bräysy and Louis-Martin Rousseau for interesting discussions and comments. We are grateful to Gilbert Laporte, Jean-Francois Cordeau, and Anne Mercier (for the Unified tabu Search) and to Michel Gendreau, Alain Hertz, and Gilbert Laporte (for Taburoute) for their very precious collaboration. Special thanks are transmitted to Prof. Dr. B. Monien and many of our colleagues at the Paderborn center for parallel computing for the use of their PC^2 cluster on which some key tests where performed in this study for the second version (LC03) of our cooperative search.

Funding for this project has been provided by the Natural Sciences and Engineering Council of Canada, through its Research Grant programs, and by the Fonds F.C.A.R. of the Province of Québec.

Annex

The Annex contains detailed results obtained by the **LC03** cooperative search meta-heuristic. Table 2.13 displays detailed results for the standard 100-customer problem instances. Tables 2.14 to 2.18 display detailed results for the extended set of problems, from 200 to 1000 customers with best-known results outlined in bold face (as of 2003/08/31 from the Sintef web site[1]).

#	R1	R2	C1	C2	RC1	RC2
01	19/1650.79	4/1253.26	10/828.94	3/591.56	14/1696.94	4/1406.94
02	17/1487.60	3/1197.66	10/828.94	3/591.56	12/1554.75	3/1407.52
03	13/1294.24	3/942.64	10/828.06	3/591.17	11/1262.98	3/1073.39
04	10/982.72	2/855.21	10/824.78	3/590.60	10/1135.48	3/798.46
05	14/1377.11	3/1013.47	10/828.94	3/588.88	13/1643.38	4/1326.83
06	12/1253.23	3/932.47	10/828.94	3/588.49	11/1427.13	3/1158.81
07	10/1113.69	2/837.20	10/828.94	3/588.29	11/1230.54	3/1062.05
08	9/964.38	2/734.85	10/828.94	3/588.32	10/1139.82	3/832.36
09	11/1199.63	3/916.47	10/828.94			
10	10/1125.04	3/963.37				
11	10/1104.83	2/923.80				
12	10/957.04					

Table 2.13: Detailed results LC03 with number of vehicles/distance, 100 customers

#	R1	R2	C1	C2	RC1	RC2
01	20/4829.21	4/4502.17	20/2708.61	6/1931.44	18/3946.11	6/3136.85
02	18/4362.31	4/3703.86	18/2972.24	6/1863.16	18/3468.85	5/2827.45
03	**18/3164.41**	4/2976.64	18/2775.40	6/1775.11	18/3203.68	4/2637.41
04	18/3141.02	4/2034.20	18/2672.45	6/1742.00	18/2963.00	4/2084.89
05	18/4213.56	4/3417.98	20/2702.05	6/1879.01	18/3902.58	4/3087.98
06	18/3659.9	4/2959.14	20/2701.04	6/1857.75	18/3632.80	**4/3086.76**
07	18/3175.75	4/2532.95	20/2723.11	6/1849.46	18/3456.98	4/2550.56
08	18/2999.35	4/1872.11	19/2778.82	6/1824.43	18/3277.21	4/2341.34
09	18/3889.96	4/3115.60	18/2739.33	6/1832.04	18/3385.56	4/2291.43
10	18/3334.02	4/2745.45	18/2663.50	6/1806.60	18/3260.31	4/2092.74

Table 2.14: Detailed results LC03 with number of vehicles/distance, 200 customers

[1]Sintef web site www.sintef.no/static/am/opti/projects/top/

#	R1	R2	C1	C2	RC1	RC2
01	40/10709.18	8/9403.89	40/7152.06	12/4116.14	36/9438.51	11/7019.89
02	36/9266.18	8/7811.05	37/7517.45	12/4048.07	36/8587.76	10/6579.41
03	36/8551.10	8/6474.06	36/8035.08	12/4179.01	36/8032.76	8/6579.41
04	37/7421.58	8/4338.61	36/7340.59	12/3929.24	36/7460.19	8/3882.74
05	36/10139.75	8/7633.26	40/7152.06	12/3941.05	36/9372.21	9/6292.63
06	36/8881.72	8/6182.17	40/7153.46	12/3877.65	36/9285.81	8/6054.46
07	36/8023.67	8/5378.68	**39/7668.33**	12/3894.13	36/8867.02	8/5727.01
08	36/7464.09	8/4349.54	38/7211.43	12/3803.17	36/8686.80	8/5093.18
09	36/9343.73	8/6711.32	37/8198.38	12/3924.39	36/8536.53	8/4745.35
10	36/8585.78	8/6094.22	36/7042.09	12/3695.85	36/8252.50	8/4418.64

Table 2.15: Detailed results LC03 with number of vehicles/distance, 400 customers

#	R1	R2	C1	C2	RC1	RC2
01	59/21875.04	11/18833.86	60/14095.64	18/7776.16	56/18295.89	15/13275.93
02	55/20131.27	11/15412.26	56/14332.05	18/8106.79	55/17413.64	12/12071.40
03	55/18517.45	12/12122.38	56/14376.87	18/8189.67	56/16422.98	11/10446.72
04	55/16656.90	11/8896.13	56/14063.21	18/7637.67	55/15847.45	11/8016.95
05	54/21580.31	12/15854.63	60/14104.35	18/7578.67	55/19769.65	13/12324.05
06	54/20291.10	11/13164.53	60/14093.83	18/7500.99	55/18482.75	12/10700.42
07	54/18570.23	11/10697.64	60/14295.01	18/7761.43	55/18286.55	11/11852.97
08	54/17030.55	11/8473.53	59/14259.39	18/7483.18	55/17598.42	11/11108.08
09	54/23250.86	11/14379.54	56/14387.23	18/7380.21	55/17228.39	11/10602.84
10	54/20794.47	11/13105.21	56/14048.18	17/8024.43	55/17435.59	11/9927.76

Table 2.16: Detailed results LC03 with number of vehicles/distance, 600 customers

#	R1	R2	C1	C2	RC1	RC2
01	80/37666.58	15/29383.35	80/25030.36	24/11677.72	73/33213.55	20/20869.21
02	73/35472.29	15/23942.54	75/25518.17	24/13724.04	72/39696.20	17/18672.43
03	73/31690.97	15/19050.31	72/27341.54	24/12369.05	72/35577.87	15/15711.84
04	73/29711.21	15/14423.95	72/26111.68	24/11932.64	72/32654.10	15/12314.59
05	72/38633.08	15/25837.8	80/25167.67	24/11559.98	73/32860.34	16/19639.76
06	72/33388.80	15/21720.99	80/25167.00	24/11418.93	73/32701.01	**15/19744.73**
07	72/30737.79	15/17810.47	80/25514.25	24/11791.88	72/43829.43	15/17964.11
08	72/29282.40	15/13741.92	77/25802.26	25/11410.32	72/43694.60	15/16939.31
09	72/34976.41	15/23734.84	74/25569.05	24/11736.54	72/41816.70	15/16767.66
10	72/33964.51	15/21929.44	73/25466.2	24/12230.00	72/41182.44	15/15792.31

Table 2.17: Detailed results LC03 with number of vehicles/distance, 800 customers

#	R1	R2	C1	C2	RC1	RC2
01	100/54724.64	19/43975.28	100/42478.95	30/16889.95	90/51529.12	22/31457.37
02	92/56362.29	19/35901.41	93/44459.70	30/18606.36	90/48146.09	19/28071.59
03	92/50604.26	20/27169.56	90/44000.84	30/17801.03	90/46201.21	18/22680.56
04	92/46017.14	19/20002.45	90/42373.79	29/19662.83	90/44072.32	18/17449.25
05	91/70838.01	20/37656.48	100/42549.13	30/16586.46	90/52780.36	**18/29352.08**
06	91/54952.03	19/32167.55	100/42479.15	30/16473.14	90/52842.71	18/28797.14
07	91/50040.50	19/25261.34	100/42689.28	31/17662.34	90/51230.16	18/27202.21
08	91/46554.80	19/19294.61	97/43302.82	29/17219.59	90/50648.85	18/26037.65
09	91/61862.40	19/35146.70	92/45372.21	30/16984.60	90/50354.98	18/25115.60
10	91/59813.39	19/32622.36	91/43133.33	30/16548.74	90/49307.75	18/23847.69

Table 2.18: Detailed results LC03 with number of vehicles/distance, 1000 customers

2.14 Synthèse

Dans cette publication, nous avons présenté le concept général de la coopération. Nous y avons étudié le contrôle de l'information entre les métaheuristiques et l'entrepôt de données. La structure de contrôle et les mécanismes de gestion des solutions au sein de l'entrepôt de données y ont aussi été détaillés.

Nous avons caractérisé le mécanisme de coopération par les quatre propriétés suivantes :

1. L'information qui est échangée entre les méthodes de recherches et l'entrepôt de données ;
2. Les méthodes de recherches impliquées ;
3. Les moments de communications ;
4. L'utilisation des informations reçues par chaque méthode et par l'entrepôt de données.

La recherche coopérative proposée échange les solutions améliorantes et utilise des recherches avec tabous afin d'éviter les pièges des optima locaux. Elle combine aussi des algorithmes génétiques qui contribuent à la création de diversité au sein de l'entrepôt de données. La communication entre les méthodes de recherches et l'entrepôt de données est asynchrone, ce qui a pour effet de ne pas interrompre les calculs en cours. Les méthodes de recherches envoient les solutions améliorantes et demandent des solutions à l'entrepôt de données selon leurs mécaniques internes (solutions de départ, parents pour les algorithmes génétiques, solutions pour diversification).

L'entrepôt de données classe les solutions en fonction d'une mesure qui combine différents facteurs liés à la qualité des solutions. Il conserve les 10*n (n étant la taille du problème) meilleures solutions, les solutions de moins bonne qualité étant éliminées au fur et à mesure. La taille de la population est proportionnelle à la taille du problème afin de représenter adéquatement la diversité des attributs des solutions possibles [16].

Le choix des solutions envoyées aux méthodes de recherche est aléatoire, mais biaisé vers les meilleures solutions. Ce mécanisme de sélection oriente vers les solutions prometteuses et laisse toutefois la place à la diversité pour éviter une convergence prématurée.

Nous avons aussi inclus un certain nombre de méthodes de créations de solutions qui offrent rapidement une base de diversité de solutions. Une méthode de post-optimisation, en plus des traditionnels 2-opt, 3-opt, OR-opt, soit les chaînes d'éjections, a permis dans plusieurs cas de vider des tournées et ainsi réduire le nombre de véhicules. L'exécution de l'ensemble des méthodes de post-optimisation permet d'obtenir un entrepôt de solutions dites « adultes ». De plus, nous avons démontré que la combinaison de solutions post-optimisées produit de meilleurs résultats qu'une post-optimisation a posteriori.

En analysant les résultats fournis par les dernières publications scientifiques, aucune méthode ne fournit toutes les meilleures solutions connues. La combinaison de différentes méthodes au sein d'une méthode coopérative est une approche qui a augmenté la robustesse de résolution et la qualité des solutions produites. Dans le présent modèle, des recherches avec tabous unifiés [23], des recherches avec tabous basées sur Taburoute [49] ainsi que des recherches basées sur la combinaison de solutions afin d'accentuer la coopération, tels des algorithmes évolutifs [65], ont formé l'essentiel des méthodes de résolution retenues pour la qualité, la robustesse, la diversité ainsi que la rapidité des solutions produites. Nous avons pu évaluer l'interaction de différentes méthodes au sein d'un entrepôt de données dans un cadre expérimental. Nous constatons que, même si certaines méthodes semblent dominantes, la combinaison avec des méthodes complémentaires améliore les résultats et la vitesse de calcul dès les premiers instants de la coopération. Ce concept d'entrepôt de données tel qu'utilisé par plusieurs implémentations comme dans Crainic et al. [31] a été utilisé afin de valider les concepts théoriques et méthodologiques énoncés.

2.14.1 Échange d'informations

Les informations trouvées par les métaheuristiques peuvent être partagés afin d'augmenter la connaissance sur l'espace de recherche. On parle de solutions complètes ou partielles, de solutions améliorant ou diversifiant les solutions déjà présentes dans l'entrepôt de données, d'informations sur l'évolution de la recherche ou bien des caractéristiques communes sur les solutions explorées [89], [41], [81]. Dans l'échange de l'information, il y a deux composantes : le choix du type d'information à échanger ainsi que les politiques d'envoi et de réception. Il est possible qu'une information qui semble pertinente à l'émetteur ne le soit pas pour tous les receveurs potentiels qui communiqueront éventuellement avec l'entrepôt de données.

L'entrepôt de données emmagasine ou élimine les informations recueillies des différentes méthodes selon la pertinence de celles-ci. Dans notre approche, la pertinence des informations de solutions a été évaluée en fonction de la qualité des solutions. Nous avons choisi de partager des solutions complètes pour plus de généralité face à d'autres problèmes difficilement décomposables et pour conserver le contexte de l'ensemble des sous-solutions qui s'y retrouve. Le partage de solutions complètes demeure un choix indépendant au problème et permet de mesurer l'avancée des méthodes qui les produisent.

Certaines méthodes partent de solutions initiales et peuvent dans des phases de diversifications utiliser une autre solution pour poursuivre leur recherche vers d'autres endroits. On remarque dans les études empiriques que les solutions de départ fournies aux diverses méthodes de résolutions ont une influence marquée sur la trajectoire de recherche.

Nous avons choisi de transmettre les solutions de façon probabiliste et biaisée vers les meilleures, ce qui, de façon empirique, a produit les meilleurs résultats.

2.14.2 Discussion des résultats

Les expériences ont été effectuées sur le jeu de données de Solomon [93] et de Gehring et Homberger [64]. Ces auteurs se sont attaqué au problème de VRPTW

respectivement de 100 noeuds et de 200 à 1000 noeuds. Les temps d'exécutions utilisés dans nos expériences ont varié de 12 minutes pour les problèmes de 100 noeuds à 50 minutes pour les problèmes de 1000 noeuds, de manière à respecter les temps utilisés par Homberger et Gehring [46]. Les résultats ont été obtenus sur un cluster de 5 Pentium III de 850Mhz avec 512Mo de mémoire sous Linux avec la librairie parallèle MPICH. Les calculs de distances entres les noeuds ont été effectués en double précision.

Les résultats obtenus démontrent d'une part que les méthodes coopératives peuvent être appliquées à un autre problème NP-difficile, celui du VRPTW, en améliorant la qualité des meilleures solutions connues et d'autre part que les méthodes coopératives sont plus performantes que les méthodes individuelles. Nous observons que les algorithmes génétiques, qui performent individuellement moins bien que les recherches avec tabous, contribuent aussi au comportement global. En effet, la recherche coopérative converge plus rapidement (prématurément) sans la diversité introduite par les algorithmes génétiques et trouve de moins bonnes solutions.

Nous concluons dans nos expériences que la coopération augmente la robustesse de la recherche pour la même puissance de calcul. Elle produit aussi des résultats de meilleure qualité. Au moment de la publication, notre recherche basée sur la coopération se retrouvait en 7e position du palmarès de SINTEF [2] pour le nombre total de véhicules (407 véhicules totaux) pour les 100 noeuds et en 3e position (10389 véhicules totaux) pour les 200 à 1000 villes. Cinq des meilleurs résultats connus ont aussi été améliorés.

En combinant des méthodes de recherches génériques - et non les meilleures méthodes, nous avons obtenu des résultats très satisfaisants. Ceci démontre que la coopération est une façon d'augmenter la qualité des méthodes existantes avec un développement minimal.

[2]SINTEF web site http ://www.sintef.no/static/am/opti/projects/top/

2.14.3 Conclusion et perspectives

Ce chapitre fut une introduction à la recherche coopérative. Nous avons présenté les mécanismes de gestion de l'entrepôt de données et des échanges qu'il effectue avec les métaheuristiques. Les solutions améliorantes sont transmises à l'entrepôt de données qui peut en fournir à la demande des métaheuristiques, et ce, selon leur propre mécanique interne. L'entrepôt conserve les solutions améliorantes et envoie, de façon probabiliste mais biaisée vers les meilleures solutions, des solutions de départ, des parents pour les algorithmes génétiques ou des solutions de diversifications aux méthodes de recherche qui en demandent. Les communications asynchrones permettent un calcul ininterrompu.

Afin d'introduire une diversité dans l'exploration, nous avons combiné des recherches avec tabous et des recherches avec algorithmes génétiques. Ces méthodes ont été adaptées pour résoudre le problème de VRPTW. D'excellents résultats et une accélération linéaire démontrent la viabilité d'un mécanisme coopératif.

CHAPITRE 3

L'IDENTIFICATION D'ÉLÉMENTS PROMETTEURS

Nous avons démontré (Le Bouthillier et Crainic [77]) que la coopération favorise la robustesse et l'efficacité de la recherche dans la résolution de problèmes combinatoires. En permettant l'échange de solutions améliorantes via un entrepôt de données, nous orientons les métaheuristiques vers les espaces de solutions prometteuses. D'un point de vue conceptuel, cet entrepôt représente une partie de la connaissance obtenue sur l'espace de solutions. Il nous donne certains éléments prometteurs ou non pour les bonnes et les moins bonnes solutions explorées jusqu'à présent. L'historique d'exploration et la fréquence d'apparition de ces éléments nous permettent de créer de nouvelles informations sur la structure du problème.

Nous explorons dans ce chapitre une méthode d'identification des éléments prometteurs ou non au sein des solutions présentes dans l'entrepôt de données. Ces patrons d'éléments prometteurs ou non nous permettent de guider les méthodes de recherche vers les espaces de solutions prometteurs et d'éviter ceux qui le sont moins.

Nous présentons la notion d'éléments de solutions, de patrons d'éléments et finalement un mécanisme de guidage associé. En comparant les résultats obtenus avec et sans mécanisme de guidage, il nous est possible de conclure sur son influence. En présence du guidage, la qualité des résultats est supérieure.

L'objectif de cette recherche est d'obtenir un mécanisme qui nous fournit une meilleure compréhension de l'espace de recherche afin d'améliorer son exploration. Ce sont des éléments clefs permettant l'accélération de la vitesse de recherche des métaheuristiques, tant parallèles que séquentielles.

3.1 Méthodologie

L'entrepôt de données contient une population de solutions, dont seules les meilleures sont conservées durant le parcours des métaheuristiques. Au sein de cette population d'élites, il y a tout un ensemble de solutions de diverses qualités. Ces solutions possèdent des éléments différents dont certains peuvent être présents dans la solution optimale. Les efforts de coopération doivent identifier ces éléments et les recherches doivent se concentrer autour de ces derniers. Restreindre l'espace de solutions (Toulouse [103]), soit en fixant les éléments désirés, soit en interdisant ceux qui ne sont pas prometteurs ou desquels on veut s'éloigner est la méthode que nous avons préconisée.

Les travaux sur la mémoire adaptative de Taillard [100] et de Crainic et Gendreau [25] [26] ont démontré l'intérêt de créer de la nouvelle information pour rendre l'entrepôt de données plus riche par une combinaison de sous-solutions. Dans cet axe de pensée, nous avons décidé d'orienter notre approche autour de la combinaison d'éléments de solutions au sens large. La solution à un problème combinatoire est constituée d'un ensemble de valeurs qui sont attribués aux variables de décisions et que nous nommons « éléments ». Il peut s'agir d'un arc, d'un noeud, d'un élément, etc. Cet élément est commun à tous les problèmes. En combinant un ensemble d'éléments qui apparaissent fréquemment, nous obtenons un patron d'éléments.

L'association des p éléments les plus fréquents au sein d'une sous-population, forme un patron de taille p. Si ce patron d'éléments apparaît dans au moins une solution de la population, nous le nommons « in-pattern » et savons qu'il est réalisable. S'il n'apparaît dans aucune solution, il constitue un « patron statistique » qui pourrait ne pas être réalisable. Pour cette étude, seuls les « in-pattern » sont considérés.

Nous divisons la population de solutions de l'entrepôt de données en trois sous-populations :

- Élite : les 10% meilleures solutions ;

- Moyenne : les solutions de qualité moyenne (80%) ;
- Non prometteuses : les 10% moins bonnes solutions.

Nous pouvons comparer l'importance des patrons les plus fréquents entre chacune de ces sous-populations. Ainsi, un patron dont la fréquence est plus élevée dans la classe « Élite » est un patron prometteur. Inversement, il est non prometteur lorsque sa fréquence d'apparition est plus élevée dans la classe non prometteuse. Fixer ou interdire les éléments de solution d'un patron prometteur ou non dans l'exploration de solution permet de restreindre l'espace de solutions. La fixation d'éléments de patrons prometteurs constitue alors une excellente façon d'intensifier les recherches autour de ces éléments. De la même façon, l'interdiction de patrons non prometteurs permet d'en écarter les recherches. Avec ce mécanisme, il est aussi possible de diversifier la recherche en interdisant les patrons prometteurs.

La publication qui figure dans la prochaine section détaille une architecture coopérative guidée. Nous effectuons un bref rappel sur la coopération et définissons le mécanisme d'identification de patrons d'éléments au sein de la recherche globale. Des phases de diversification et d'intensification ont pu être créées, grâce à la fixation et à l'interdiction de patrons de tailles variables. Les résultats sur le problème du VRPTW avec le mécanisme de guidage proposé ont permis d'obtenir le meilleur nombre de véhicules cumulatif connu (405) pour les problèmes de 100 noeuds et de s'assurer de la deuxième place en terme de qualité pour l'ensemble des problèmes. Une synthèse sur les résultats produits, les défis et les contributions succèdent à la publication.

3.2 Permission de l'éditeur

Alexandre Le Bouthillier

From: j.hansson@ieee.org
Sent: Thursday, May 12, 2005 10:33 AM
To: alexleb
Subject: Re: request for Reuse IEEE-Copyrighted Material IEEE Intelligent Systems, ISSI-0076-0205.R1

Comments/Response to Case ID: 005702B9

ReplyTo: Copyrights@ieee.org

From: Jacqueline Hansson   Date: 05/12/2005

Subject: Re: request for Reuse IEEE-Copyrighted Material IEEE Intelligent Systems, ISSI-0076-0205.R1   Send To: "alexleb" <alexleb@crt.umontreal.ca>

Dear Alexandre Le Bouthillier:

This is in response to your letter below, in which you have requested permission to reprint, in your upcoming thesis/dissertation, the described IEEE copyrighted material. We are happy to grant this permission.

Our only requirement is that the following copyright/credit notice appears prominently on the first page of each reprinted paper, with the appropriate details filled in:

' 2005 IEEE. Reprinted, with permission, from (complete publication information).

Sincerely yours,

Jacqueline Hansson

-=-
IEEE Intellectual Property Rights Office
445 Hoes Lane, Piscataway, NJ 08855
Telephone: +1 732-562-3966
Fax: +1 732-981-8062
w.hagen@ieee.org
http://www.ieee.org/copyright

Figure 3.1: Permission de l'éditeur

3.3 Publication 2: A Guided Cooperative Search for the Vehicle Routing Problem with Time Windows

Alexandre Le Bouthillier
Département d'informatique et de recherche opérationnelle and
Centre de recherche sur les transports
Université de Montréal
C.P.6128, succursale Centre-ville, Montréal, QC, CANADA H3C 3J7
alexleb@crt.umontreal.ca

Teodor Gabriel Crainic
Département de management et technologie
Université du Québec à Montréal
C.P. 8888, Succursale Centre-Ville, Montréal, QC, CANADA H3C 3P8 and
Centre de recherche sur les transports,
Université de Montréal
C.P.6128, Succursale Centre-Ville, Montréal, QC, CANADA H3C 3J7
theo@crt.umontreal.ca

Peter Kropf
Institut d'informatique,
Université de Neuchâtel, Faculté des sciences
Rue Emile-Argand 11 CH-2007 Neuchâtel, Suisse
peter.kropf@unine.ch

3.4 Abstract

An enhanced cooperative-search mechanism creates new information from exchanged solutions and *guides* the global search with a pattern identification mechanism.

Keywords : parallel computation, parallel cooperative search, Vehicle Routing Problem with Time Windows

3.5 Introduction

Efficiency and robustness are the primary qualities of good meta-heuristics and parallel computing may significantly enhance these characteristics. Three forms of parallelism can be applied to meta-heuristics: (i) Division of computer-intensive tasks at a low algorithmic level, (ii) Explicit domain decomposition of the solution or search space, (iii) Multi-thread search. An independent multi-thread search produces the best solution among those found by each independent method. Multi-thread cooperative search implements a mechanism that allows information (e.g., solutions) to be exchanged among the search threads. Several such mechanisms have been proposed in the literature and in many cases cooperative multi-search offered superior performance in terms of better solutions and shorter resolution times (Crainic and Toulouse [29]).

While all parallelization strategies may speed up the resolution, cooperative search methods may also increase the robustness of the global search (Crainic and Toulouse 2003). Cooperative search combines the efforts of several independent meta-heuristics by using a so-called solution warehouse (according to the information stored, the names "adaptive" and "central memory" are also used). This device receives "good" solutions from the search threads and, on demand and according to their own internal logic, provides them in return with solutions to, for example, diversify the search. This simple mechanism allows the asynchronous communication and exchange of solutions that influences the search trajectory of each method. Enhanced with simple extraction rules for the returned solutions, simple cooperation was successfully used to address a number of difficult combinatorial problems: network design, multi-commodity location-allocation (Crainic, Toulouse and Gendreau [30]), circuit partitioning and Vehicle Routing Problem with Time Windows (VRPTW) (Le Bouthillier and Crainic [77]).

From a conceptual point of view, the solution warehouse contains useful indirect information for the global search. Thus, for example, the history of solution discovery and the frequency of appearance of certain elements in solutions of particular

quality can be used to create new information about the search space. This process of information creation can transform the solution warehouse in an intelligent data warehouse that holds more complex information and guides individual methods towards promising or unexplored regions.

The goal of this paper is to present a mechanism that endows cooperative search with capabilities to create new information and guide the global search. The proposed identification pattern mechanism sends information to individual metaheuristics about promising and unpromising patterns of the solution space. By fixing or prohibiting specific solution element values in particular search methods, we can focus the search to desired regions. This mechanism may thus be applied to enforce a better coordination between the individual methods and control the diversification and intensification of the global search. We apply this mechanism to the Vehicle Routing Problem with Time Windows (VRPTW). Experimental results on an extended set of benchmark problem sets illustrate the benefits of the proposed methodology.

The main contribution of this paper is the introduction of an enhanced cooperative search mechanism that creates new information from exchanged solutions and perform global intensification and diversification phases. The proposed framework does not suppose any specific problem structure and may thus be applied to a wide range of combinatorial problems. The paper also reports very good solutions on benchmark problem sets for the VRPTW: 139 best known results found and a new best known average on the problems of size 200 and 1000.

The paper is organized as follow. Section 2 briefly presents a framework of cooperative search. Section 3 introduces the pattern identification mechanism and the global search phases. Section 4 details the implementation of a guided cooperative search applied to the VRPTW. Section 5 presents the computational results and analyzes them both from the point of view of solving the VRPTW and from that of the performance of the parallel strategy. Conclusions and perspectives are the subject of the last section.

3.6 A Cooperative Search Framework

Figure 3.2 illustrates the cooperative search framework made up of a number of independent processes (threads) of possibly different types, which communicate through the solution warehouse. A search thread either heuristically constructs new solutions, or executes a neighborhood-based improving meta-heuristic through the search space, or implements a population-based meta-heuristic (e.g., evolutionary algorithms, Scatter Search, Path relinking), or performs post-optimization procedures (e.g., intensive local search) on solutions in the solution warehouse. Improving meta-heuristics, such as Tabu search, aggressively explore the search space, while population-based methods contribute toward increasing the diversity of solutions exchanged among the cooperating methods. When the same meta-heuristic is used by several search threads, the initial solution and particular setting of a number of important search parameters differentiate each search thread from the others.

The cooperation aspect of the parallelization scheme is achieved through asynchronous exchanges of information. Information is shared through a solution warehouse or pool of solutions. In this scheme, whenever a thread desires to send out information (e.g., when a new local optimum is identified), it sends it to the pool. Similarly, when a thread accesses outside information (to diversify the search, for example), it reaches out and takes it from the pool. Communications are initiated exclusively by the individual threads, irrespective of their role as senders or receivers of information. No broadcasting is taking place and there is no need for complex mechanisms to select the threads that will receive or send information and to control the cooperation. The solution warehouse is thus an efficient implementation device that allows for a strict asynchronous mode of exchange, with no predetermined connection pattern, where no process is interrupted by another for communication purposes, but where any thread may access at all times the data previously sent out by any other search thread.

The solution warehouse keeps the information in an order appropriate for the exchange mechanism considered. To fully characterize the cooperation process, one

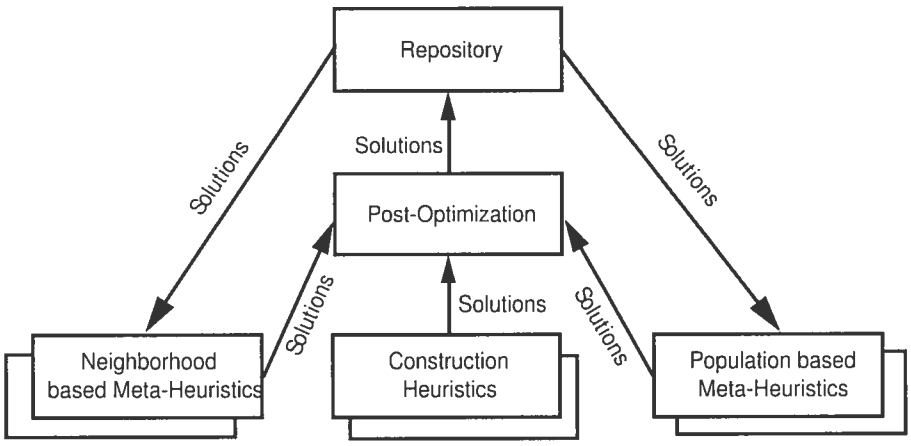

Figure 3.2: Abstract Framework of a Cooperative Search

has to specify (i) the information which is to be shared; (ii) the particular methods that makeup the cooperative search; (iii) when and how communications occur; (iv) the utilization each thread makes of the imported information (Crainic and Toulouse [29]). The information exchanged among cooperating procedures has to be meaningful, in the sense that it has to be useful for the decision process of the receiving threads. Information indicative of the current status of the global search or, at least, of some individual meta-heuristic is, in this sense, meaningful.

Two main classes of cooperation mechanisms are found in the literature, based on partial and complete solutions, respectively. Adaptive memory methods (Rochat and Taillard [87]) usually store partial elements of good solutions and combine them to create new complete solutions that are improved then by the cooperating threads. Central memory approaches exchange complete elite solutions among neighborhood and population-based meta-heuristics (Crainic, Toulouse, Gendreau [30], Crainic and Toulouse [29], Le Bouthillier and Crainic [77]).

In a simple central memory cooperation scheme (e.g., Le Bouthillier and Crainic [77]), threads share information about their respective good solutions identified so far. When a thread improves the imported solution or when it identifies a new best solution, it sends it to the solution warehouse. This scheme is intuitive and simple, and it satisfies the meaningfulness requirement. The selection of the

methods involved in cooperative search should be oriented toward obtaining: (i) Good quality solutions; (ii) A broad diversity of solutions to facilitate the discovery of promising regions; (iii) The rapid production of intermediate solutions to feed the information exchange mechanisms; (iv) A mechanism that combine various solutions to create diversity; (v) A mechanism that has the ability to escape local optima.

The solution warehouse is thus the core of the cooperation mechanism. It keeps good solutions and is dynamically updated by the independent search processes. The pool of solutions forms an elite population from which the independent procedures require solutions at various stages of execution. Solutions are ordered according to a predetermined utility measure that quantifies the solution quality. The solution utility can be its objective value or a combined measure of solution's properties.

Independent methods send their improved solutions to the post-optimization algorithms. These solutions are considered in-training until they have been post-optimized and sent as Adult solutions to the solution warehouse. Duplicate solutions received in the solution warehouse are eliminated.

All requests for solutions initiated by the independent processes are sent to the solution warehouse that responds by sending an Adult solution. Solutions are selected randomly according to probabilities biased toward the best based on the same function used to order solutions in the solution warehouse.

The population size in the solution warehouse is set relatively to the problem size and the worst results are eliminated as needed. No direct communications take place between processes thus enforcing their independence and the asynchronous mode of exchange. This scheme makes the cooperation design simpler and, eventually, allows easy modification of the parallel system by adding new methods or dropping inefficient ones. Moreover, it does not assume any specific problem, which makes it equally relevant for problems where solution components may be easily defined (e.g., the routes in vehicle routing problems) and for problems where such structures are much less apparent (e.g., network design). The goal now is to im-

prove upon this simple cooperating scheme by extracting new knowledge from the information exchanged, to yield a more efficient global search.

3.7 Pattern Identification Mechanism and Global Search

In this section, we introduce a mechanism to extract knowledge from the information exchanged and guide each search method towards promising or unexplored regions of the solution space. It uses a pattern identification mechanism on the solutions present in the solution warehouse that is then used to fix or prohibit specific solution elements (e.g., arcs in network-based problems) for part of the search performed by particular individual meta-heuristics. One may therefore constrain the search space of particular meta-heuristics in cooperation and thus perform global intensification and diversification phases to guide the exploration of the solution space and control the quality and diversity of the solution warehouse population.

To enhance the clarity of the presentation, we focus in the following on problems that may be described in terms of inclusion/exclusion of arcs in given networks. Network design, traveling salesman, and vehicle routing problems are important combinatorial problem classes that belong to this category. We emphasize, however, that the concepts presented in this section may be extended to any problem definition and solution element.

3.7.1 Pattern definition

For the class of problems described above, a very simple and general pattern definition may be based on the inclusion of arcs in particular solutions.

Consider the frequency of inclusion of arcs in a given subset of the solution warehouse. In particular, this subset may be the entire population, an elite (e.g., with solution in the 10% best), average (between the 10% and 90% best) or worst (the last 10%) group of solutions. An arc with a high frequency in a given group signals that the meta-heuristics participating to the cooperation have often produced solutions that include that arc. Tagging solutions to identify the last algorithm that

sent it, one may induce similar information by subset of participating algorithms as well. When the frequency of inclusion of several arcs is considered, patterns emerge among the solutions of the solution warehouse or the specific group examined.

We define a pattern relative to a subset of solutions as a vector containing the arcs with the highest or lowest frequency of appearance among the arcs of the given solution subset. The length of the pattern can be n = `1, 2, ... maximum number of arcs`. A frequent (infrequent) pattern relative to a set of arcs is built of arcs with high (low) frequency of appearance in the solutions of the set. High (low) frequency arcs are selected sequentially in decreasing order from the highest (increasing from the lowest) frequency value.

We may select patterns from specific sub-populations (e.g., elite, average, and worst) and compare the rate of appearance of a specific pattern between them. These comparisons form the basis of the guidance mechanism proposed in this paper.

3.7.2 Comparing pattern-appearance frequencies among sub-populations

Not all patterns correspond to sets of arcs appearing in at least one of the solutions of a given set (e.g., the subset used to generate the pattern or even the entire solution warehouse). We define an *in-pattern* as a pattern that actually appears in at least one solution of the subset of solutions considered, as opposed to a *statistical pattern* that does not appear in any solution of the subset. A statistical pattern is thus only a consequence of the statistical process of accounting for the frequency of individual arcs.

Consider an *in-pattern* of length n common to the three sets of elite, average, and worst solution groups. Two meaningful situations can occur with respect to the frequency of appearance of the pattern in these sets as one move from the elite to the average to the worst subpopulation: The frequency is either increasing or it is stable or decreasing. We call the *in-pattern unpromising* in the first case and *promising* in the latter.

Promising and unpromising patterns may then be used to constrain for a certain

time the search space of particular methods in the cooperation and thus induce global intensification and diversification phases, as described in the next subsection.

3.7.3 Using pattern identification - Global search

Global search intensification and diversification phases may be triggered by fixing (including) or prohibiting (excluding) arcs in the solutions a given meta-heuristic explores during a certain period.

Consider an unpromising pattern of length n. To intensify the search around solutions with "good" elements, one prohibits the arcs defining the pattern. On the other hand, fixing these arcs for a number of iterations will diversify the search relative to the current set of "good" element values. Symmetrically, given a promising pattern of length n, one intensifies the search by fixing the arcs in the pattern, while prohibiting them diversifies it.

We define the global search as the cumulative search effort of the individual methods. To prevent individual methods from converging too rapidly, we favor diversification by prohibiting arcs during the initial phases of the global search. Later on, we encourage intensification by fixing promising arcs to enforce the exploration of theses promising regions of the solution space. We also vary the length of the patterns as a mean of modulating the intensity of global diversification and intensification phases and thus influence the evolution of the diversity and quality of solutions in the solution warehouse.

At the beginning of the search there is not sufficient information gathered as the solution space is insufficiently explored. Therefore, the solution warehouse cannot be considered representative of the search of the individual meta-heuristic, even when the population is diverse. Consequently, initially, we build patterns of increasing length from the average subpopulation only, in order to identify more rapidly promising ones. As the search progresses, patterns from elite and worst subpopulations are built as described above. When a statistical pattern is created, we reduce the length of the pattern until an in-pattern is obtained.

Several modifications must be made to the initial framework presented in the

previous section. The solution warehouse now includes a process to identify and manage patterns. This process includes decisions on when to compute particular patterns and to what individual meta-heuristic to send them. It becomes, in fact, the "new" global meta-heuristic corresponding to the global guided cooperative search. As for the solution warehouse, it now contains solutions and pattern information, thus a data warehouse. Communications from the solution warehouse to the individual meta-heuristics are modified as well. The appropriate pattern and instructions on fixing or prohibiting arcs are sent along with the solution (selected according to the original criteria). With respect to the individual meta-heuristics, each needs to be modified to cope with the instructions relative to fixing or prohibiting arcs. Figure 3.3 illustrates the guided cooperative search framework as applied to the VRPTW.

3.8 Guided Cooperative Search for the VRPTW

To illustrate the mechanism described in the previous sections, we developed a guided cooperative search for the Vehicle Routing Problem with Time Windows (VRPTW). The application is described in this section. Experimental results are discussed in the next section.

The VRPTW is a well-known combinatorial problem that has been extensively studied and is thus well suited for benchmarking. We address the single depot VRPTW, as illustrated in Figure 3.4, where one is given a set of customers with known positive demands and specific time intervals when service can be provided. A fleet of homogeneous vehicles of known capacity is available at a given depot to perform this service. The objective is to find a set of closed routes (or tours) that start and end at the depot within its opening hours, such that the total cost of performing the service is minimized, customers are visited and served during their specified time windows, and vehicles are not overloaded.

In the problem version we address, cost is a combination of two factors: the number of vehicles (routes) used and the total distance traveled. A high cost is

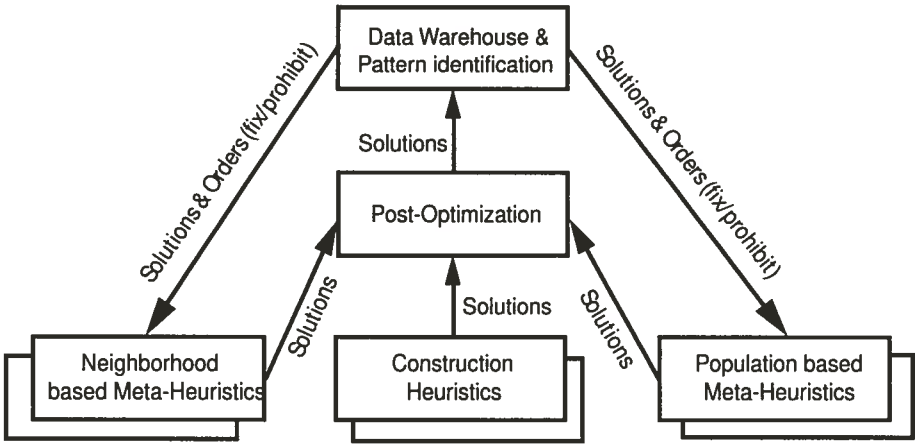

Figure 3.3: Guided Cooperative Search applied to VRPTW

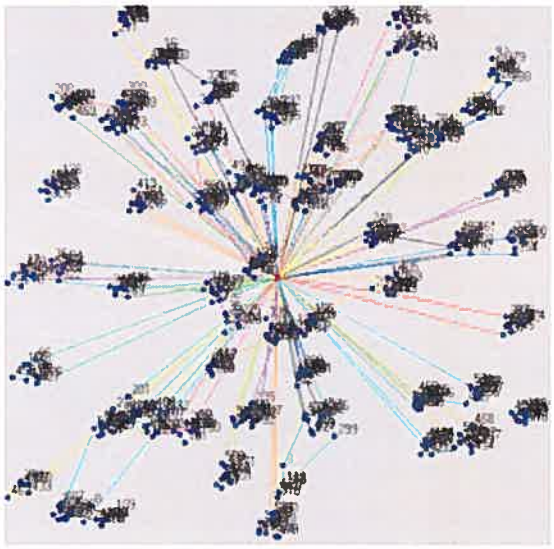

Figure 3.4: Best known solution for problem C1-6-4

associated with vehicle utilization to enforce the search towards solutions with reduced number of vehicles. Each customer is visited only once. A vehicle cannot arrive later than the customer's closing time, but is allowed to arrive before the associated opening time, in which case it waits, without explicit penalty, until the customer is ready. Once the service starts, it is carried on until completion, even if the service ending time might be later than the expiration of the time window. Cordeau et al. [21] review problem variants, formulations, and solution methods for the VRPTW.

Le Bouthillier and Crainic [77] presented a cooperative parallel method for the VRPTW based on the simple solution warehouse mechanism presented in Section 2. The cooperation involved two Tabu search methods that perform well sequentially, the Unified Tabu (Cordeau, Laporte, and Mercier [23]) and Taburoute (Gendreau, Hertz, and Laporte [49]), two simple evolutionary algorithms with Order and Edge Recombination Crossover, respectively, as well as a number of post-optimization methods (2-opt, 3-opt, Or-opt, and Ejection Chains) that were used to reduce the number of vehicles and the total traveled distance. Four simple construction algorithms were used to provide initial solutions to the population.

We applied the proposed cooperation framework to the parallel method of Le Bouthillier and Crainic [77]. In the VRPTW context, the definition of an in-pattern of length n is straightforward. The problem is defined on a network, where an arc corresponds to a possible movement between two customers or between a customer and a depot. The procedures introduced in the previous section were used to define patterns.

Fixing and prohibiting arcs in the solutions explored by the four meta-heuristics is straightforward as well. Fixing (including) arcs always leads to a non empty solution space that, at the most extreme, may reduce to a single solution that represents a very (too) long pattern. Prohibiting (excluding) patterns may lead to an empty feasible solution space when patterns are too long. To avoid both situations and strike a balance between the number of feasible solutions and the size of the constrained solution space, we limit the pattern length to 25% of the

size of the problem.

The computing time allocated to the cooperative method is divided into four phases: Two phases of diversification at the beginning to broaden the search, followed by two intensification phases to focus the search around promising regions. The four phases proceed as follows:

- Phase I. Built unpromising in-patterns of frequent arcs in the average sub-population and prohibit them in the independent meta-heuristics;
- Phase II. Prohibit arcs from frequent unpromising in-patterns from the worst subpopulation;
- Phase III. Work with the average subpopulation and fix arcs from frequent promising in-patterns;
- Phase IV. Build frequent promising in-patterns from the elite sub-population and fix the arcs for the meta-heuristic searches.

Pattern lengths are explored in decreasing length in the first two phases and in increasing length in the last two phases, by increments of 1 unit.

3.9 Computational Experiments

The experimentation has a dual objective. On the one hand, we aim to compare the guided cooperative search to the simple version and to the best performing methods proposed in the literature for the VRPTW and, thus, to validate our claim that the proposed method offers competitive performance in terms of both solution quality and computational effort. On the other hand, we also aim to evaluate the impact of guiding the search toward or away from specific patterns and performing diversification and intensification to control the entropy of the population.

A different search method, Taburoute, an Unified Tabu Search algorithm, and two evolutionary algorithms (OX and ER), is run on each of the four processors.

The solution warehouse, the post-optimization procedures, the pattern identification method, and the construction methods are run on another processor for a total of five processors in this study.

For Taburoute we use the parameter settings indicated in the original paper for the VRP (Gendreau, Hertz, and Laporte [49]). Tabu tags were set to a length varying between 5 and 10 iterations, 15% of the nodes were evaluated in the p-neighborhood dimensions and the initial solution was selected as the best from 15 initially generated solutions and the best solution from the solution warehouse. A penalty of 1 was used for frequently moved arcs. The parameters for the value function presented in the initial article by Cordeau, Laporte, and Mercier (Cordeau, Laporte, and Mercier [23]) (alpha:=1; beta=1; gamma=1) were used for the Unified Tabu. Finally, an arc mutation probability of 1 percent was used on temporary copies of the parents for the crossovers used by the evolutionary algorithms.

Tests have been carried out on the standard set of test problems proposed by Solomon [93]. The set contains 56 problems of 100 customers each. We also used the extended set produced by Homberger and Gehring [64] with 300 problem instances that vary from 200 to 1000 customers. The Solomon and extended problems are divided into six categories, named C1, C2, R1, R2, RC1, and RC2. For all problem instances, customers are distributed in a [0,100] square unit. The customers in sets C are clustered together, while those in sets R are distributed randomly. Problems in sets RC combine the two characteristics. Time windows at the depot are relatively small for problems of type 1, to allow fewer customers to be served by each route; time windows are larger for problems of type 2. The service time is of 10 units by customer for problems of type R and RC, and of 90 units for class C.

Solutions in the solution warehouse are sorted, first by the number of vehicles, second by a weighted sum, C(p), of attributes: the total time required to serve all customers, the associated total distance and total waiting time at customers, and the sum of the slack left in each time window: C(p)=W1*totalTime + W2*totalDistance + W3*totalWait + W4*totalSlack.

Parameters W1 to W4 were set to 1 in all the reported experiments. This mea-

sure combined with the number of vehicles gives us an overall idea of the solution quality (totalTime and totalDistance) and flexibility (totalWait and totalSlack). The last two measures indicate how much slack there exists in the solution and how easily feasible neighboring solutions may be explored.

In previous research (Le Bouthillier and Crainic [77]), cooperative search was found to provide faster results of equivalent or better quality than each of its independent searches. We therefore compare only simple and guided parallel cooperative searches.

Runs of 12 min wall-clock time were performed by the cooperative meta-heuristics for each of the 100 city problems. Longer running times, equal to those reported by Homberger and Gehring [64] were allowed for the larger problem instances. These times go up to 50 min wall-clock time for the 1000 city problem. To be able to compare to the wall-clock time of the simple cooperative search, we created virtual machines under VMware and forced their CPU clocks to emulate the machines used for the previous study (Le Bouthillier and Crainic 2005). Using faster virtual or physical machines will only decrease the wall-clock time for the same results. A virtual cluster of five Pentium III 850MHz CPU computers with 512MB of RAM under Linux was used. Computations of distances were carried out in double precision. The implementation is machine independent and can be run with MPICH on Unix, Windows, or Linux.

The four global phases that prohibit or fix arcs were each allocated 1/4 of the total wall-clock execution time. In-pattern lengths were at most 25% of the problem size.

Tables 1 and 2 display the average results per class for the cumulative number of vehicles (CNV) and distance (CTD) for the standard Solomon problems and for extended set of problems, respectively. Best results are shown in bold face and, in Table 1, authors are presented in decreasing CNV/CTD value.

Results of the simple Cooperative Search (LC03) and Guided Cooperative Search (LCK05) are compared to those of the best methods (published or not)

Author	R1	R2	C1	C2	RC1	RC2	CNV/CTD	Experiment
RT	12.25	2.91	**10.00**	**3.00**	11.88	3.38	415	SG. 100MHz
	1208.50	961.72	**828.38**	**589.86**	1377.39	1117.44	57231	1 run, 92.2 Min
CLM	12.08	2.73	**10.00**	**3.00**	11.50	3.25	407	n/a
	1210.14	969.57	**828.38**	**589.86**	1389.78	1134.52	57555	Sun U2 300MHz
LC03	12.08	2.73	**10.00**	**3.00**	11.50	3.25	407	5xP850 MHz.
	1209.19	963.62	**828.38**	**589.86**	1389.22	1143.70	57412	1 run, 12 Min
H99	11.92	2.73	**10.00**	**3.00**	11.63	3.25	406	P200 MHz.
	1228.06	969.95	**828.38**	**589.86**	1392.57	1144.43	57 876	10 runs, 13 Min
GH01	12.00	2.73	**10.00**	3.00	11.50	3.25	406	4 P400 MHz.
	1217.57	961.29	**828.63**	590.33	1395.13	1139.37	57641	5 runs, 13.5 Min
B01	11.92	2.73	**10.00**	**3.00**	11.50	3.25	405	P200 MHz.
	1222.12	975.12	**828.38**	**589.86**	1389.58	1128.39	57710	1 run, 82.5 Min
LCK05	**11.92**	2.73	**10.00**	**3.00**	11.50	3.25	405	5xP850 MHz.
	1214.20	954.32	**828.38**	**589.86**	1385.30	1129.43	57360	1 run, 12 Min
BVH	12.18	**2.73**	**10.00**	**3.00**	**11.50**	**3.25**	**405**	Sun U10 440Mhz
	1231.08	**954.18**	**828.38**	**589.86**	**1384.17**	**1124.47**	**57272**	5 run, 120 Min

Table 3.1: Comparison of average results on 100-customer problems

for the VRPTW on the benchmark site of SINTEF [1] on February 15, 2005: The unified Tabu Search of Cordeau, Laporte and Mercier ([23], denoted CLM), the evolutionary algorithm of Homberger and Gehring ([64], denoted HG), the two stage hybrid local search of Bent and Van Hentenryck ([5], denoted BVH), the Active Guided Evolution Strategies of Mester and Bräysy ([82], denoted MB), Rochat and Taillard ([87], denoted RT), Gehring and Homberger ([47], denoted GH01), and Bräysy ([12], denoted B01). We refer the reader to Le Bouthillier, Crainic, and Kropf [78] for detailed results on standard and extended Solomon's benchmark sets of problems.

The proposed method, LCK05, yields very good results. It appears in second place in both tables according to the CNV/CTD ratio. Relative to the standard Solomon problems, the method we propose yields a total number of vehicles of 405, which is the lowest number obtained so far and makes the guided cooperative search one of the best meta-heuristics currently available for the VRPTW.

We observe that, compared to the simple cooperative search, the new method reduced the number of vehicles by 2 and the distance by 52.38 units. We found 24 of the best known results and report the best known CNV. The guided cooperative search reports a new best average for the R1 problem class. For C1 and C2 problem classes of 100 customers, almost all methods found the best known average number

[1] Sintef web site www.sintef.no/static/am/opti/projects/top/

of vehicles and the total distance. For the other classes of problems, we found the best known number of vehicles in all instances and at most 0.16% of increase in distance when compared to the other methods.

Problem	MB		LC03		LCK05		HG	
	CNV	CTD	CNV	CTD	CNV	CTD	CNV	CTD
200 total	694	168573	694	173061.631	**694**	**169958.49**	694	173312.32
400 total	1389	390386	1390	410329.98	1389	396611.15	**1388**	**409763.38**
600 total	2082	796172	2088	840582.74	2086	809493.46	**2076**	**851680.33**
800 total	2765	1361586	2766	1475435.14	2761	1443399.01	**2755**	**1479801.56**
1000 total	3446	2078110	3451	2225366.13	3442	2133644.57	**3439**	**2236582.89**
Total	10376	4794827	10389	5124775.621	10372	4953106.68	**10352**	**5151140.48**

Table 3.2: CNV/CTD for 200-1000 customers

Table 2 displays the results for the cumulative number of vehicles (CNV) and distance (CTD) aggregated by problem size for the extended set of problems. There are not many authors that addressed the entire problem set, which explains the limited number of entries in the table (the best methods were selected).

We observe that we obtain a new best known CTD value for problems of size 200. For the other problem classes, the difference to the other methods is less than 0.48% in terms of the cumulative number of vehicles. In all cases, we improved upon the simple cooperative search. We report a slightly higher total number of vehicles compared to HG for problem of size 200 to 1000 but report a reduction in the total cumulative distance by 3.94% for all classes.

3.10 Conclusions and Perspectives

We presented an enhanced cooperative search mechanism that creates new information from exchanged solutions and thus performs global intensification and diversification phases. The proposed framework does not suppose any specific problem structure, even though we illustrated the methodology in the context of problems defined by the inclusion/exclusion of arcs in particular networks. We also applied this methodology to the VRPTW and report very good solutions on the extended benchmark problem sets: 139 best known results found and the best known average on the problems of size 200 and 1000.

Experimental results showed that pattern identification method yields good information to guide the global search. Patterns of elements may be constructed independently of particular solution structures and applied to a wide range of combinatorial problems. Patterns of elements could then be used in various sequential and parallel methods to orient the search.

Cooperative search is thus quite simple to implement (represents less than 10% of the project in lines of code) and the pattern identification method provides an easy mechanism to constrain the search within promising regions of the search space or away from unpromising regions. Very good quality solutions were found in linear speed up by combining good off-the-shelf methods without any particular parameter tuning. The quality of the individual methods influences the global search quality. Improved solutions may be found, however, by generating new information from the frequency of pattern appearance in the best solutions visited and by using the new information to guide the global search.

3.11 Acknowledgements

We are grateful to Jean-François Cordeau for the generous offer of sharing his Unified Tabu search code and to Michel Gendreau for his ideas and assistance with the original version of Taburoute. This research has been partially supported by the Natural Sciences and Engineering Council of Canada through its Discovery Grants program.

3.12 Synthèse

Nous avons présenté un mécanisme permettant la construction de patrons d'éléments indépendants du type et de la structure du problème. Ces patrons d'éléments peuvent êtres prometteurs ou non et être imposés aux métaheuristiques afin de contraindre leur espace de solutions.

Le défi était de créer une meilleure cohésion entre les différentes métaheuristiques impliquées dans la coopération, en laissant la liberté à chaque méthode d'utiliser sa mécanique interne lors de l'exploration. L'imposition d'un mécanisme de guidage nous semblait appropriée, pour autant qu'il ne soit pas lié à un problème précis ou à une structure particulière. Nous voulions aussi limiter les modifications algorithmiques de chaque métaheuristique afin d'obtenir un mécanisme simple. Dans notre cas, fixer ou interdire certains éléments ne changent pas la mécanique interne des métaheuristiques, qui nécessite de plus un minimum de modifications.

La publication [79] met en contexte les recherches coopératives et en justifie l'utilisation. Les métaheuristiques impliquées dans la coopération ont été choisies pour plusieurs raisons :

- La qualité des solutions produites ;
- La diversité des solutions produites afin de faciliter l'exploration de nouvelles régions ;
- La production rapide de solutions pour une mise en place accélérée de la coopération ;
- La présence de mécanismes permettant d'échapper aux minima locaux.

Les métaheuristiques utilisées au sein du guidage coopératif furent les mêmes que dans [77], à savoir, deux algorithmes avec tabous : un taburoute et une recherche avec tabous unifiés ainsi que deux algorithmes génétiques : l'un avec crossover ER et l'autre avec crossover OX.

Afin d'améliorer la qualité des solutions produites, nous avons remplacé un des Taburoutes [49] utilisés dans [77] par une recherche avec tabous unifiés [23]. Pour

leur part, les algorithmes génétiques assurent une diversité au sein de l'entrepôt de données.

L'échange de solutions améliorantes demeure toujours le mécanisme de communication asynchrone entre les métaheuristiques et l'entrepôt de données qui conserve les meilleures solutions. Pour permettre une diversité dans les solutions, la dimension de l'entrepôt est proportionnelle à la taille du problème. Les mécanismes de gestions de l'entrepôt qui contrôlent la sélection et l'élimination de solutions sont basés sur la même mesure combinée de leur qualité.

L'information échangée au sein des recherches coopératives présentées dans [77] a été enrichie. En effet, en identifiant les éléments fréquents au sein des bonnes et moins bonnes solutions, il est possible de créer des patrons de solutions prometteuses ou non. En imposant ces patrons aux métaheuristique, nous contraignons l'espace de solutions. Ce mécanisme générique de guidage n'est pas spécifique à un problème particulier, seule la définition de l'élément change. Dans le contexte du VRPTW, nous avons utilisé les arcs apparaissant dans les solutions. Ce mécanisme de guidage n'est pas spécifique à une métaheuristique particulière : nous l'avons appliqué avec succès à un mécanisme coopératif qui incluait des recherches avec tabous et des algorithmes évolutifs. Ce mécanisme générique s'applique aussi à toute métaheuristique séquentielle ou parallèle pourvu qu'elles produisent des solutions en cours de recherche. Ces solutions sont ainsi utilisées pour l'identification de patrons.

Pour orienter les recherches aux endroits prometteurs et les faire dévier de ceux qui le sont moins, il est nécessaire d'enrichir les communications. En plus d'envoyer des solutions complètes aux métaheuristiques, nous leur transmettons des ordres pour fixer ou interdire certains éléments, en l'occurrence des arcs pour le problème de VRPTW.

3.12.1 Discussion des résultats

L'objectif était de comparer les méthodes de recherches coopératives avec et sans guidage. Afin de conserver une puissance de calcul équivalente dans les expériences,

nous avons utilisé la virtualisation pour limiter la cadence de processeur à celle utilisée dans la publication [77].

Le guidage a été effectué en phases de diversification et d'intensification afin de créer une oscillation stratégique autour d'espaces de solutions prometteuses ou non. La longueur des patrons a aussi été variée pour laisser place à l'exploration des métaheuristiques. Nous avons toutefois limité leur taille pour que l'espace de solutions soit suffisamment grand.

Les méthodes LC03 (recherches coopératives sans guidage) et LCK05 (recherches coopératives avec guidage) ont pu être comparées. Il en ressort que l'ajout d'un mécanisme de guidage est suffisant pour diminuer le nombre de véhicules par 19 et la distance de 171 718 unités pour l'ensemble des classes de problèmes (100 à 1000 noeuds).

3.12.2 Conclusion et perspectives

En fixant ou en interdisant certains éléments prometteurs ou non des solutions visitées au cours des recherches, nous créons des phases d'intensifications et de diversifications. Notre expérimentation à la section 3.9 a démontré que les recherches coopératives produisent de meilleurs résultats lorsque les recherches sont guidées.

Nous croyons que le mécanisme général de guidage proposé permet d'améliorer la connaissance des méthodes de recherche sur la structure d'un problème combinatoire. Ces méthodes peuvent fonctionner tant de manière parallèle ou non que de manière coopérative ou non, de même qu'avec ou sans mémoire. Il est ainsi possible d'enrichir les métaheuristiques existantes par l'ajout du mécanisme de guidage proposé.

CHAPITRE 4

GUIDAGE DYNAMIQUE DE LA RECHERCHE COOPÉRATIVE

4.1 Introduction

La recherche coopérative permet d'améliorer la qualité et la robustesse des solutions obtenues par les métaheuristiques impliquées. Nous avons démontré (Le Bouthillier et al. [79]) empiriquement une amélioration de la performance et de la robustesse des solutions par l'ajout d'un mécanisme simple de guidage à la coopération. Ce mécanisme consiste à identifier les patrons d'éléments (prometteurs ou non) des solutions de l'entrepôt de solutions. Selon la phase d'intensification ou de diversification, ces patrons de longueurs variables sont fixés ou interdits aux sein des métaheuristiques. La conception et la prédétermination de ces phases fut une première étape dans la mise en place d'un mécanisme de guidage.

La présente étude propose un mécanisme de guidage dynamique des phases d'intensification et de diversification. Ce contrôle est basé sur une mesure de l'exploration de l'espace de solutions. Les résultats sur un problème de tournées de véhicules avec fenêtres de temps (VRPTW) sont comparés avec le mécanisme où ces phases sont prédéterminées.

Ce mécanisme de guidage dynamique est aussi étudiée dans un environnement indépendant des méthodes de recherche afin de mieux en cerner les bénéfices.

Nous détaillons la méthodologie utilisée pour ensuite présenter la publication liée à ces travaux dans son intégralité. Finalement, suivent une synthèse et les perspectives.

4.2 Méthodologie

Le contrôle dynamique des phases de diversifications et d'intensifications selon une métrique mesurant la diversité d'exploration des solutions améliorantes doit se baser sur une métrique. La variation d'entropie des solutions de l'entrepôt est

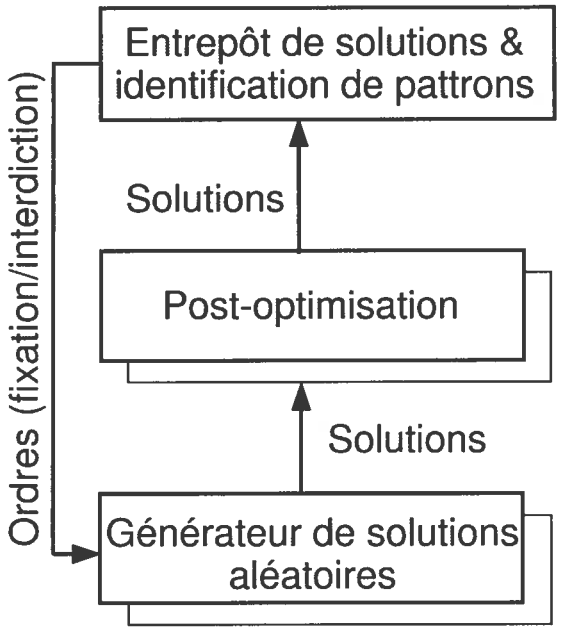

Figure 4.1: Recherche aléatoire guidée

utilisée dans la métrique choisie.

Dans une première phase et afin de s'abstraire des comportements spécifiques des métaheuristiques, nous remplaçons celles-ci par des méthodes de génération de solutions aléatoires tel qu'illustré en Figure 4.1. Cette méthode nous permet de bien cerner l'influence de la coopération et du guidage dynamique proposé. Nous en effectuons l'étude détaillée sur un sous ensemble de problèmes représentatifs du VRPTW.

Dans une deuxième phase, selon les conclusions de notre première phase, nous utilisons le guidage dynamique avec des métaheuristiques sur l'ensemble des problèmes du VRPTW.

Le cadre expérimental permet de suivre dans le temps l'évolution de la qualité des solutions. Nous mesurons aussi dans l'entrepôt de solutions, le nombre de solutions entrantes, la variation d'entropie des solutions et l'impact des valeurs des paramètres de pourcentage d'éléments fixés et interdits.

La publication présentée étudie, dans la première section, les mécanismes fondamentaux de la recherche coopérative et les avantages d'un guidage. Dans la

deuxième section, nous étudions l'influence du guidage et définissons une métrique simple basée sur l'entropie pour en mesurer son impact. Un mécanisme dynamique, basé sur le maintien d'une valeur idéale de cette métrique, est proposé dans la troisième section. Pour sa part, la quatrième section consiste en la description du cadre expérimental et des résultats. Quelques perspectives et conclusions font l'objet de la dernière section.

4.3 Publication 3 : Pattern-based dynamic guided cooperative search for the Vehicle Routing Problem with Time Windows

Alexandre Le Bouthillier
Département d'informatique et de recherche opérationnelle and
Centre de recherche sur les transports
Université de Montréal
C.P.6128, succursale Centre-ville, Montréal, QC, CANADA H3C 3J7
alexleb@crt.umontreal.ca

Teodor Gabriel Crainic
Département de management et technologie
Université du Québec à Montréal
C.P. 8888, Succursale Centre-Ville, Montréal, QC, CANADA H3C 3P8 and
Centre de recherche sur les transports,
Université de Montréal
C.P.6128, Succursale Centre-Ville, Montréal, QC, CANADA H3C 3J7
theo@crt.umontreal.ca

Peter Kropf
Institut d'informatique,
Université de Neuchâtel, Faculté des sciences
Rue Emile-Argand 11 CH-2007 Neuchâtel, Suisse
peter.kropf@unine.ch

4.4 Abstract

A guided cooperative-search mechanism identifies promising patterns in previously identified solutions. It *guides* the global search while controlling the diversification and intensification phases to solve combinatorial problems.

The mechanism is based on the solution warehouse strategy, in which several search processes cooperate by asynchronously exchanging information on the best solutions identified. Each of the independent processes implements a different search, a metaheuristic such as evolutionary algorithm, a tabu search, a random search or a post-optimization procedure. Patterns of elements in promising and unpromising solutions are identified and used to guide the search.

We propose an enhancement to the guided cooperative search, using a mechanism to orient the search by dynamically controlling the diversification and intensification phases. The variation in entropy of the solution warehouse is used as a metric to control these phases and to prevent premature convergence.

No attempt has been made to calibrate the individual search methods, the parallel cooperative method or the dynamic guidance mechanism. The results obtained on a set of selected test problems of the vehicle routing problem with time windows shows that the dynamically guided cooperative search performs better than other cooperatives methods. It identifies solutions of comparable quality to those obtained by the best methods in the literature and find one new best result.

Keywords : parallel cooperative search, dynamic guidance, entropy, vehicle routing with time windows.

4.5 Introduction

In recent years, parallel computing has evolved to increase both the efficiency and robustness with which combinatorial problems are addressed. Metaheuristics have been parallelized in the following three forms: division of compute-intensive tasks at a low algorithmic level, explicit domain decomposition of the solution or search space, and multithread search. Multithread searches are qualified as cooperative (Crainic and Toulouse [29]) when metaheuristic threads exchange information to create a global knowledge of the search space. This global knowledge allows information to be exchanged and can be used to guide individual searches towards promising areas, and away from unpromising ones. Cooperative searches have successfully been used to address several combinatorial problems: multi-commodity location-allocation (Crainic, Toulouse and Gendreau [30]), circuit partitioning (Aiex *et al.* [2]), and vehicle-routing problems with time windows (VRPTW)(Le Bouthillier and Crainic [77]).

More recently, Le Bouthillier et al. [79] showed that quality and robustness of the search can be improved even further with a static guidance mechanism. It was shown that the frequent and infrequent elements of good and bad solutions available in the solution warehouse were helpful in designing promising and unpromising patterns of solution elements. Imposing these patterns onto each metaheuristic constrains the search to the promising regions of the solution search space. The guidance mechanism provides control on the diversification and intensification phases by fixing or prohibiting specific element patterns in the solution examined. In this static guidance implementation, these phases were fixed and were not modified according to the behaviour of the global search. Nevertheless, results showed improved quality and robustness in almost all instances tested. While the methodology was evaluated for a specific set of problems, the guided cooperative search remains a generic concept that can easily be applied to various combinatorial problems that have a spatial structure.

Evolving from a static to a dynamic guided search is the key to adapt the

cooperation to the specific behavior of each set of metaheuristics and problem landscape in the search space. The goal of this work is to study the effect of the guidance mechanism in a cooperative framework. We investigate the influence on the global search of a metric-based control mechanism for the intensification and diversification phases. We claim that, by controlling these phases based on the quality and the entropy of the solutions in the solution warehouse (dynamic guidance) rather than in fixed phases (static guidance), we can provide a finer control of the global search, and thus, lead to a better solution quality. To our knowledge, no study controlling the diversification and intensification phases of the searches involved in a cooperative search has been carried out with the use of specific metrics of quality and population diversity. Usually these phases are controlled on an individual basis by the metaheuristics but not at a high level.

The main challenge in studying the effect of the guided search on the global search is obtaining an abstraction from any method-specific performance that generates noise in the interpretation. To address this issue and focus on the guidance mechanism and its impact on cooperation, we replace each independent metaheuristic with a random solution generator. The resulting generator threads receive instructions to fix and prohibit certain elements while generating solutions. These instructions are coming from work on the solutions sent to the guidance mechanism, but no solution is sent to the individual threads, which are thus influenced by the guided mechanism only.

We are using the Vehicle Routing Problem with Time Windows (VRPTW) to illustrate and test the concepts presented. To try to avoid biasing the results, we do not construct our guidance mechanism on routing solution elements, such as routes. Arcs, which are ubiquitous in network-based optimization problems, are used instead.

The solutions obtained with the guided cooperative method using random solution generators as individual search threads are reasonably good. We even improve the best result for one benchmark problem instance. This prompts us to feel confident that insights and conclusions drawn from the experiments presented in this

paper are applicable to general cooperative methods, as well as a broader set of problems and problem instances. We present, in fact, the convincing results of a dynamic guided cooperative search applied to the standard benchmark VRPTW problem instances.

To sum up, the contributions of this paper are, first, to provide a general model for cooperative guided search without specific calibration or customization. Second, we study its properties and behavior in a controlled environment that eliminates the possible "noise" introduced by particular meta-heuristics. Experimentation shows the model is sound, and that as dynamic guidance outperforms static guidance in solution quality and robustness with respect to varying problem characteristics.

In the first section, we recall the fundamentals of the cooperative search and the advantages of a guided mechanism. In the second section, we outline the bases for understanding the influence of a guided search and we define a specific metric for measuring its behavior. Furthermore, ideas for improving the current simple static guidance mechanism are suggested. The third section provides details on the dynamic guidance mechanism. The fourth section is devoted to the experimental framework. We evaluate the proposed dynamic guided search with a random solution generator to remove specific metaheuristic behaviors, and test the performance of the proposed guided mechanism with meta-heuristics large combinatorial problems. The proposed method is currently the best known method in average for standard benchmark problem of size 200,400 and 1000. Analysis and perspectives are the subject of the final section.

4.6 The cooperative search

This section recalls the basics of our implementation of a cooperative search. For further details, the reader may refer to previous studies (Le Bouthillier and Crainic [77], Le Bouthillier et al. [79]). Combining the specific behavior of various meta-heuristics helps orient the search toward promising regions of the solution space. By allowing various metaheuristics to exchange solutions through a solution ware-

house according to their internal logic increases the quality and the robustness of the search.

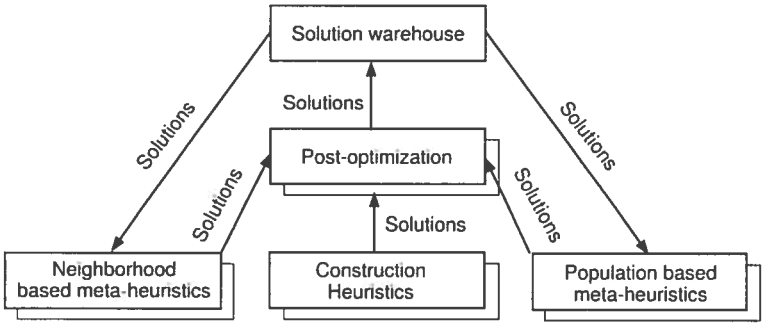

Figure 4.2: An abstract framework for a cooperative search

Figure 4.2 illustrates a simple cooperative search using different search methods. The arrows indicate a solution transfer and the boxes represent the search methods. The search methods in the lower part of the figure send solutions that are better than their current best to the post-optimization procedures. Solutions are then sent from post-optimization to a solution warehouse that holds and manages the pool of solutions. In turn, the solution warehouse will send, upon request, solutions to metaheuristics according to their internal logic. In our implementation, these solutions are selected in a probabilistic way toward the best. The solution warehouse orders the solutions with a modified objective function and eliminates the worst ones to make room for better solutions in case the warehouse is full.

No direct communications take place between search methods, thus reinforcing their independence and the asynchronous mode of exchange. This scheme makes the cooperation design simpler and, eventually, allows easy modification of the parallel system by adding new methods or dropping inefficient ones.

Cooperative searches have been applied to various problem types with great success, improving robustness, solution quality and speed-up (Crainic and Toulouse [29]).

4.7 The guided cooperative search

To increase search speed, robustness and solution quality, a guiding mechanism aims to restrict the search space to promising elements and to prohibit the search of unpromising ones. We can identify these promising and unpromising elements based on the frequency with which they appear in good and bad solutions.

A very simple and general pattern-based guided search definition is based on the inclusion or prohibition of promising and unpromising elements in given solutions found in these patterns.

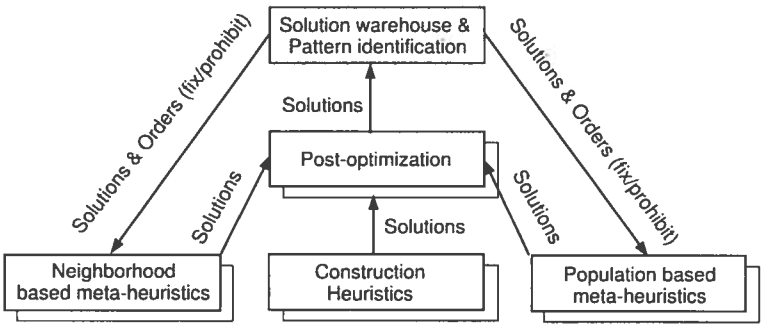

Figure 4.3: An abstract framework for a guided cooperative search

Figure 4.3 illustrates the additional orders to fix or prohibit solution elements that make up the guidance in a cooperative search. Adding a guided mechanism to a cooperative framework is relatively easy. In fact, Figure 4.3 is the same as Figure 4.2 except that i) the patterns need to be identified and evaluated and ii) the solutions that come from the warehouse are sent with additional orders to fix or prohibit solution elements.

The warehouse sends vectors of elements to be fixed and elements to be prohibited with each search method. This very simple method provides an effective method of inducing positive or negative feedback to individual methods, in a way that is similar to the communication mechanism used in self-organizing systems (Bonabeau et al. [10], Camazine et al.[15]).

The idea behind the global intensification and diversification is to increase the influence of the cooperation. By fixing or prohibiting elements from patterns of

promising or unpromising solutions in the individual search methods, we constrain their search space. Having the largest search space at the beginning of the search gives us a better idea of the solution search landscape. Terminating the search with a deeper exploration of promising search spaces follows naturally. Thus, we used phases of intensification at first, followed by diversification.

In a previous study (Le Bouthillier et al. [79]), a simple pattern-identification mechanism was used to fix or prohibit promising and unpromising element patterns in the exploration of individual metaheuristics. In the following, we recall the basic mechanism of this pattern identification and the way in which it was used. Prior to that, however, we outline how we define a pattern.

4.8 Pattern definition

Consider the frequency of inclusion of arcs in a given subset of the solution warehouse. In particular, this subset may be the entire population, an elite (e.g., with solution in the 10% best), average (between the 10% and 90% best) or worst (the last 10%) group of solutions. An element with a high frequency in a given group signals that the meta-heuristics participating to the cooperation have often produced solutions that include that element. Tagging solutions to identify the last algorithm that sent it, one may induce similar information by subset of participating algorithms as well. When the frequency of inclusion of several elements is considered, patterns emerge among the solutions of the solution warehouse or the specific group examined.

We define a pattern relative to a subset of solutions as a vector containing the elements with the highest or lowest frequency of appearance among the elements of the given solution subset. The length of the pattern can be n = `1, 2, ... maximum number of elements`. A frequent (infrequent) pattern relative to a set of elements is built of elements with high (low) frequency of appearance in the solutions of the set. High (low) frequency elements are selected sequentially in decreasing order from the highest (increasing from the lowest) frequency value.

We may select patterns from specific sub-populations (e.g., elite, average, and worst) and compare the rate of appearance of a specific pattern between them. From these patterns, we can guide the search by fixing or prohibiting, in the search methods, frequent or infrequent elements that appear in the solutions pool. Fixation is achieved by imposing the elements to be included in any solution that is produced by the search methods, and prohibition is achieved in the opposite manner.

A pattern of length n is defined as an element subset of cardinality $n = 1, 2, 3, ...$ to the maximum number of elements that form a solution definition. In Le Bouthillier et al. [79], the number of elements to include in a promising pattern were selected in decreasing order of incidence for promising patterns, and in increasing order of incidence for unpromising ones. The pattern selection was made within sub-populations (Good: 10% of solutions; Average: 80%; and Bad: 10%). It is thus possible to find promising or unpromising patterns by comparing the statistical occurrence of a pattern in these three populations. Figure 4.4 depicts the five possible scenarios with pattern occurrences.

1. Pattern occurs less frequently, on the average sub-population, than in the Bad or Good sub-populations;

2. Pattern occurs more frequently, on average, than in the Bad or Good sub-populations;

3. A pattern that appears consistently in all sub-populations;

4. Increasing occurrence of the pattern when the solutions decrease in quality;

5. Decreasing occurrence of the pattern when the solution increase in quality.

Increasing rates of occurrence for a specific pattern, when moving from good to worse solutions, are an indication that this pattern has in the past led to unpromising solutions (see red dashed line U - number 4 in Figure 4.4); it is thus an unpromising pattern. Conversely, decreasing rates of occurrence for a pattern when

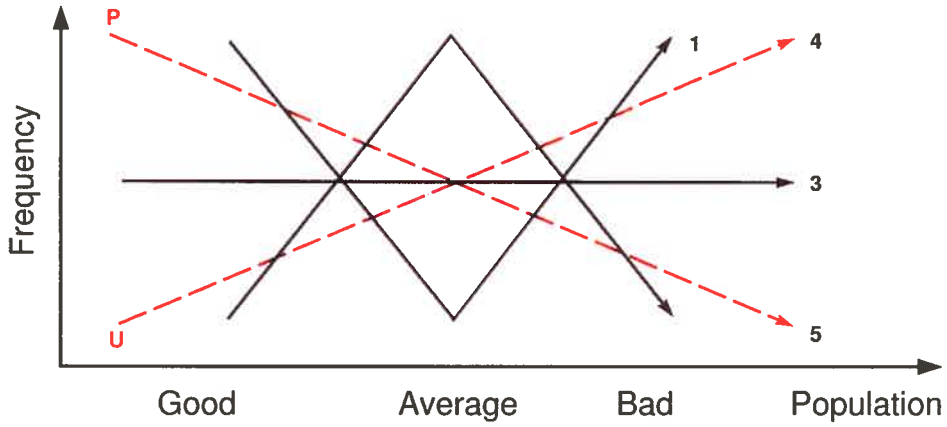

Figure 4.4: Occurrence of a pattern

moving from good to worse solutions is an indication that this pattern leads to good solutions; it is, therefore, a promising pattern (see red dashed line P - number 5 in Figure 4.4). These situations represents strong patterns as they clearly show the influence of the pattern on the quality of the solutions.

4.9 Pattern-based diversification and intensification

As shown in the guided cooperative search (Le Bouthillier et al. [79]), fixing and prohibiting element patterns can intensify or diversify the search. To intensify the search around solutions with good elements, we fix a number of elements from a promising pattern of the population in each individual search. Conversely, prohibiting these arcs for several iterations diversifies the search relative to the current set of good element values. Symmetrically, given an unpromising pattern of length n, fixing the element of that pattern intensifies the search in unpromising regions, while prohibiting it diversifies the search.

Not all patterns correspond to sets of elements appearing in at least one of the solutions of a given set (e.g., the subset used to generate the pattern or even the entire solution warehouse). We define an *in-pattern* as a pattern that actually appears in at least one solution of the subset of solutions considered, as opposed to a *statistical pattern* that does not appear in any solution of the subset. A statistical

pattern is thus only a consequence of the statistical process of accounting for the frequency of individual elements.

Consider an *in-pattern* of length n common to the three sets of elite, average, and worst solution groups. Two meaningful situations can occur with respect to the frequency of appearance of the pattern in these sets as one move from the elite to the average to the worst subpopulation: The frequency is either increasing or it is stable or decreasing. We call the *in-pattern unpromising* in the first case and *promising* in the latter.

Promising and unpromising patterns may then be used to constrain for a certain time the search space of particular methods in the cooperation and thus induce global intensification and diversification phases, as described in the next subsection.

For the present study, we will accept patterns that are not necessarily in-patterns, but we will ensure that the proposed patterns are feasible, i.e., that at least one feasible solution can be constructed from each one. This simple enhancement can facilitate the creation of new solutions that are based on the statistical occurrence of good diverse elements not necessarily related to any previous solution.

The guided cooperative search of Le Bouthillier et al. [79] used four-phase approach to provide diversification and intensification.

Choosing a pattern from various sub-populations of good, average or bad solutions allows the global search to diversify or intensify using their respective patterns. To diversify the search, we prohibited elements and to intensify, we fix elements in the individual search methods. We vary the length of the pattern during these phases to increase or decrease the intensification/diversification effect. The four phases are as follows:

1. Diversification: Prohibiting elements from real patterns of decreasing length from the average sub-population;

2. Diversification: Prohibiting elements from real patterns of decreasing length from the bad sub-population;

3. Intensification: Fixing elements from real patterns of increasing length from the average sub-population;

4. Intensification: Fixing elements from real patterns of increasing length from the good sub-population.

These phases were successfully applied to the vehicle problem with time windows (100-node benchmark problems from Solomon [93] and 200 to 1000-node benchmark problems from Homberger and Gehring [64]).

This guidance mechanism is very intuitive and simple: fix or prohibit elements in the solution space in each solution generator. We claim that by replacing this static guided search by a dynamic guided search, we will have better control over the diversification and intensification phases.

4.10 Dynamic guided mechanism

To provide a dynamic guidance mechanism, we need to identify a metric that will measure the evolution of the search, and upon which we can act. The desired characteristic of this metric is to be able to measure if the exploration has been diverse and thus if the solutions found have similarities or not. The variation in the entropy of the population in the solution warehouse seems an appropriate measure, as it measures the different number of solution elements in that population.

The intensification or diversification phases have a clear influence on the entropy. Hopefully, by controlling the entropy, we can reduce the premature convergence of the search. The metric used is based on the entropy and the quality of the solutions in the solution warehouse.

We shall, in the following define the metric and how to control the evolution of the search based on the change of its value. A series of experiment will help us to understand its influence.

We propose a new guidance mechanism that will act upon the variation of the entropy population value to specify the intensification and the diversification

phases. We also consider the relative influence that a particular element can have onto a solution and thus its utility. Before moving on to defining the entropy, we first define the concept of the utility of an element.

4.11 Entropy-based guidance

Entropy is a concept in thermodynamics, statistical mechanics and information theory. The concept of population entropy as a measure of population diversity in an evolutionary algorithm context was introduced in Capacarrère et al. [16]. We use the number of different elements in the population as a measure of entropy. If the number of solutions in the warehouse is N_{rep} and the size of the problem in number of nodes is N_{size} then the entropy S is

$$S = \sum_{i \in elements} p_i \ln p_i \tag{4.1}$$

where "elements" is the set of distinct elements from the solutions in the warehouse, and p_i is the proportion of all elements in the solutions ($N_{size}N_{rep}$ of them) that are element i.

The maximum entropy occurs when all elements of all solutions are distinct, i.e.,

$$S_{max} = \ln N_{size}N_{rep} \tag{4.2}$$

And the minimum occurs when all solutions are identical, i.e.,

$$S_{min} = \ln N_{size} = S_{max} - \ln N_{rep} \tag{4.3}$$

Solutions will be diverse, but not necessarily of better quality at first. It might be a good choice to replace "good" solutions with more diverse solutions that are "almost as" good. This again could be controlled by a parameter that would regulate the balance between the diversity and quality of what goes into the population.

The warehouse will update the solution pattern and the utility function of each

of its elements for all solvers as soon as a new solution enters the warehouse. Each solver converts the utility function into a probability function to bias route choices for each node. For example, the selected random generator chooses an element with probability 0.5(u+1), where u is the utility of the new element being added.

We start by fixing a percentage of promising elements and prohibiting a percentage of unpromising ones. We then look at more dynamic mechanisms, based on the entropy of the population and the utility function of each element, to fix and prohibit them.

In the following we propose a method to remove the specific behavior of the metaheuristic, allowing us to focus on the impact of cooperation and the guidance mechanism. Replacing metaheuristics with a random search allows for a faster exploration of the search space without any specific search pattern.

4.12 Relative utility function

We define the utility of an element (or a series of elements) by its influence, positive or negative, on the solutions that use it. We can measure this utility with different factors and normalize its value within $[-1, 1]$. For example, a value of -1 means that, when present, this element always degrades the solution quality, and, a value of 1 means that it always increases it. Thus, each element that is part of a pattern can have a probability of being fixed or prohibited based on its utility value.

The relative utility of each element is defined in the following. The elements are first ordered according to their frequency of appearance within the population. Based on the average value of solutions in which they respectively appear, they are given a proportional value within $[-1, 1]$.

This utility will be used to bias the choice of elements instead of fixing or prohibiting them.

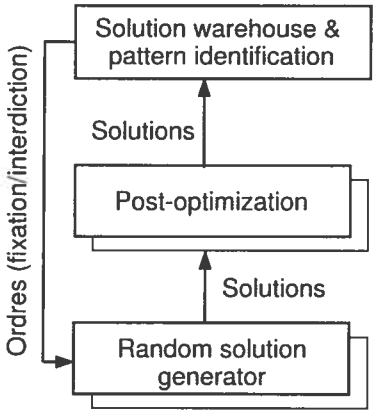

Figure 4.5: An abstract framework for a guided random cooperative search

4.13 Random solution generator

The main challenge in studying the effect of the guided search on the global search is obtaining an abstraction from any method-specific performance that generates noise in the interpretation. To address this issue and focus on the guidance mechanism and its impact on cooperation, replace each independent meta-heuristic with a random solution generator as shown in Figure 4.5.

To better understand the impact of the guided mechanism, we were looking for a solution generator with the following property :

- No memory;
- Be easily and repeatedly instantiated with non-specific behaviors;
- Not specific to a particular problem type.

It appears that a random solution generator, while providing average-quality solutions, was the best generator to fully test the guidance mechanism without inducing any quality bias, as the other "search" methods would have done. We will thus use multiple copies of a random generator (with different random seeds), which will send random solutions to the warehouse and receive instruction to"fix" and "prohibit" elements.

4.14 The vehicle routing problem with time windows

The following describes combinatorial problem we selected to conduct our experiment: the vehicle routing problem with time windows.

The VRPTW is a well-known and extensively studied combinatorial problem and thus well suited for benchmarking. The single-depot VRPTW assumes a set of customers with known positive demands and specific time intervals during which service is available. A fleet of homogenous vehicles of known capacity resides at a given depot to perform the service. The objective is to find a set of closed routes (or tours) that start and end at the depot during its opening hours, while also

- Minimizing the total cost of performing the service;
- Ensuring that customers receive exactly one service during their specified time window;
- Avoiding overloading vehicles.

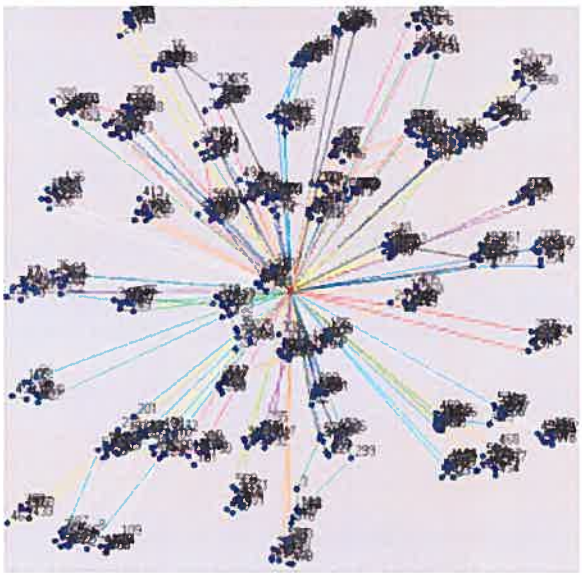

Figure 4.6: Best known solution for problem C1-6-4

Figure 4.6 shows the best-known solution for problem C1-6-4, which involves 600 clustered customers. The depot appears at the centre of the image. The

different routes for serving customers are shown in different colors.

Comparison and evaluation of the proposed mechanism will be done experimentally, with a selected set of 1000-node vehicle routing problems with time windows (VRPTW) from the Homberger and Gehring benchmark set [64]. Without any loss of generality, the entire concept proposed in this paper can be applied to various combinatorial problems. In terms of VRPTW, the elements that compose a solution are arcs.

Tests have been carried out on the standard set of test problems proposed by Solomon [93] and used by all authors. The set contains 56 problems of 100 customers each. We also used the extended set produced by Homberger and Gehrin [64] with 300 problem instances that vary from 200 to 1000 customers.

The standard and extended problems are divided into six categories, named C1, C2, R1, R2, RC1, and RC2. For all problem instances, customers are distributed in a [0,100] square unit. The customers in sets C are clustered together, while those in sets R are distributed randomly. Problems in sets RC combine the two characteristics. Time windows at the depot are relatively small for problems of type 1, to allow less customers to be served by each route; time windows are larger for problems of type 2. The service time is of 10 units by customer for problems of type R and RC, and of 90 units for class C.

To illustrate our findings, we have selected a problem set of six standard problems, one from each class, denoted C11010, C21010, R11010, R21010, RC11010 and RC21010. Experiment on the dynamic guided search with metaheuristics will be performed on all 356 problems.

4.15 Experimental framework

The goal of using a random generator was to be able to better understand the guided mechanism. Surprisingly, there are many ways to generate a random solution.

The experiment objectives are to reveal the impact of a controlled entropy on the convergence of the search and to show the quality of the solution produced by

a dynamic guidance mechanism.

The methodology is twofold. First, we create a guided search that combines the utility of the elements and the control of the entropy to form the implementation of Figure 4.3. Second, we disregard any particular behavior of specific metaheuristic with random searches. This will allow the comparaison of a pure random search (without any guidance mechanism) to a guided random search. Putting it all together, the new dynamic guided mechanism is incorporated into a framework with metaheuristics.

Our new dynamic guided search with tabu search and genetic algorithm produces new best average results for the problem of size 200, 400 and 1000 in the standard benchmark of VRPTW.

In the following, we explain our choice of a random solution generator, the influence of the guidance parameters, the impact of arc fixation and prohibition. Follows, the performance analysis of a pure random vs a guided random search. Finally, performance analysis of a static and a dynamic guided search when used with metaheuristics are illustrated over the 200 to 1000 standard VRPTW benchmarks.

4.15.1 Choosing a random solution generator

While we take into consideration various methods of generating random solutions, we work with only one, since it serves its simple purpose. Even if the selected random generator has the lowest performance (in time $O(n^2)$ and memory $O(n)$) of all the generators we designed, it can generate a solution closest to the number of vehicles requested, r. The random generator named Create&Add creates r routes and tries to randomly add the non-visited nodes. If this addition is impossible for all of the routes, a new route is created with the node in question. The pseudo-code for this simple random generator is shown in Algorithm 1.

- Line 1 creates a permutation of the n elements of a solution RandN;
- Line 3-5 create r routes and add a random element i from the solution RandN;

Algorithm 1 Create&Add

```
1: RandN ← Permutation of N
2: R ← φ
3: for i = 1..r do
4:    R = R⋃RandN_i
5: end for
6: for i = r+1..n do
7:    rAdd = 1,2,..|R|
8:    while |rAdd| ≠ 0 do
9:       rCour ← a route of rAdd
10:      if RandN_i can be added to rCour then
11:         add
12:         break
13:      else
14:         rAdd ← rAdd − rCour
15:      end if
16:   end while
17:   if rAdd = φ then
18:      R = R⋃RandN_i
19:   end if
20:   return R
21: end for
```

- Line 6-7 iterate through the elements that are not in the solution yet;
- Line 8-16 try to add each element to each route;
- In line 18, the element that can't be added will form a new route.

Create&Add is not very efficient, in time $o(n), O(n^2)$ and space O(n), but it does find a solution close to the requested number of vehicle, r. It is worth mentioning that random solutions are not to be compared with state-of-the-art methods (about 2.5 times more vehicles on average and twice the distance) since our purpose is only to study the added value of the guided mechanism in an independent fashion.

4.15.2 Initial parameter exploration

To provide a self-adapting mechanism, we need to identify the impact of each parameter. The parameter exploration consists in a series of run-throughs of all the possible combinations of parameter values within a certain increment to study their impact on the search. The methodology proposed here is to perform a parameter exploration on six selected problems with the random search only. We report the best solution found after a sufficient amount of CPU time, to show the effect of the parameters. To limit the number of experiments, we performed a parameter exploration with a variation of 0.10 for the 4 parameters studied. Subsequent experiments within a variation of 0.005 on promising experiments gave us a better idea of the parameters' influence.

Our approach considered the 4 parameters and the following steps:

1. Performing a parameter exploration to determine the best value of the algorithm's parameters: the fraction of the warehouse to be considered good solutions (goodfrac);
2. Defining the fraction of the elements of these solutions to be considered promising (good_arc_frac) and symmetrically for the worse solutions (badfrac), the fraction of unpromising elements (bad_arc_frac);

3. Determining the effect of entropy control on preventing premature convergence of the solution warehouse .

Our tests revealed that less than 300 seconds of 4 CPU time was sufficient to draw conclusions on the effect of the parameter values.

Figures 4.10- 4.21 show the distance and the numbers of vehicles, respectively. With the random generator, they both decrease until a plateau is reached for a typical problem. When the run starts, solutions quickly enter and fill the warehouse. After this ramp-up period, the randomly generated solutions are not providing improved solutions to the warehouse, where the solutions are converging towards a specific number of vehicles and a travel distance. With the guided random search, the number of vehicles and travel distance is reduced at regular interval when the search space is constrained.

Figure 4.7 shows the results for problem instance C11010 as a function of good_frac and good_arc_frac. The best solutions were found for $good\_frac = 0.245$ and $good\_arc\_frac = 0.025$. However a similar study for problem instance C21010 (Figure 4.8) shows a completely different set of values, and in fact the same setting of good_frac and good_arc_frac was not good for all problems. This figure plots the number of vehicles for the best solution found within 300 seconds of CPU time, as a function of good_frac and good_arc_frac for the C11010 problem.

Figure 4.8 plots the number of vehicles for the best solution found within 300 seconds of CPU time, as a function of good_frac and good_arc_frac for the C21010 problem.

The effect of arc prohibition ($bad\_frac > 0$ and $bad\_arc\_frac > 0$) was entirely negative when used for a complete run. Figure 4.10- 4.21 show the results when there is no prohibition (plain red line). With any form of prohibition, the best solution remained at the plateau level (around 228 vehicles in the C11010 case).

Each problem has a different solution landscapes and the impact of the parameters values has been clearly shown. The goal is to find these ideal values.

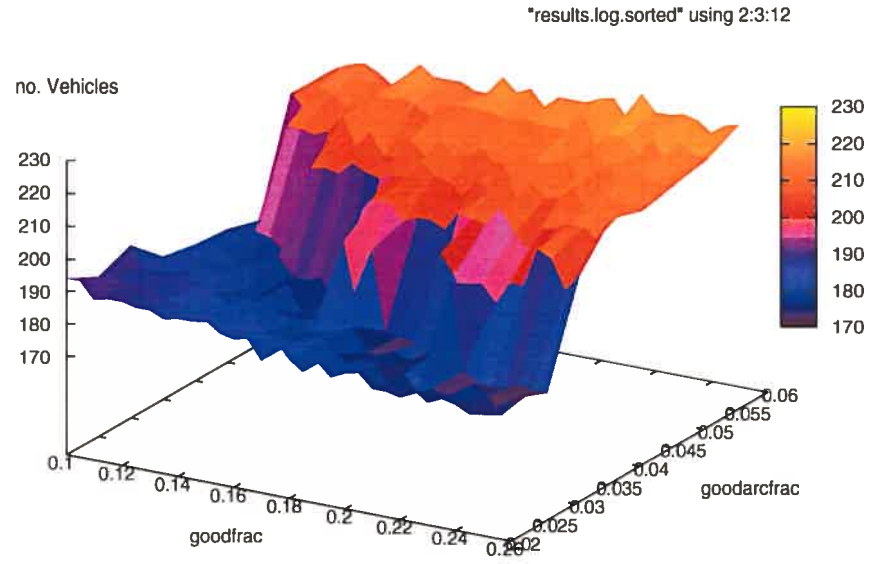

Figure 4.7: Plot of nb. of vehicles for best solution found within 300 seconds CPU time, as a function of good_arc and good_arc_frac for the C11010 problem

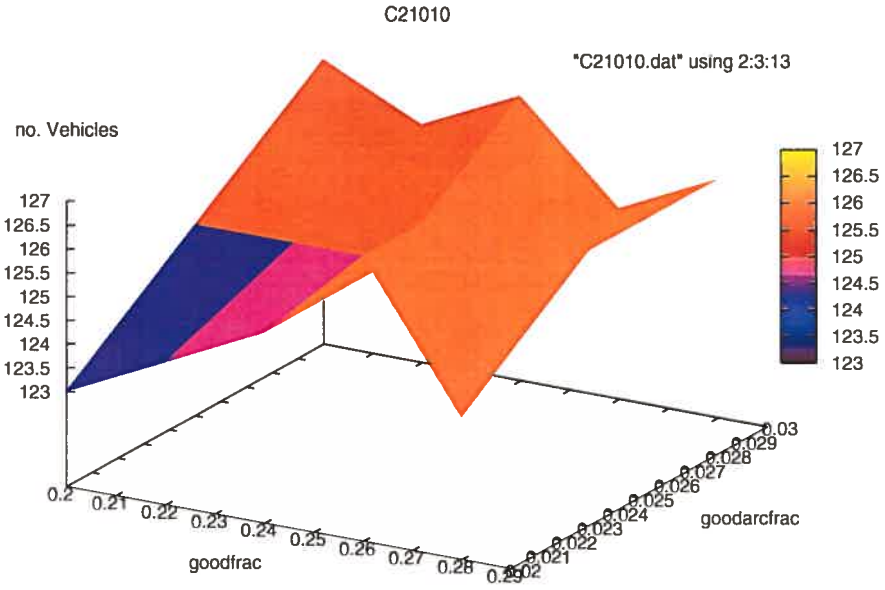

Figure 4.8: Plot of nb. of vehicles for best solution found within 300 seconds CPU time, as a function of good_arc and good_arc_frac for the C21010 problem

4.15.3 The effects of arc fixation and prohibition

Since there was no ideal value for fixation or prohibition that was appropriate for all problems, we performed an experiment to see what were the effects of these parameters.

In this dynamic algorithm, we set good_frac and bad_frac equal to 1.0 and vary only the arc_frac parameters. These settings are meaningful because when the warehouse's entropy falls, these parameters will have little effect, as all solutions in the warehouse become similar.

Without any arc fixation or prohibition, the best solution remains poor, both in number of vehicles and in length. The warehouse's entropy remains high. The algorithm does not find a good solution and, once the warehouse is full, the rate of new solutions entering the warehouse drops.

As soon as a minimal amount of fixation is introduced (0.001 in the case of a 1000-customer VRPTW problem), new solutions begin to enter the warehouse, the best solution improves dramatically and the entropy starts to fall. Of course, these solutions are mostly hybrids of existing solutions in the warehouse, keeping the most successful partial routes. Thus fixation acts a little bit like crossover in a genetic algorithm by preserving hopefully good elements.

After some time, the number of solutions entering the warehouse drops, and the algorithm again stagnates. Increasing the level of fixation restarts the algorithm. Eventually, however, the entropy drops too low, and the algorithm converges prematurely.

This interactive study shows that:

1. Arc prohibition has significantly less impact than arc fixation since few newly produced solution is able to enter in the warehouse and influence the cooperation;

2. The ideal values for fixing and prohibiting arcs depend on the problem and the entropy;

3. Fixing an arc is equivalent to prohibiting all other exiting or entering arcs of a node.

Therefore, to get prohibition and guidance to work properly, we need to:

1. Prune the warehouse by throwing away a portion of the warehouse, as otherwise no new solutions can compete. Our tests show that removing 50% of the solutions when no solution has entered in the warehouse for a certain number of iterations;

2. Use different arc fixation and prohibition proportions as the entropy changes.

4.15.4 Performance analysis of a guided search

These two modifications were implemented in the proposed dynamic guided search as shown in the pseudo-code of Algorithm 2, where count returns the number of solutions that entered the warehouse since the last iteration.

The main idea of the Dynamic Guided Search is to increase or decrease the control of the arc fixation and prohibition based on the quality of the solution in the pool and their diversity.

We refer the reader to section 4.15.2 for a definition of goodfrac, badfrac, good_arc_frac, bad_arc_frac used in Algorithm 2.

- Line 1 indicates that the guidance needs to be modified when no solution has entered the warehouse for a certain amount of time;

- Line 2-4 increase the fraction *bad_arc_frac* by *daf* when entropy is below a threshold, and when the fraction of elements in the bad solutions (*bad_arc_frac*) that are considered unpromising is below 0.5;

- Line 6-8 increase *good_arc_frac* by *daf* if it is below 0.5;

- Line 9-11 decrease *bad_arc_frac* by *daf* if it is above 0 (but below 0.5).

Algorithm 2 Dynamic guided search

```
1: if count < thresh * utility_interval then
2:   if entropy < entropy_thresh AND bad_arc_frac < 0.5 then
3:     bad_arc_frac ← bad_arc_frac + daf
4:     Return
5:   else
6:     if good_arc_frac < 0.5 then
7:       good_arc_frac ← good_arc_frac + daf
8:     end if
9:     if bad_arc_frac > 0 then
10:      bad_arc_frac ← bad_arc_frac − daf
11:    end if
12:  end if
13:  if count = 0 then
14:    Return
15:  end if
16: end if
```

Each time the entropy reaches a certain threshold, we increase the fraction of good arcs that are fixed (*good_arc_frac*) by a small increment (*daf*) and decrease the fraction of bad arcs that are prohibited (*bad_arc_frac*) by the same small increment. We found that using a threshold *thresh* of 1.3*Smin* of the entropy to introduce prohibition performs better than other values. Once the warehouse is refilled, and the new solution rates fall again, prohibition is turned off, and the algorithm is iterated with fixation. When entropy drops below the threshold, prohibition is performed again.

We also modify the acceptance criteria of the solution warehouse to take into account not only the quality of the solutions but also their diversity. Solutions can enter the warehouse only if they differ by δ elements or improve the best. δ is divided by two every 50 updates of *good_arc_frac*.

In the following, we display results of a random guided search experiment on the 6 selected problems.

Table 4.1 shows the best solution found for a pure random run (no guiding mechanism as shown in Figure 4.9) versus a dynamic guided random search (as shown in Figure 4.1). The stopping criterion was set to 7200 seconds of 8 CPU

	Pure random			Dynamic random				
Problem	T(s)	nbVeh	Length	T(s)	nbVeh	Lenght	nbVeh	Lenght
C11010	129.07	228	217355.09	865.54	161	125957,37	142%	173%
C21010	832.15	124	223948.81	878.62	86	154105,83	144%	145%
R11010	143.19	236	233032.50	790.56	148	145954,09	159%	160%
R21010	169.81	148	260560.31	877.72	84	195668,047	176%	133%
RC11010	235.25	222	198574.81	876.09	119	112494,60	187%	177%
RC21010	595.37	138	242201.20	801.48	66	160751,64	209%	151%
						Average	170%	156%

Table 4.1: Pure random vs. guided random search

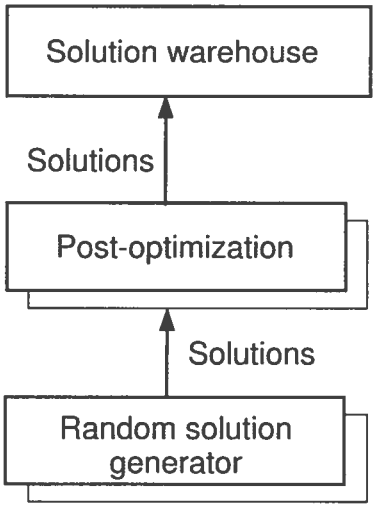

Figure 4.9: An abstract framework for a pure random search

time.

We could have stopped the run at 2500 seconds in all cases as the dynamic guidance on a random search shows a clear advantage over the pure random search. The average improvement for the number of vehicles is 170% and for the total travel length, 156%. The dynamic random alone can't compete with current best methods but produces average solution very easily.

In Figures 4.10- 4.21, as soon as the warehouse is filled and the fixation and prohibition orders are given, we see that the solutions generated by the guided random search (dotted green line) starts to improve compared to the pure random search (plain red line). The random search (as expected) converges quickly and is not able to generate good solutions in the provided 7200 seconds of 8 CPU time.

We plotted the following quantities as a function of time: best solution, num-

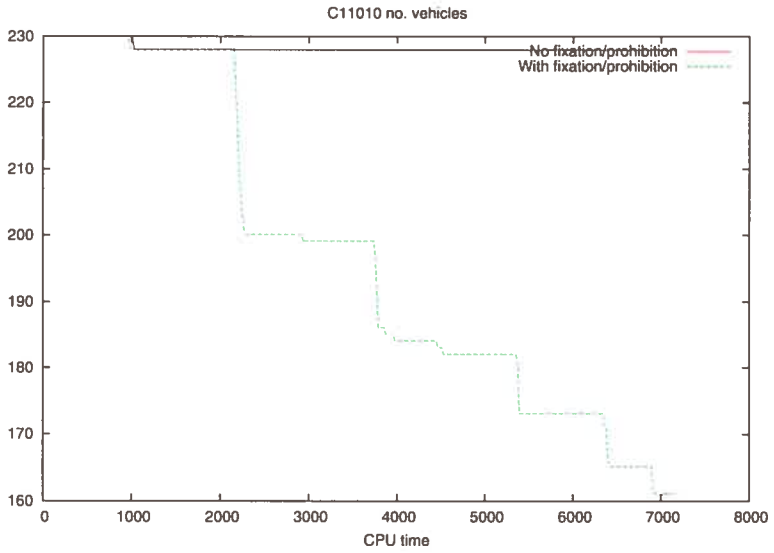

Figure 4.10: Pure random vs guided random search in number of vehicle - time for C11010

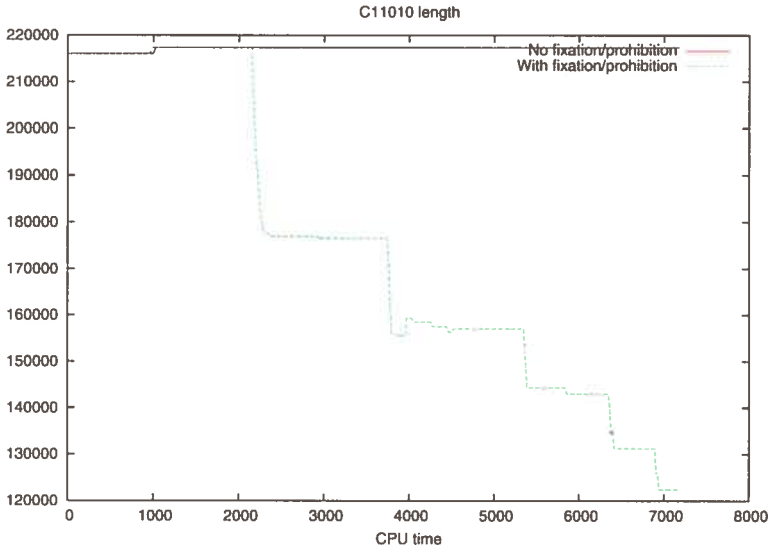

Figure 4.11: Pure random vs guided random search in tour length - time for C11010

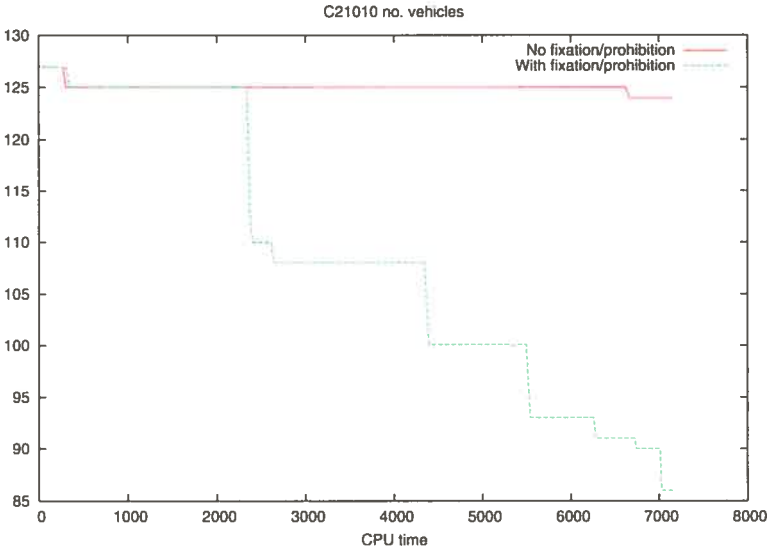

Figure 4.12: Pure random vs guided random search in time for C21010

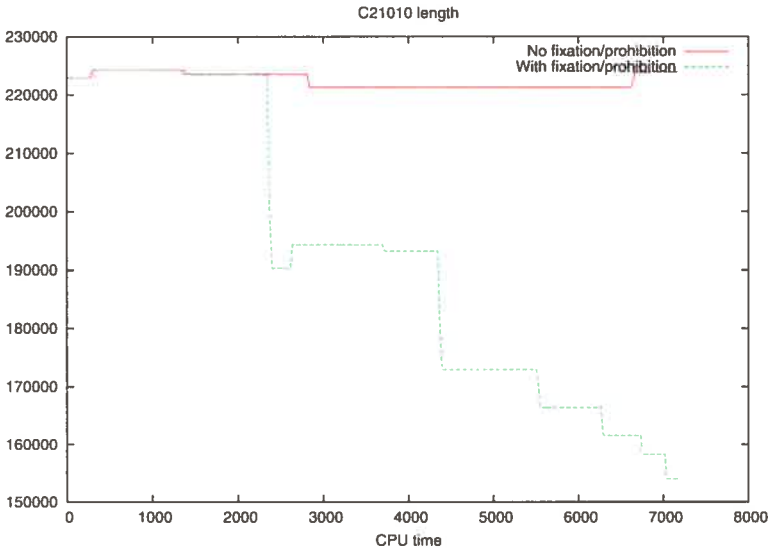

Figure 4.13: Pure random vs guided random search in time for C21010

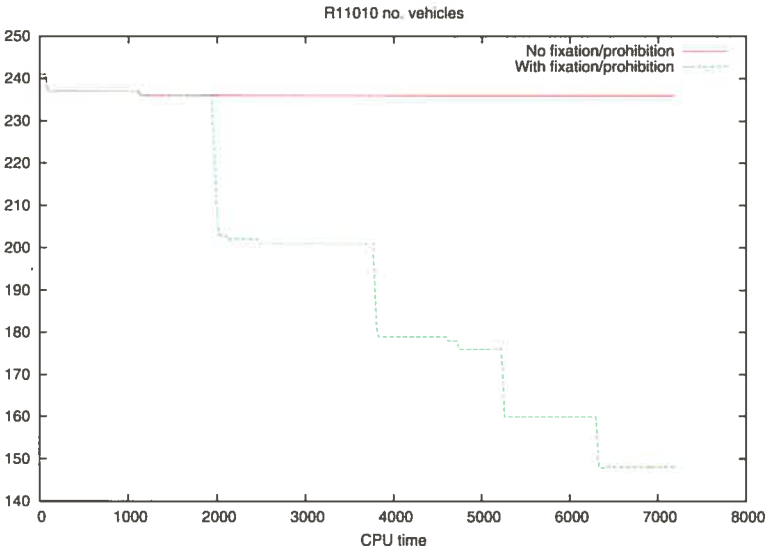

Figure 4.14: Pure random vs guided random search in time for R11010

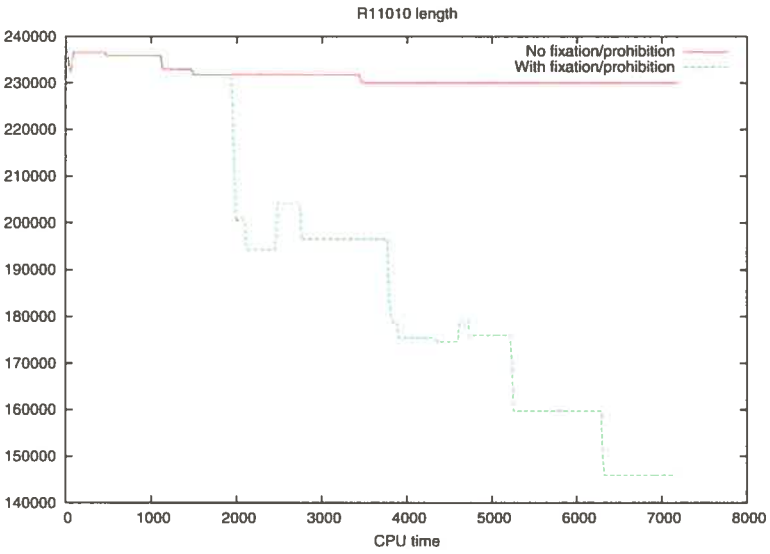

Figure 4.15: Pure random vs guided random search in time for R11010

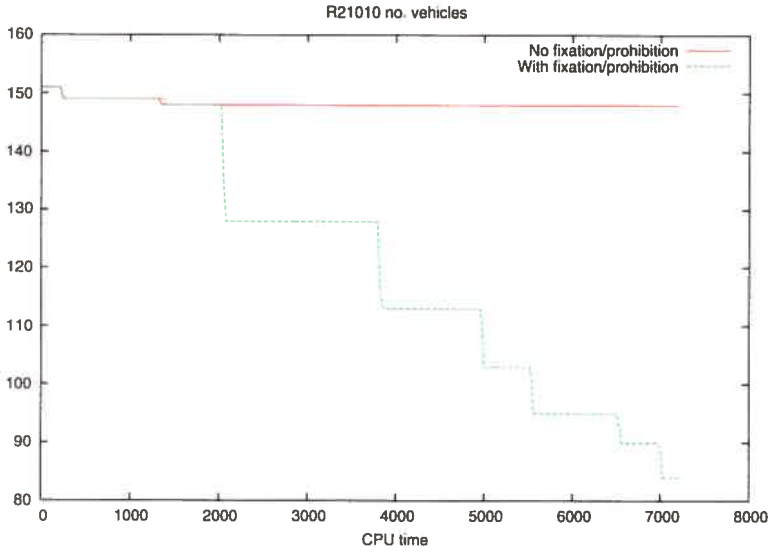

Figure 4.16: Pure random vs guided random search in time for R21010

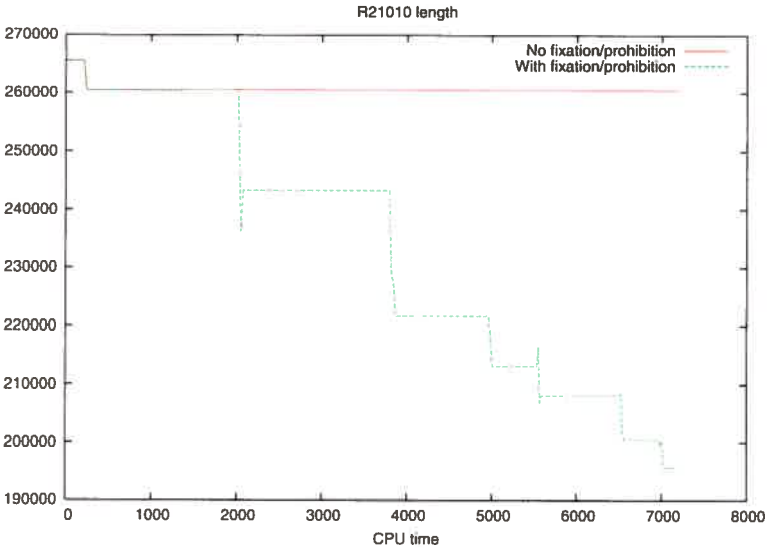

Figure 4.17: Pure random vs guided random search in time for R21010

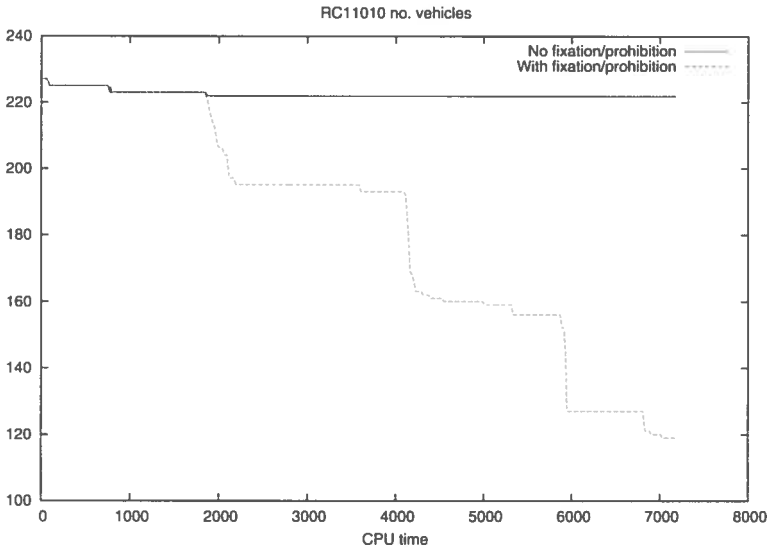

Figure 4.18: Pure random vs guided random search in time for RC11010

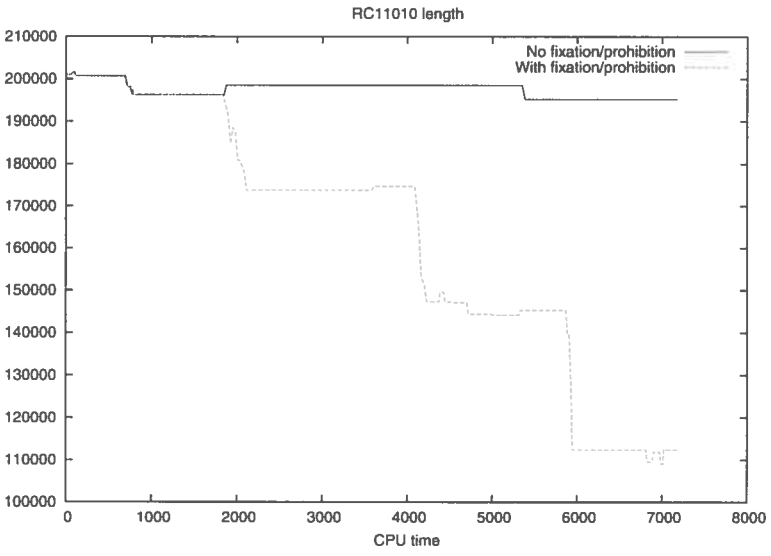

Figure 4.19: Pure random vs guided random search in time for RC11010

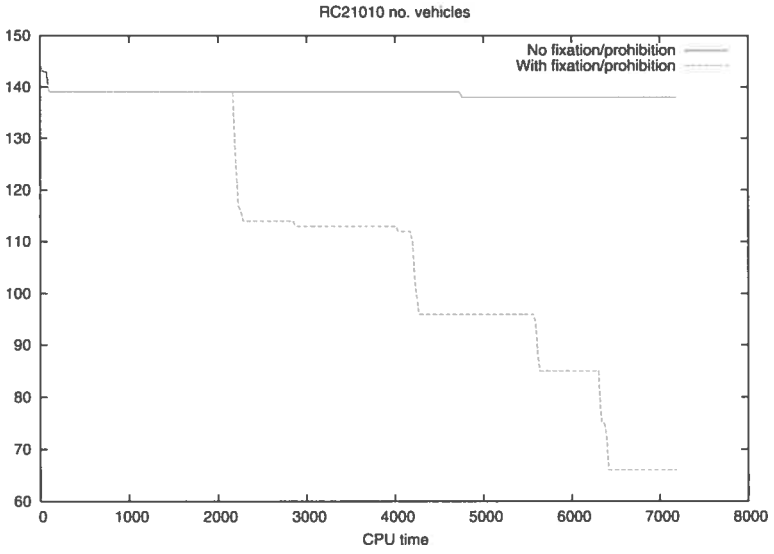

Figure 4.20: Pure random vs guided random search in time for RC21010

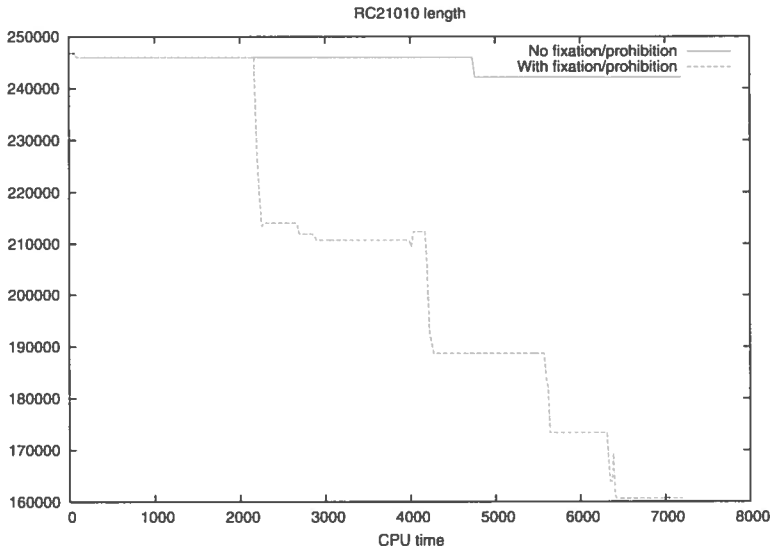

Figure 4.21: Pure random vs guided random search in time for RC21010

ber of vehicles, length of tour, entropy and the number of solutions entering the warehouse. Several plots of these quantities and of fixation and prohibition (good_arc_frac and bad_arc_frac respectively) for the proposed dynamic guided search algorithm are shown in Figures 4.10-4.21, which illustrates the positive impact of the dynamic guidance.

Figure 4.22 plots the evolution of the best solution for the dynamic guided search for a typical run (C11010). Entropy control and post-optimization are used on this run.

The proposed dynamic guided search performs better than the simple random search. As shown, even using the random search, the quality of solutions improved with the guided mechanism. Using prohibition and fixation, we are also able to control the entropy and to influence the number of new best solutions entering the warehouse, as shown in Figure 4.22 (Solutions Entering Repository) which increases when we increase the entropy in the diversification phase.

Between a time of about 50 and 200 seconds, the number of arcs that are fixed is slowly increased, thus allowing a better solution to be found (recall the use of a random search here). By fixing and prohibiting, and by removing 50% of the solutions in the warehouse before the prohibition, we postpone convergence of the warehouse until a time of 450 seconds. This improves the quality of the overall best solution found, when compared to the results of a search with no dynamic guidance.

Each time the warehouse is emptied of 50% of its solutions (500 solutions out of 1000 are removed in the experiments), the entropy rises and new solutions can enter the warehouse. In the plot of solutions entering the solution warehouse, in Figure 4.22, every peak goes above 500 solutions entering the warehouse thus replacing previous good solutions. This is a clear indication that prohibition allows the finding of new best solutions, thereby diverting the random creation of solutions from unpromising solutions.

With the random guided search, a new best result for problem C1_2_1 is found as shown in Figure 4.23 and Figure 4.24 : 20 vehicles, for distance of 2639.60. As

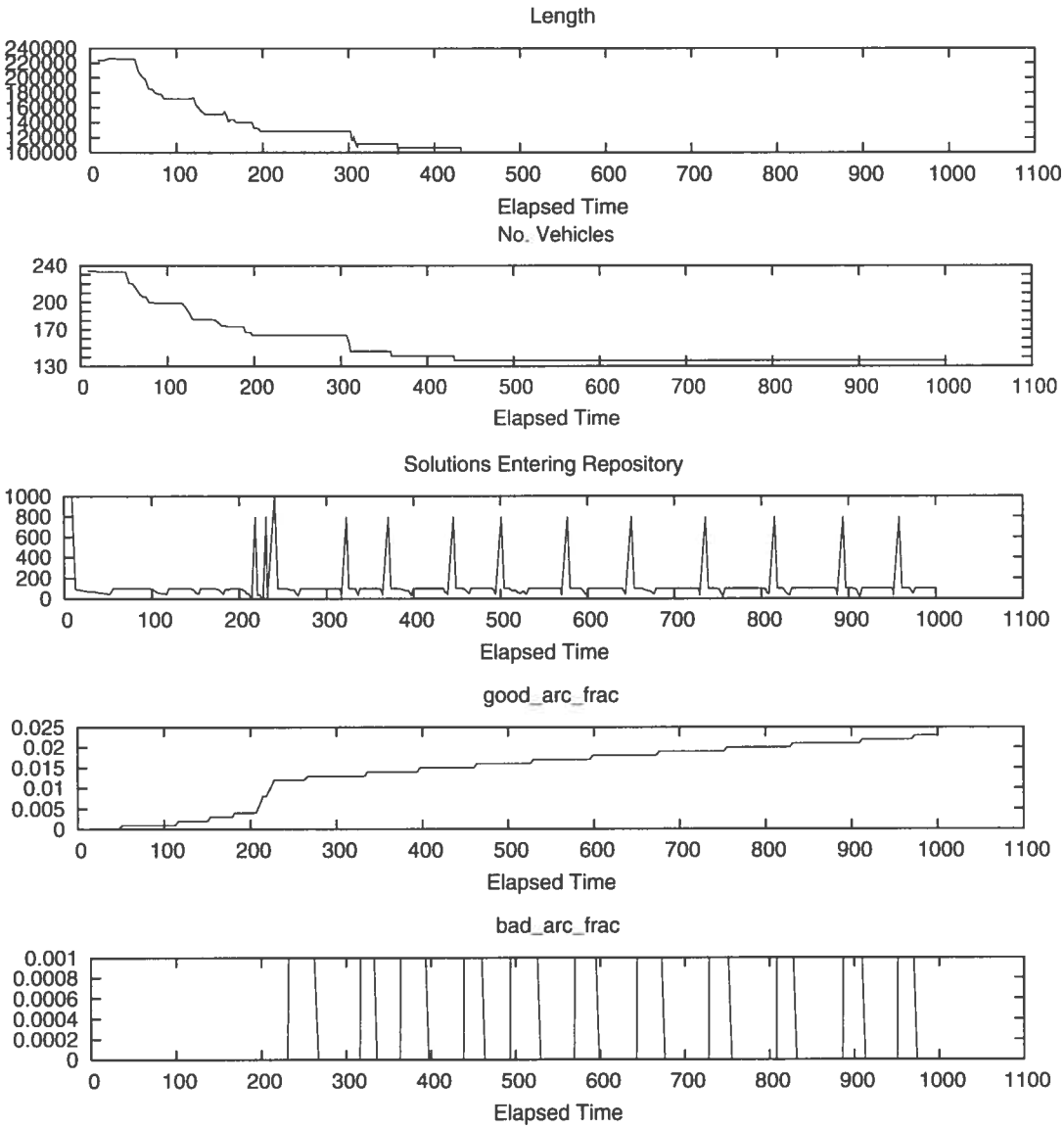

Figure 4.22: Plot of solution state evolution for the guided generation

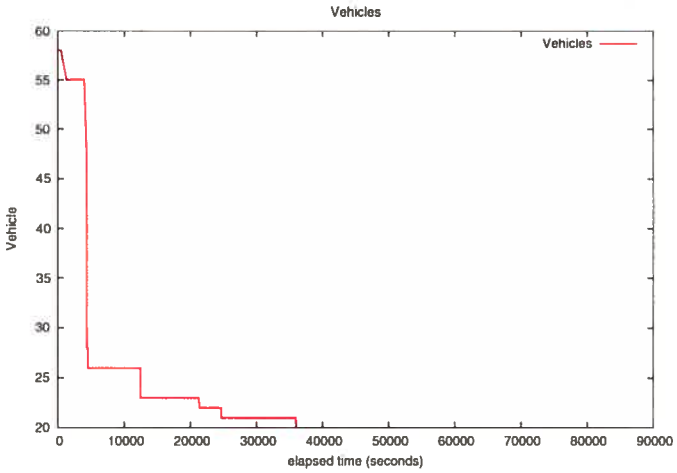

Figure 4.23: Number of Vehicle for guided random search, problem C1_2_1

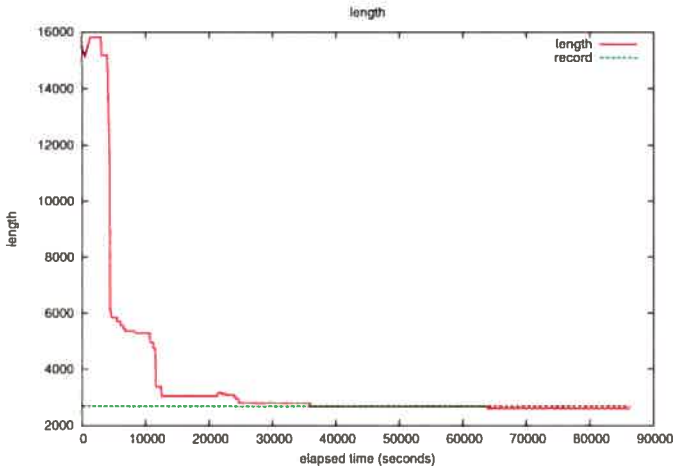

Figure 4.24: Length for guided random search, problem C1_2_1

illustrated, the number of vehicles and total traveled distance is reduced over time by the guided mechanism. The red plain line represents this guided random search while the green represents the previously best known solution.

4.16 Static vs dynamic guided search

Now that we have a better understanding on how to regulate the global search control, it is natural to put metaheuristics in the dynamic guided search framework and observe the performance of a dynamic vs static guided search.

In the following experiments, we show the results of a static guided search [79] and a dynamic guided search (this work), both using evolutionary algorithm and tabu search.

As with the static guided search, a different search method, Taburoute, an Unified Tabu Search algorithm, and two evolutionary algorithms (OX and ER), is run on each of the four processors. The solution warehouse, the post-optimization procedures, the pattern identification method, and the construction methods are run on another processor for a total of five processors in this study. In previous studies, we already have shown that having a random generator introduce diversity and increase the overall solution quality, so we still keep one random generator in our run after the construction mechanism have completed.

For Taburoute we use the parameter settings indicated in the original paper for the VRP (Gendreau, Hertz, and Laporte [49]). Tabu tags were set to a length varying between 5 and 10 iterations, 15% of the nodes were evaluated in the p-neighborhood dimensions and the initial solution was selected as the best from 15 initially generated solutions and the best solution from the solution warehouse. A penalty of 1 was used for frequently moved arcs. The parameters for the value function presented in the initial article by Cordeau, Laporte, and Mercier (Cordeau, Laporte, and Mercier [23]) (alpha:=1; beta=1; gamma=1) were used for the Unified Tabu. Finally, an arc mutation probability of 1 percent was used on temporary copies of the parents for the crossovers used by the evolutionary algorithms.

Tests have been carried out on the standard set of test problems proposed by Solomon [93]. The set contains 56 problems of 100 customers each. We also used the extended set produced by Homberger and Gehring [64] with 300 problem instances that vary from 200 to 1000 customers. The Solomon and extended problems are divided into six categories, named C1, C2, R1, R2, RC1, and RC2. For all problem instances, customers are distributed in a [0,100] square unit. The customers in sets C are clustered together, while those in sets R are distributed randomly. Problems in sets RC combine the two characteristics. Time windows at the depot are relatively small for problems of type 1, to allow fewer customers to be served by each route; time windows are larger for problems of type 2. The service time is of 10 units by customer for problems of type R and RC, and of 90 units for class C.

Solutions in the solution warehouse are sorted, first by the number of vehicles, second by a weighted sum, C(p), of attributes: the total time required to serve all customers, the associated total distance and total waiting time at customers, and the sum of the slack left in each time window: C(p)=W1*totalTime + W2*totalDistance + W3*totalWait + W4*totalSlack.

Parameters W1 to W4 were set to 1 in all the reported experiments. This measure combined with the number of vehicles gives us an overall idea of the solution quality (totalTime and totalDistance) and flexibility (totalWait and totalSlack). The last two measures indicate how much slack there exists in the solution and how easily feasible neighboring solutions may be explored.

In previous research (Le Bouthillier and Crainic [77]), cooperative search was found to provide faster results of equivalent or better quality than each of its independent searches. We therefore compare only the static guided and dynamic guided parallel cooperative searches.

Runs of 12 min wall-clock time were performed by the cooperative meta-heuristics for each of the 100 city problems. Longer running times, equal to those reported by Homberger and Gehring [64] were allowed for the larger problem instances. These times go up to 50 min wall-clock time for the 1000 city problem. Experiments were performed on the Mammouth-MP cluster from the University of Sherbrooke

on five 3.6Ghz Xeon processor with 8GB of RAM and Infiniband 4x connectivity. Computations of distances were carried out in double precision.

The implementation is machine independent and can be run with MPICH parallel library on Unix or Linux.

Tables 4.2 and 4.3 display the average results per class of problems for the cumulative number of vehicles (CNV) and distance (CTD) for the standard Solomon problems and for the extended set of problems defined by Homberger, respectively. Best results are shown in bold.

Results of the dynamic guided Cooperative Search (LCK07) are compared to those of the best methods (published or not) for the VRPTW on the benchmark site of SINTEF [1] on February 26, 2007: The unified Tabu Search of Cordeau, Laporte and Mercier ([23], denoted CLM), the evolutionary algorithm of Homberger and Gehring ([64], denoted HG), the two-stage hybrid local search of Bent and Van Hentenryck ([5], denoted BVH), the Active Guided Evolution Strategies of Mester and Bräysy ([82], denoted MB), Rochat and Taillard ([87], denoted RT), Gehring and Homberger ([47], denoted GH01), Bräysy ([12], denoted B01) and Le Bouthillier et al. ([79], the static guided search denoted LCK05).

The proposed method, LCK07, yields very good results. It appears in second place in both tables according to the CNV/CTD ratio. Relative to the standard Solomon problems, the method we propose yields a total number of vehicles of 405, which is equal to the best metaheuristics currently available for the VRPTW. It reduces the total distance by 34.63 units compared to the static guided search.

We observe that, compared to the static guided cooperative search, the new method reduces the number of vehicles by 8 on large problems (200 to 1000 customers). We also report 1 new best. The guided cooperative search reports new best averages for the R1 and R2 problem classes on the 100 customer problem. It also reports the best cumulative number of vehicles for the 200, 400 and 1000 customer problem. Our method is the second best in terms of CNV and CTD.

Table 4.3 displays the results for the cumulative number of vehicles (CNV) and

[1] Sintef web site www.sintef.no/static/am/opti/projects/top/

Author	R1	R2	C1	C2	RC1	RC2	CNV/CTD	Experiment
RT	12.25	2.91	**10.00**	**3.00**	11.88	3.38	415	SG. 100MHz
	1208.50	961.72	**828.38**	**589.86**	1377.39	1117.44	57231	1 run, 92.2 Min
CLM	12.08	2.73	**10.00**	**3.00**	11.50	3.25	407	n/a
	1210.14	969.57	**828.38**	**589.86**	1389.78	1134.52	57555.00	Sun U2 300MHz
LC03	12.08	2.73	**10.00**	**3.00**	11.50	3.25	407	5xP850 MHz.
	1209.19	963.62	**828.38**	**589.86**	1389.22	1143.70	57412.00	1 run, 12 Min
H99	11.92	2.73	**10.00**	**3.00**	11.63	3.25	406	P200 MHz.
	1228.06	969.95	**828.38**	**589.86**	1392.57	1144.43	57 876.00	10 runs, 13 Min
GH01	12.00	2.73	**10.00**	3.00	11.50	3.25	406	4 P400 MHz.
	1217.57	961.29	**828.63**	590.33	1395.13	1139.37	57641.00	5 runs, 13.5 Min
B01	11.92	2.73	**10.00**	**3.00**	11.50	3.25	405	P200 MHz.
	1222.12	975.12	**828.38**	**589.86**	1389.58	1128.39	57710.00	1 run, 82.5 Min
LCK05	11.92	2.73	**10.00**	**3.00**	11.50	3.25	405	5xP850 MHz.
	1214.20	954.32	**828.38**	**589.86**	1385.30	1129.43	57360.00	1 run, 12 Min
LCK07	**11.92**	**2.73**	**10.00**	**3.00**	11.50	3.25	405	5xP3.6 GHz, IB4x.
	1213.06	**952.73**	**828.38**	**589.86**	1385.14	1129.16	57325.37	1 run, 12 Min
BVH	12.18	2.73	**10.00**	**3.00**	**11.50**	**3.25**	**405**	Sun U10 440Mhz
	1231.08	954.18	**828.38**	**589.86**	**1384.17**	**1124.47**	**57272.00**	5 run, 120 Min

Table 4.2: Comparison of average results on 100-customer problems

Problem	MB		LCK07		LCK05		HG	
	CNV	CTD	CNV	CTD	CNV	CTD	CNV	CTD
200 total	694	168573	**694**	**169903.27**	694	169958.49	694	173312.32
400 total	1389	390386	**1388**	**396158.55**	1389	396611.15	1388	409763.38
600 total	2082	796172	2084	809882.12	2086	809493.46	**2076**	**851680.33**
800 total	2765	1361586	2759	1444524.75	2761	1443399.01	**2755**	**1479801.56**
1000 total	3446	2078110	**3439**	**2133673.47**	3442	2133644.57	3439	2236582.89
Total	10376	4794827	10364	4954142.16	10372	4953106.68	**10352**	**5151140.48**

Table 4.3: CNV/CTD for 100-1000 customers

distance (CTD) aggregated by problem size for the extended set of problems. Not many authors addressed the entire problem set, which explains the limited number of entries in the table (the best methods were selected).

For all problem classes, the difference with the other methods is less than 0.4 percent in terms of the cumulative number of vehicles. In all cases, we improved upon the static cooperative search.

4.17 Conclusion and perspectives

The main challenge in this work is studying the effect of the guided search on the global search is obtaining an abstraction from any method-specific performance that generates noise in the interpretation. To address this issue and focus on the guidance mechanism and its impact on cooperation, we replace each independent meta-heuristic with a random solution generator. The resulting generator threads

receive instructions to fix and prohibit certain elements while generating solutions. These instructions are coming from work on the solutions sent to the guidance mechanism, but no solution is sent to the individual threads, which are thus influenced by the guided mechanism only.

In this work, we have studied the effect of the guidance mechanism in a cooperative framework. The use use of random solution generator instead of individual metaheuristics gave us a better understanding of the guidance effect.

We were then able to improve the best known average solution quality over 3 standard benchmark problem size (200,400,1000) with a tabu search and an evolutionary algorithm guided dynamically within a cooperative framework. Adapting the parameters that constrain the search based on the entropy value of the solutions in the warehouse has proved to be successful.

As patterns of information reflect the solutions present in the warehouse, it might be interesting to maintain a perpetual pattern that would tell us which elements were visited often, and which, less often.

Patterns and global information from the solution are now being constructed and sent to the individual search. We should also look at contextual information and derive search patterns from each local search to better inform the warehouse of the actual search performed.

As one last comment, we emphasize that entropy is not just an old friend, it is a concept that is rich in new ideas and scientific directions. It is being very actively considered by a number of researchers [15] [34] [59] [16]. This study indicates that population- and cooperation-based methods should certainly consider entropy as an important ingredient in properly guiding a search. From this we conclude that the guiding mechanism could be applied to any search method that produces solutions, either a sequential or parallel one.

Further studies will analyze these new avenues for an adaptive guided search and test the improved search with state-of-the-art metaheuristics.

4.18 Acknowledgments

We would like to thank the "Fonds Québécois Recherche Nature et Technologies" for its financial support; Bernard Brosseau-Villeneuve for the initial version of the random generator and Standish Russell for his parallel library, Ecolab, and for precious help in setting up the experiment. Experiments were done with the use of the bqTools library on the Mammouth-MP cluster from the University of Sherbrooke.

4.19 Synthèse

Dans cette section, nous avons étudié si un mécanisme dynamique de contrôle global des phases d'intensification et de diversification performait mieux qu'un mécanisme statique.

Notre méthodologie consistait, dans une première phase, à cerner l'impact de la coopération et du guidage en remplaçant les métaheuristiques par des méthodes de recherches aléatoires. Dans une deuxième phase et basé sur les conclusion de notre première phase, nous avons pu intégrer notre mécanisme de guidage dynamique avec des métaheuristiques pour en observer les performances.

Les résultats des recherches (c.f section 4.15) coopératives obtenus sur le VRPTW montrent que les paramètres de guidage ont une influence différente selon le problème résolu. Un mécanisme de guidage dynamique est donc une façon naturelle d'adapter la recherche coopérative aux différentes topologies de l'espace de solutions des problèmes. Ce mécanisme est utilisé dans notre approche afin de contrôler la diversité de la population de l'entrepôt, d'éviter la convergence prématurée des recherches et de conserver une représentation adéquate des propriétés des solutions prometteuses.

Quels sont les impacts de la fixation ou de l'interdiction d'éléments prometteurs ou non sur l'entropie et sur la qualité des solutions produites ? Une étude empirique a permis de répondre à cette question en effectuant une variation paramétrique des proportions des patrons d'éléments fixés ou non. Afin d'éliminer toute influence de comportement particulier de métaheuristiques ou d'échanges de solutions, un générateur aléatoire de solutions est utilisé.

Nous identifions cinq scénarios de fréquence d'apparition de patrons d'éléments au sein d'une population (c.f. figure 4.4). Seulement deux de ces scénarios sont indicateurs de patrons, prometteurs ou non, pour lesquels il peut être pertinent de fixer ou d'interdire les éléments dans les recherches. La définition de patrons inclut maintenant les patrons statistiques qui peuvent faire partie d'une solution réalisable. Cette extension permet de générer de nouvelles solutions qui combinent

les éléments prometteurs de différentes solutions.

Fixer ou interdire un élément sans considération quantitative de son influence dans les solutions où il est présent n'est pas représentatif de l'objectif global. Afin de palier ce problème, nous attribuons une fonction d'utilité à chaque élément qui caractérise son influence relative. Il est possible de fixer ou d'interdire de façon probabiliste les éléments de manière plus ou moins prononcée en fonction de leur utilité respective.

Une fixation trop imposante d'éléments créée nécessairement une diminution prématurée de l'entropie au détriment d'une exploration diversifiée de l'espace de solutions. Ainsi, s'assurer que l'entropie de la population demeure à une certaine distance de l'entropie idéale permet d'améliorer le mécanisme de guidage, en laissant orienter les recherches vers un nombre plus important d'espaces de solutions prometteuses. L'entropie utilisée mesure le nombre d'éléments distincts dans la population. La première phase d'expérimentation 4.15 a démontré que le contrôle de l'entropie limitait la convergence prématurée de la population mais offrait très peu de diversification. En effet, interdire certains éléments dans le but de diversifier les solutions produit, certes, des solutions différentes, mais pas nécessairement de qualité suffisante pour remplacer les solutions existantes de l'entrepôt de données. Une méthode simple consiste à éliminer une partie de la population avant la diversification, comme bon nombre d'algorithmes génétiques le font. Enlever 50% de la population, selon nos expériences, a produit les meilleurs résultats. Ceci permet au mécanisme de guidage de prendre en compte de nouveaux patrons et de mieux aiguiller les recherches. Suite à l'entrée de solutions diverses dans la population et aux nouvelles instructions de guidage qui en ont découlé, de meilleures solutions ont été observées.

4.19.1 Discussion des résultats

Sans simple fixation d'éléments, les solutions produites par un générateur aléatoire sont de piètre qualité. L'entropie des solutions de l'entrepôt de données demeure élevée. Dès que l'entrepôt est rempli, le taux d'acceptation de nouvelles

solutions tombe radicalement, puisqu'il y a très peu de nouvelles bonnes solutions produites.

Par une variation paramétrique, notre étude montre l'influence du mécanisme d'interdiction et de fixation de patron d'élément basé sur la qualité respective de leur solution, l'élimination de solutions de la population ainsi que le guidage de la recherche pour maintenir un niveau déterminé d'entropie. Les valeurs identifiées ont conféré les meilleurs résultats aux VRPTW.

Le mécanisme de guidage dynamique combiné à l'élimination d'une partie des solutions de l'entrepôt permet à de meilleures solutions d'y entrer. En effet, à chaque fois qu'une interdiction d'éléments a lieu et que 50% de la population est enlevée (soit 500 solutions), il y a en moyenne 700 solutions qui entrent dans l'entrepôt pour les problèmes étudiés. Ceci signifie que 200 nouvelles meilleures solutions ont été trouvées.

Sur six problèmes sélectionnés parmi les 1000 noeuds (1 par classe de problème), l'utilisation du mécanisme de guidage dynamique confère une amélioration moyenne de 170% en nombre de véhicules et de 156% en distance par rapport à une recherche aléatoire pure. Pour l'ensemble des problèmes du VRPTW, notre mécanisme de guidage dynamique combiné à des recherches avec tabous et algorithmes évolutionaire a dominé, en terme de qualité de solution le guidage statique.

4.19.2 Conclusion

En isolant la coopération et le guidage des méthodes de recherche particulière, nous avons identifié le degré d'influence de la fixation ou de l'interdiction de patrons d'éléments sur l'entropie de la population et le lien avec la qualité des solutions produites. Nous avons aussi observé que le maintien d'une entropie constante à 1.3 fois l'entropie minimale procure de meilleurs résultats pour les problèmes étudiés. Afin de conserver l'entropie à ce niveau, nous avons proposé un mécanisme permettant de modifier la proportion des patrons d'éléments qui doivent êtres fixés ou interdits, et ce, de façon dynamique. Le mécanisme de guidage dynamique a permis de trouver un meilleur résultat de littérature et d'améliorer la robustesse de la re-

cherche en obtenant une meilleure moyenne des résultats pour les problèmes des tailles 200,400 et 1000.

CHAPITRE 5

CONCLUSION

La recherche coopérative est un paradigme qui laisse entrevoir des perspectives d'avancées significatives dans la qualité et la robustesse des solutions produites. En ce sens, nous avons proposé une architecture générique et des mécanismes permettant :

1. d'enrichir la gestion de l'entrepôt de données supportant la recherche ;
2. de guider les métaheuristiques vers les espaces de solutions prometteuses ;

Ces mécanismes confèrent une vision et un contrôle améliorés sur la recherche globale. Ils ont permis une gestion et une distribution appropriée de l'information ainsi que l'implémentation efficace sur des systèmes parallèles.

En plus de la résolution du VRPTW, les applications de la recherche coopérative avec guidage dynamique sont nombreuses. Elle peut être utilisée pour résoudre plusieurs problèmes d'optimisation par la simple intégration de métaheuristiques appropriées.

5.1 Contributions

L'amélioration d'une architecture de recherche coopérative applicable à un ensemble de problèmes combinatoires est la principale réalisation de cette thèse. Ainsi nous avons fourni une amélioration de la robustesse des résultats de littérature pour le VRPTW avec plusieurs nouveaux résultats et le deuxième meilleur algorithme[1] en terme de la moyenne des résultats de problèmes de 100 à 1000 noeuds. Nous avons proposé d'identifier des patrons d'éléments prometteurs ou non au sein des solutions explorées pour faire un guidage en phases de diversifications et d'intensificaitons global de la recherche. Un guidage dynamique basé sur une métrique de

[1]SINTEF : www.sintef.no/static/am/opti/projects/top/

l'entropie des solutions de la population fut proposé. Finalement, nous avons effectué, une évaluation empirique du comportement de la coopération et de l'influence du guidage.

Une introduction aux recherches coopératives a été illustrée par une publication démontrant le bien-fondé de la méthode appliquée à un VRPTW. En effet, l'échange de solutions améliorantes entre deux recherches avec tabous et deux algorithmes génétiques a permis de trouver des meilleures solutions à cinq problèmes de littérature.

Le troisième chapitre a élargi le concept de coopération en proposant un mécanisme de guidage. Nous identifions des patrons (prometteurs ou non) selon la variation de fréquence des éléments entre les meilleures et moins bonnes solutions. Ces patrons d'éléments peuvent ensuite être fixés ou interdits aux métaheuristiques afin d'orienter les recherches vers des endroits prometteurs ou de les dévier de certains endroits déjà explorés. Ce mécanisme de guidage permet d'imposer des phases successives de diversification et d'intensification des recherches. En ajoutant un guidage en quatre phases d'intensification et de diversification à la recherche coopérative, nous obtenons des résultats de qualité identique ou supérieure pour tous les problèmes de VRPTW. Les perspectives soulignent qu'il serait intéressant d'appliquer ce mécanisme de guidage avec des méthodes parallèles non coopératives, de même qu'avec des méthodes séquentielles qui disposent ou non de mémoire.

Pour sa part, le quatrième chapitre a enrichi le mécanisme de guidage en le rendant dynamique. Nous avons observé que le maintien d'un niveau d'entropie dans la population avait des effets bénéfiques sur la qualité de la coopération. Puisque le guidage permet d'intensifier ou de diversifier les recherches en certains endroits et donc d'influencer l'entropie, une adaptation dynamique de l'intensité du guidage permet de maintenir la population à un niveau d'entropie donné. Les expériences effectuées à la section 4.15 à l'aide d'un générateur aléatoire, qui permet de s'abstraire de toutes les particularités inhérentes à une métaheuristique en particulier, ont démontré des améliorations en moyenne de 170% sur le nombre de véhicules et de 156% en distance.

5.2 Perspectives

Cette section discute les perspectives de recherche qui font suite à nos travaux.

Jusqu'à présent, les méthodes de résolutions ont été considérées homogènes, c'est-à-dire que les mécanismes d'envoi de solution et de guidage sont invariables, peu importe leur comportement. Nous croyons qu'un mécanisme de sélection des solutions de l'entrepôt ainsi qu'un guidage spécifique à chaque méthode selon son comportement offriraient de meilleures performances.

D'un point de vue général, il est possible pour l'entrepôt de données d'émettre des instructions modifiant individuellement le comportement des méthodes de résolution par fixation et interdiction d'éléments. Les trajectoires de recherches de chaque méthode sont ainsi adaptée en conséquence.

Nous abordons cette piste sous trois aspects méthodologiques, que nous définissons dans les prochaines sections : la transmission des trajectoires de recherches, la répartition ou la concentration de la recherche et la fonction dynamique de sélection de solutions.

5.2.1 Transmission des trajectoires de recherches

L'identification de caractéristiques récurrentes et divergentes des solutions, comme l'utilisation fréquente ou l'absence de certains éléments, permet d'identifier celles comportant des particularités propres aux espaces de solutions prometteuses ou non. Le mécanisme d'identification de patron pourrait aussi servir à trouver les éléments prometteurs ou non au sein d'une même méthode.

Plus spécifiquement, la connaissance des trajectoires de recherches des métaheuristiques par l'entrepôt, c'est-à-dire la liste des solutions visitées par métaheuristique, devrait permettre d'obtenir :

- Les éléments fréquents des solutions visitées (améliorantes ou non) ;
- L'efficacité de la recherche à moyen et long terme ;
- L'historique des solutions visitées lorsqu'une solution améliorante est trouvée.

L'importante quantité d'informations sous-jacentes aux trajectoires de recherches

impose une agrégation d'information pour leur transmission et leur entreposage. L'agrégation de l'information permettant une transmission et un entreposage efficace est un problème à résoudre.

La méthode la plus simple pour l'obtention de l'historique de recherche est l'envoi de toutes les solutions à l'entrepôt de données. Il est aussi possible d'établir une fréquence d'éléments sur les solutions explorées et de transmettre ce vecteur de fréquence à intervalle régulier.

Dans tous les cas, une analyse appropriée de ces informations transmises à l'entrepôt de données permettrait un guidage individuel et une meilleure coordination des recherches. Ceci nous amène à considérer un guidage et une sélection de solutions propres à chaque méthode de résolution. Un mécanisme d'apprentissage devrait orienter l'entrepôt dans le contrôle global et individuel de la recherche.

5.2.2 Répartition et concentration de la recherche

L'obtention des trajectoires de recherche des métaheuristiques par l'entrepôt de données permet de connaître l'historique et l'état actuel de la recherche. Cette connaissance permet d'orienter l'exploration en évitant une convergence prématurée. Nos expériences ont montré qu'il existait des valeurs d'entropie limitant ces effets. Il serait ainsi possible de répartir ou de concentrer toutes les recherches dans des endroits différents ou aux mêmes endroits. Ceci pourrait s'effectuer en utilisant un mécanisme de guidage dynamique propre à chaque métaheuristique. Il serait possible d'influencer les paramètres propres à chaque méthodes selon une perspective globale (i.e. longueur de la liste tabu, taux de mutation). Ainsi, les éléments fixés et les paramètres des méthodes pourraient varier d'une méthode à l'autre en fonction de son comportement ponctuel et de celui désiré.

D'un point de vue méthodologique, les phases de diversification et d'intensification pourraient être effectuées non pas d'un point de vue global, soit sur l'ensemble des solutions, mais plutôt effectuées sur chaque méthode de recherche selon les solutions fournies.

5.2.3 Fonction dynamique de sélection des solutions

Afin d'améliorer le contrôle global, nous croyons qu'une fonction dynamique de sélection de solutions dans l'entrepôt de données adaptée à chaque métaheuristique est préférable à une fonction globale qui a tendance à uniformiser les points de départ. La fonction de sélection proposée doit pouvoir tenir compte de la trajectoire de recherche ainsi que des résultats obtenus par chaque méthode de recherche, afin de de sélectionner de façon appropriée les phases d'intensification et de diversification.

En tenant compte des trajectoires de recherche, il serait possible de choisir une solution qui est éloignée, en terme de distance de Hamming, des solutions visitées précédemment afin de diversifier la recherche.

5.2.4 Synthèse des perspectives

Ces axes potentiels de recherche permettraient de mieux utiliser les informations locales. Cette section en formule une synthèse et surtout pose la question de leurs interactions.

En identifiant les éléments fréquents des solutions visitées (améliorantes ou non), il devient possible d'utiliser un mécanisme de guidage local et de le synchroniser avec le mécanisme de guidage global. On obtient un mécanisme qui met aussi l'emphase sur la diversification des mauvaises solutions explorées.

Un vecteur de fréquences d'éléments pondéré en fonction de la qualité de la solution permet d'identifier les éléments prometteurs ou non. En comptant le nombre de fois que chaque élément fait partie d'une solution améliorante ou non, nous pouvons connaître la répartition des solutions explorées depuis le début de la recherche. En créant un vecteur supplémentaire initialisé à chaque solution améliorante, nous obtenons une vue agrégée des derniers éléments explorés. Il est aussi possible de maintenir un vecteur qui est initialisé à chaque information reçue de l'entrepôt (nouvelle solution ou variation dans le guidage). L'impact de la coopération peut être apprécié en maintenant un historique d'évolution de ces vecteurs et en communiquant à l'entrepôt l'historique des derniers éléments visités lorsqu'une solution

de métaheuristique.

BIBLIOGRAPHIE

[1] Huberman A.B. *The value of cooperation.* Elsevier Science Publisher B.V., 1992.

[2] Aiex, R.M., Martins, S.L., Ribeiro, C.C. et Rodriguez, N.R. Cooperative Multi-Thread Parallel Tabu Search with an Application to Circuit Partitioning. Dans *Proceedings of IRREGULAR'98 - 5th International Symposium on Solving Irregularly Structured Problems in Parallel*, volume 1457 de *Lecture Notes in Computer Science*, pages 310–331. Springer-Verlag, 1998.

[3] Badeau, P., Guertin, F., Gendreau, M., Potvin, J.-Y. et Taillard, É.D. A Parallel Tabu Search Heuristic for the Vehicle Routing Problem with Time Windows. *Transportation Research C : Emerging Technologies*, 5(2):109–122, 1997.

[4] Beame, P., Cook, S., Edmonds, J., Impagliazzo, R. et Pitassi, T. The relative complexity of NP search problems. *Proceedings of the Twenty-Seventh Annual ACM Symposium on the Theory of Computing*, pages 303–314, 1995.

[5] Bent, R. et Van Hentenryck, P. A Two-Stage Hybrid Local Search for the Vehicle Routing Problem with Time Windows. *Transportation Science*, 38 (4):515–530, 2004.

[6] J. Bentley. Fast algorithms for geometric salesman problems. *ORSA J. Comp.*, pages 388–411, 1992.

[7] Berger, J., Barkaoui, M. et Bräysy, O. . A Parallel Hybrid Genetic Algorithm for the Vehicle Routing Problem with Time Windows. Working paper, Defense Research Establishment Valcartier, Canada, 2001.

[8] Boctor, F.F., Laporte, G. et Renaud J. Heuristics for the Traveling Purchaser Problem. *Computers & Operations Research*, 30(4):491–504, 2003.

[9] Bodin, L.D. et A. Mingozzi. Street Routing and Scheduling Problems. Dans Hall, R.W., éditeur, *Handbook of Transportation Science*. Kluwer Academic Publishers, Norwell, MA, 1999. forthcoming.

[10] Bonabeau, E., Theralauz, G., Deneubourg, J.-L. et Camazine, S. Self-organization in social insects. *Trends in Ecology and Evolution*, 12(5):188–193, 1997.

[11] O. Bräysy, G. Hasle et W. Dullaert. A multi-Start Local Search Algorithm for the Vehicle Routing Problem with Time Windows. *to appear in European Journal of Operational Research*, 2003.

[12] Bräysy, O. A Reactive Variable Neighborhood Search Algorithm for the Vehicle Routing Problem with Time Windows. *INFORMS Journal on Computing*, 15(4):347–368, 2003.

[13] Bräysy, O. et Dullaert, W. A Fast Evolutionary Metaheuristic for the Vehicle Routing Problem with Time Windows. *International Journal on Artificial Intelligence*, 12(2), 2003.

[14] Bräysy, O. et Gendreau, M. Tabu search heuristics for the vehicle routing problem with time windows. *Top*, 10(2):211–238, 2003.

[15] Camazine, S., Deneubourg, J.-L., Franks, N.R., Sneyd, J., Theraulaz, G. et Bonabeau, E. *Self-Organization in Biological Systems*. Princeton Studies in Complexity, princeton and oxford édition, 2001.

[16] Capacarrère, W., Tettamanzi, A., Tomassini, M. et Sipper M. Studying parallel evolutionary algorithms : The cellular programming case. Dans PPSN V. A. E. Eiben, Bäck, T., Schoenauer, M. et Schwefel,H.P., éditeurs, *Parallel Problem Solving from Nature*, volume 1498 de *Lecture Notes in Computer Science*. Springer-Verlag, 1998.

[17] Y. Caseau et F. Laburthe. Heuristics for Large Constrained Vehicle Routing Problems. *Journal of Heuristics*, 5:281–303, 1999.

[18] Chabrier, A. Vehicle Routing Problem with Elementary Shortest Path based Column Generation. Working paper, ILOG, Madrid, Spain, 2002.

[19] Christofides, N., Mingozzi A. et Toth, P. The Vehicle Routing Problem. Dans N. Christofides, Mingozzi A., P. Toth et C. Sandi, éditeurs, *Combinatorial Optimization*, pages 315–338. John Wiley, New York, 1979.

[20] Cook, W. et Rich, J.L. A parallel cutting plane algorithm for the vehicle routing problem with time windows. Working paper, Computational and Applied Mathematics, Rice University, Houston, TX, 1999.

[21] Cordeau, J.-F., Desaulniers, G., Desrosiers, J., Solomon, M. M. et Soumis, F. The VRP with Time Windows. Dans Toth, P. et Vigo, D., éditeurs, *The Vehicle Routing Problem*, SIAM, Monographs on Discrete Mathematics and Applications, chaptitre 7, pages 157–193. SIAM, Philadelphia, PA, 2002.

[22] Cordeau, J.-F., Gendreau, M., Laporte, G., Potvin, J.-Y. et Semet, F. A Guide to Vehicle Routing Heuristics. *Journal of the Operational Research Society*, 53:512–522, 2002.

[23] Cordeau, J.-F., Laporte, G. et Mercier, A. A Unified Tabu Search Heuristic for Vehicle Routing Problems with Time Windows. *Journal of the Operational Research Society*, 52:928–936, 2001.

[24] Crainic, T.G. Parallel Computation, Co-operation, Tabu Search. Dans C. Rego et B. Alidaee, éditeurs, *Adaptive Memory and Evolution : Tabu Search and Scatter Search*. Kluwer Academic Publishers, Norwell, MA, 2002.

[25] Crainic, T.G. et Gendreau, M. Towards an Evolutionary Method - Cooperating Multi-Thread Parallel Tabu Search Hybrid. Dans S. Voß, S. Martello, C. Roucairol et Osman, I.H., éditeurs, *Meta-Heuristics 98 : Theory & Applications*, pages 331–344. Kluwer Academic Publishers, Norwell, MA, 1999.

[26] Crainic, T.G. et Gendreau, M. Cooperative Parallel Tabu Search for Capacitated Network Design. *Journal of Heuristics*, 8(6):601–627, 2002.

[27] Crainic, T.G., Li, Y. et Toulouse, M. A simple cooperative multilevel algorithm for the capacitated multicommodity network design. *Computer & Operations Research*, 33(9):2602–2622, September 2006.

[28] Crainic, T.G. et Toulouse, M. Parallel Metaheuristics. Dans T.G. Crainic et G. Laporte, éditeurs, *Fleet Management and Logistics*, pages 205–251. Kluwer Academic Publishers, Norwell, MA, 1998.

[29] Crainic, T.G. et Toulouse, M. Parallel Strategies for Meta-heuristics. Dans F. Glover et G. Kochenberger, éditeurs, *State-of-the-Art Handbook in Metaheuristics*, pages 475–513. Kluwer Academic Publishers, Norwell, MA, 2003.

[30] Crainic, T.G., Toulouse, M. et Gendreau, M. Parallel Asynchronous Tabu Search for Multicommodity Location-Allocation with Balancing Requirements. *Annals of Operations Research*, 63:277–299, 1995.

[31] Crainic, T.G., Toulouse, M. et Gendreau, M. Synchronous tabu search parallelization strategies for multicommodity location-allocation with balancing requirments. *OR Spektrum*, 17(2/3), 1995.

[32] Cung, V.-D., Martins, S.L., Ribeiro, C.C. et Roucairol, C. Strategies for the Parallel Implementations of Metaheuristics. Dans C.C. Ribeiro et P. Hansen, éditeurs, *Essays and Surveys in Metaheuristics*, pages 263–308. Kluwer Academic Publishers, Norwell, MA, 2002.

[33] B. De Backer, V. Furnon, P. Shaw, P. Kilby et P. Prosser. Solving Vehicle Routing Problems Using Constraint Programming and Metaheuristics. *Journal of Heuristics*, 6:501–523, 2000.

[34] L. Demetrius. Growth rate, population entropy and evolutionary dynamics. *Theor. Popul. Biol.*, 41:208–236, 1992.

[35] G. Desaulniers, J. Desrosiers, I. Ioachim, Solomon, M.M., F. Soumis et D. Villeneuve. A Unified Framework for Deterministic Time Constrained Vehicle

Routing and Crew Scheduling Problems. Dans T.G. Crainic et G. Laporte, éditeurs, *Fleet Management and Logistics*, pages 57–93. Kluwer Academic Publishers, Norwell, MA, 1998.

[36] Desrosiers, J., Dumas, Y., Solomon, M.M. et Soumis, F. Time Constrained Routing and Scheduling. Dans M. Ball, Magnanti, T.L., Monma, C.L. et Nemhauser, G.L., éditeurs, *Network Routing*, volume 8 de *Handbooks in Operations Research and Management Science*, pages 35–139. North-Holland, Amsterdam, 1995.

[37] M. Dror, éditeur. *Arc Routing : Theory, Solutions and Applications.* Kluwer Academic Publishers, Norwell, MA, 2000.

[38] Durfee, E.H., Lesser, V.R. et Corkill, D.D. Coherent cooperation among communicating problem solvers. *IEEE Transactions on Computers*, C-36: 1275–1291, 1987.

[39] Durfee, E.H., Lesser, V.R. et Corkill, D.D. Cooperation through communication in a distributed problem solving network. *Distributed Artificial Intelligence*, pages 29–58, 1987.

[40] Enslow, H.P. What is a "Distributed" Data Processing System ? *Computer*, 11:13–21, 1978.

[41] Festa, P. et Resende, M.G.C. GRASP : An Annotated Bibliography. Dans C.C. Ribeiro et P. Hansen, éditeurs, *Essays and Surveys in Metaheuristics*, pages 325–367. Kluwer Academic Publishers, Norwell, MA, 2002.

[42] Fisher, M.L. Vehicle Routing. Dans M. Ball, Magnanti, T.L., Monma, C.L. et Nemhauser, G.L., éditeurs, *Network Routing*, volume 8 de *Handbooks in Operations Research and Management Science*, pages 1–33. North-Holland, Amsterdam, 1995.

[43] Flynn, M.J. Very High-Speed Computing Systems. *Proceedings of the IEEE*, 54:1901–1909, 1966.

[44] L. M. Gambardella, E. Taillard et G. Agazzi. MACS-VRPTW : A Multiple Ant Colony System for Vehicle Routing Problems with Time Windows. Dans M. Dorigo D. Corne et F. Glover (eds), éditeurs, *New Ideas in Optimization*, pages 63–76. McGraw-Hill, 1999.

[45] Garcia, B.L., J.-Y. Potvin et Rousseau, J.M. A Parallel Implementation of the Tabu Search Heuristic for Vehicle Routing Problems with Time Window Constraints. *Computers & Operations Research*, 21(9):1025–1033, 1994.

[46] H. Gehring et J. Homberger. A Parallel Two-phase Metaheuristic for Routing Problems with Time Windows. *Asia-Pacific Journal of Operational Research*, 18:35–47, 2001.

[47] Gehring, H. et Homberger, J. A parallel two-phase metaheuristic for routing problems with time windows. *Asia-Pacific Journal of Operational Research*, 18(1):35–47, 2001.

[48] Gendreau, M., Hertz, A. et Laporte, G. New Insertion and Postoptimization Procedures for the Traveling Salesman Problem. *Operations Research*, 40(6): 1086–1094, 1992.

[49] Gendreau, M., Hertz, A. et Laporte, G. A Tabu Search Heuristic for the Vehicle Routing Problem. *Management Science*, 40:1276–1290, 1994.

[50] Gendreau, M., Hertz, A., Laporte, G. et Stan, M. A Generalized Insertion Heuristic for the Traveling Salesman Problem with Time Windows. *Operations Research*, 46(3):330–335, 1998.

[51] Gendreau, M., Laporte, G. et Potvin, J.-Y. Metaheuristics for the Vehicle Routing Problem. Dans P. Toth et D. Vigo, éditeurs, *The Vehicle Routing Problem*, volume 9 de *SIAM Monographs on Discrete Mathematics and Applications*, pages 129–154. SIAM, 2002.

[52] F. Glover. Multilevel Tabu Search and Embedded Search Neighborhoods for the Traveling Salesman Problem. Working paper, College of Business & Administration, University of Colorado, U.S.A., 1991.

[53] Glover, F. Tabu Search – Part I. *ORSA Journal on Computing*, 1(3):190–206, 1989.

[54] Glover, F. Tabu Search – Part II. *ORSA Journal on Computing*, 2(1):4–32, 1990.

[55] Glover, F. Ejection Chains, Reference Structures and Alternating Path Methods for Traveling Salesman Problems. *Discrete Applied Mathematics*, 49: 231–255, 1992.

[56] Glover, F. et Laguna, M. *Tabu Search.* Kluwer Academic Publishers, Norwell, MA, 1997.

[57] Golden, B.L. et Assad, A.A., éditeurs. *Vehicle Routing : Methods and Studies.* North-Holland, Amsterdam, 1988.

[58] Golden, B.L., Wasil, E.A., Kelly, J.P. et Chao, I.M. Metaheuristics in Vehicle Routing. Dans T.G. Crainic et G. Laporte, éditeurs, *Fleet Management and Logistics*, pages 33–56. Kluwer Academic Publishers, Boston, MA, 1998.

[59] J.J. Grefenstette. Incorporating problem specific knowledge into genetic algorithms. Dans L. Davis, éditeur, *Genetic Algorithms and Simulated Annealing*, pages 42–60. Morgan Kaufmann, London, 1987.

[60] Hansen, P. et Mladenović, N. An Introduction to Variable Neighborhood Search. Dans S. Voß, S. Martello, C. Roucairol et Osman, I.H., éditeurs, *Meta-Heuristics 98 : Theory & Applications*, pages 433–458. Kluwer Academic Publishers, Norwell, MA, 1999.

[61] Hansen, P. et Mladenović, N. Developments of Variable Neighborhood Search. Dans C.C. Ribeiro et P. Hansen, éditeurs, *Essays and Surveys in Metaheuristics*, pages 415–439. Kluwer Academic Publishers, Norwell, MA, 2002.

[62] Hogg, T. et Colin, P.W. Solving the really hard problems with cooperative search. *AAAI-93*, pages 231–236, 1993.

[63] J. Homberger. Verteilt-parallele Metaheuristiken zur Tourenplanung. Unpublished, 2000.

[64] Homberger, J. et Gehring, H. Two Evolutionary Metaheuristics for the Vehicle Routing Problem with Time Windows. *INFOR*, 37:297–318, 1999.

[65] Homberger, J. et Gehring, H. Two Evolutionary Metaheuristics for the Vehicle Routing Problem with Time Windows. *INFOR*, 37:297–318, 1999.

[66] B.A. Huberman. The performance of cooperative processes. *Physica D*, 42: 38–47, 1990.

[67] Huberman A.B. et Hogg, T. *The behavior of computational ecologies.* Elsevier Science Publisher B.V., 1998.

[68] Kallehauge, B., Larsen, J. et Madsen, O.B.G. Lagrangean duality and non-differentiable optimization applied on routing with time windows - experimental results. Internal report imm-rep-2000-8, Department of Mathematical Modelling, Technical University of Denmark, Lyngby, Denmark, 2000.

[69] G. Kindervater et M. Savelsbergh. Vehicle Routing : Handling Edge Exchanges. Dans E. Aarts et J.K. Lenstra (eds), éditeurs, *Local Search in Combinatorial Optimization*, chaptitre 10, pages 337–360. John Wiley & Sons Ltd., 1997.

[70] N. Kohl, J. Desrosiers, O. B. G. Madsen, M. M. Solomon et F. Soumis. 2-Path Cuts for the Vehicle Routing Problem with Time Windows. *Transportation Science*, Vol. 33(1):101–116, 1999.

[71] G. Laporte. The Vehicle Routing Problem : An Overview of Exact and Approximate Algorithms. *European Journal of Operational Research*, 59: 345–358, 1992b.

[72] Laporte, G., Gendreau, M., Potvin, J.-Y. et Semet, F. Classical and Modern Heuristics for the Vehicle Routing Problem. *International Transactions in Operational Research*, 7(4/5):285–300, 2000.

[73] Laporte, G. et Semet, F. Classical Heuristics for the Vehicle Routing Problem. Dans P. Toth et D. Vigo, éditeurs, *The Vehicle Routing Problem*, volume 9 de *SIAM Monographs on Discrete Mathematics and Applications*, pages 109–128. SIAM, 2002.

[74] J. Larsen. *Parellellization of the vehicle routing problem with time windows.* Ph.d. thesis imm-phd-1999-62, Department of Mathematical Modelling, Technical University of Denmark, Lyngby, Denmark, 1999.

[75] Le Bouthillier, A. Modélisation UML pour une architecture coopérative appliquée au problème de tournées de véhicules avec fenêtres de temps. Mémoire de maîtrise, Université de Montréal, Montréal, QC, Canada, Décembre 2000.

[76] Le Bouthillier, A. et Crainic, T.G. A Cooperative Parallel Meta-Heuristic for the Vehicle Routing Problem with Time Windows. Publication CRT-2002-55, Centre de recherche sur les transports, Université de Montréal, Montréal, QC, Canada, Décembre 2002.

[77] Le Bouthillier, A. et Crainic, T.G. A Cooperative Parallel Meta-Heuristic for the Vehicle Routing Problem with Time Windows. *Computers & Operations Research*, 32(7):1685–1708, 2005.

[78] Le Bouthillier, A., Crainic, T.G. et Kropf, P. Towards a Guided Cooperative Search. Publication CRT-05-09, Centre de recherche sur les transports, Université de Montréal, Montréal, QC, Canada, 2005.

[79] Le Bouthillier, A., Crainic, T.G. et Kropf, Peter. A Guided Cooperative Search for the Vehicle Routing Problem with Time Windows. *IEEE Intelligent Systems*, 20(4):36–42, July/August 2005.

[80] Lesser, V.R. et Corkill, D.D. *Functionally accurate, cooperative distributed systems*, volume C-36 de *Readings in Distributed Artificial Intelligence*. Morgan Kaufmann, 1988.

[81] Maniezzo, V. et Carbonaro, A. Ant Colony Optimization : An Overview. Dans C.C. Ribeiro et P. Hansen, éditeurs, *Essays and Surveys in Metaheuristics*, pages 469–492. Kluwer Academic Publishers, Norwell, MA, 2002.

[82] Mester, D. et Bräysy, O. Active guided evolution strategies for large scale vehicle routing problems with time windows. *Computers & Operations Research*, 32:1593–1614, 2005.

[83] Mladenović, N. et Hansen, P. Variable Neighborhood Search. *Computers & Operations Research*, 24:1097–1100, 1997.

[84] Narazaki, S., Yamamura, H. et Yoshida, N. Strategies for selecting communication structures in cooperative search. *International Journal of Cooperative Information Systems*, 4(4):405–422, 1995.

[85] C. Rego. A Subpath Ejection Method for the Vehicle Routing Problem. *Management Science*, 44:1447–1459, 1998.

[86] C. Rego. Node Ejection Chains for the Vehicle Routing Problem : Sequential and Parallel Algorithms. *Parallel Computing*, 27:201–222, 2001.

[87] Rochat, Y. et Taillard, É.D. Probabilistic Diversification and Intensification in Local Search for Vehicle Routing. *Journal of Heuristics*, 1(1):147–167, 1995.

[88] Rousseau, L.-M., Gendreau, M. et Pesant, G. Using Constraint-based Operators with Variable Neighborhood Search to Solve the Vehicle Routing Problem with Time Windows. *Journal of Heuristics*, 8(1):43–58, 2002.

[89] Schilham, R. M. F. *Commonalities in local search.* Ph.D. Thesis, Eindhoven University of Technology, The Netherlands, 2000.

[90] Schulze, J. et Fahle, T. A parallel algorithm for the vehicle routing problem with time window constraints. *Annals of Operations Reseach*, 86:585–607, 1999.

[91] P. Shaw. Using constraint programming and local search methods to solve vehicle routing problems. *Lecture Notes in Computer Science*, 1520:417–431, 1998.

[92] So, Y.-P. et Durfee, E.H. A distributed problem-solving infrastructure for computer network management. *International Journal of Intelligent &S Cooperative information systems*, 1(2), June 1992.

[93] Solomon, M.M. Time Window Constrained Routing and Scheduling Problems. *Operations Research*, 35:254–265, 1987.

[94] Solomon, M.M. et Desrosiers, J. Time window constrained routing and scheduling problems. *Transportation Science*, 22:1–13, 1988.

[95] Sycara, K.P., Roth, S.F., Sadeh, N. et Fox, M.S. Distributed constrained heuristic search. *IEEE Trans. Syst. Man Cyberm. (Special issue on Distributed AI)*, 21(6):1446–1461, 1991.

[96] É.D. Taillard. Robust taboo search for the quadratic assignment problem. *Parallel Computing*, 17:443–455, 1991.

[97] É.D. Taillard. Parallel taboo search techniques for the job shop sheduling problem. *ORSA Journal on Computing*, 6(2):108–117, 1994.

[98] Taillard, É.D. *Recherches itératives dirigées parallèles.* Thèse de doctorat, École Polytechnique Fédérale de Lausanne, 1993.

[99] Taillard, É.D., Badeau, P., Gendreau, M., Guertin, F. et Potvin, J.-Y. A Tabu Search Heuristic for the Vehicle Routing Problem with Soft Time Windows. *Transportation Science*, 31(2):170–186, 1997.

[100] Taillard, É.D., Gambardella, L.M., Gendreau, M. et M., Potvin, J.-Y. Programmation à mémoire adaptative. *Calculateurs Parallèles, Réseaux et Systèmes répar tis*, 10:117–140, 1998.

[101] P. Toth et D. Vigo. Exact Solution of the Vehicle Routing Problem. Dans T.G. Crainic et G. Laporte, éditeurs, *Fleet Management and Logistics*, pages 1–31. Kluwer Academic Publishers, Norwell, MA, 1998.

[102] P. Toth et D. Vigo, éditeurs. *The Vehicle Routing Problem*, volume 9 de *SIAM Monographs on Discrete Mathematics and Applications.* SIAM, 2002.

[103] M. Toulouse. *Heuristiques parallèles de recherche sans contrôle global explicite.* Thèse de doctorat, École Polythecnique, Université de Montréal, Montréal, Canada, 1996.

[104] M. Toulouse. *Étude d'heursitiques distribuées de recherche sans contrôle global explicite.* Thèse de doctorat, Université de Montréal, C.P. 6128 Succ. Centre-Ville, Montréal, Qc, H3C 3J7, 1996.

[105] Toulouse, M., Crainic, T.G. et Gendreau, M. Communication Issues in Designing Cooperative Multi Thread Parallel Searches. Dans I.H. Osman et J.P. Kelly, éditeurs, *Meta-Heuristics : Theory & Applications*, pages 501–522. Kluwer Academic Publishers, Norwell, MA, 1996.

[106] Toulouse, M., Crainic, T.G., Sansó, B. et Thulasiraman, K. Self-Organization in Cooperative Search Algorithms. Dans *Proceedings of the 1998 IEEE International Conference on Systems, Man, and Cybernetics*, pages 2379–2385. Omnipress, Madisson, Wisconsin, 1998.

[107] S. Wolfram. Computation theory of cellular automata. *Comm. Math. Phys*, 96:15–57, 1984.