José Eduardo Pécora Junior

Iterative Restricted Space Search: a solving approach based on hybridization

Thèse présentée
à la Faculté des études supérieures de l'Université Laval
dans le cadre du programme de doctorat en sciences de l'administration
pour l'obtention du grade de Philosophiæ Doctor (Ph.D.)

Faculté des sciences de l'administration
UNIVERSITÉ LAVAL
QUÉBEC

2008

José Eduardo Pécora Junior

Iterative Restricted Space Search: a solving approach based on hybridization

Thèse présentée
à la Faculté des études supérieures de l'Université Laval
dans le cadre du programme de doctorat en sciences de l'administration
pour l'obtention du grade de Philosophiæ Doctor (Ph.D.)

Faculté des sciences de l'administration
UNIVERSITÉ LAVAL
QUÉBEC

2008

Résumé

Face à la complexité qui caractérise les problèmes d'optimisation de grande taille, l'exploration complète de l'espace des solutions devient rapidement un objectif inaccessible. En effet, à mesure que la taille des problèmes augmente, des méthodes de solution de plus en plus sophistiquées sont exigées afin d'assurer un certain niveau d'efficacité. Ceci a amené une grande partie de la communauté scientifique vers le développement d'outils spécifiques pour la résolution de problèmes de grande taille tels que les méthodes hybrides. Cependant, malgré les efforts consentis dans le développement d'approches hybrides, la majorité des travaux se sont concentrés sur l'adaptation de deux ou plusieurs méthodes spécifiques, en compensant les points faibles des unes par les points forts des autres ou bien en les adaptant afin de collaborer ensemble. Au meilleur de notre connaissance, aucun travail à date n'à été effectué pour développer un cadre conceptuel pour la résolution efficace de problèmes d'optimisation de grande taille, qui soit à la fois flexible, basé sur l'échange d'information et indépendant des méthodes qui le composent.

L'objectif de cette thèse est d'explorer cette avenue de recherche en proposant un cadre conceptuel pour les méthodes hybrides, intitulé la recherche itérative de l'espace restreint, «Iterative Restricted Space Search (IRSS)», dont, la principale idée est la définition et l'exploration successives de régions restreintes de l'espace de solutions. Ces régions, qui contiennent de bonnes solutions et qui sont assez petites pour être complètement explorées, sont appelées espaces restreints «Restricted Spaces (RS)». Ainsi, l'IRSS est une approche de solution générique, basée sur l'interaction de deux phases algorithmiques ayant des objectifs complémentaires. La première phase consiste à identifier une région restreinte intéressante et la deuxième phase consiste à l'explorer. Le schéma hybride de l'approche de solution permet d'alterner entre les deux phases pour un nombre fixe d'itérations ou jusqu'à l'atteinte d'une certaine limite de temps.

Les concepts clés associées au développement de ce cadre conceptuel et leur validation seront introduits et validés graduellement dans cette thèse. Ils sont présentés de manière à permettre au lecteur de comprendre les problèmes que nous avons rencontrés en cours de développement et comment les solutions ont été conçues et implémentées. À cette fin, la thèse a été divisée en quatre parties. La première est consacrée à la synthèse de l'état de l'art dans le domaine de recherche sur les méthodes hybrides. Elle présente les principales approches hybrides développées et leurs applications. Une brève description des approches utilisant le concept de restriction d'espace est aussi présentée dans cette partie.

La deuxième partie présente les concepts clés de ce cadre conceptuel. Il s'agit du processus d'identification des régions restreintes et des deux phases de recherche. Ces concepts sont mis en œuvre dans un schéma hybride heuristique et méthode exacte. L'approche a été appliquée à un problème d'ordonnancement avec deux niveaux de décision, relié au contexte des pâtes et papier : «Pulp Production Scheduling Problem».

La troisième partie a permit d'approfondir les concepts développés et ajuster les limitations identifiées dans la deuxième partie, en proposant une recherche itérative appliquée pour l'exploration de RS de grande taille et une structure en arbre binaire pour l'exploration de plusieurs RS. Cette structure a l'avantage d'éviter l'exploration d'un espace déjà exploré précédemment tout en assurant une diversification naturelle à la méthode. Cette extension de la méthode a été testée sur un problème de localisation et d'allocation en utilisant un schéma d'hybridation heuristique-exact de manière itérative.

La quatrième partie généralise les concepts préalablement développés et conçoit un cadre général qui est flexible, indépendant des méthodes utilisées et basé sur un échange d'informations entre les phases. Ce cadre a l'avantage d'être général et pourrait être appliqué à une large gamme de problèmes.

MOTS CLÉS: Cadre conceptuel flexible, méthodes hybrides, méthodes heuristiques, méthodes exactes, problèmes d'ordonnancement, problèmes de localisation et d'allocation.

Abstract

When dealing with very large problems attempting to explore the whole solution space is an unattainable goal. Moreover, as the size of the problem increases more sophisticated solving methods are required to maintain a certain level of efficiency. This has lead to the development of specific tools, such as hybrid methods. Despite all the efforts and advances in the development of hybrid approaches, most researchers focused on how to exploit the strengths and weakness of two specific methods while adapting them to work together. Until now, little has been done to develop a generic framework that is independent of the component methods and has a foundation based on the exchange of information.

This thesis will explore this gap in the development of hybrid methods by proposing a framework, named the Iterative Restricted Space Search (IRSS), for the design of efficient hybrid search algorithms. The main idea of the IRSS is to iteratively define and explore restricted regions – subsets of the global solution space – that have a high potential of containing good solutions and yet are small enough to be thoroughly explored. These regions are named Restricted Spaces (RS). IRSS is a generic search approach based on the interaction of two algorithmic phases having complementary goals. The first phase identifies a restricted space and the second phase explores it. The algorithm alternates between the two phases for a fixed number of iterations or until the allotted time has expired.

Several concepts and ideas related with the development of this framework are presented in chronological order, leading to a natural increase in the complexity of the concepts and also allowing the reader to understand the problems that we encountered and how the solutions were conceived and implemented. This thesis is therefore divided into four parts: the first is dedicated to a description of the literature on hybrid solution methods. It presents a review of the most recent hybrid methods and applications of this special class of solving methods. In addition, solving techniques that use the "space restriction" concept are described.

The second part introduces the main concepts of this framework, such as the identification of the Restricted Space and the two-phase search. These ideas are exploited using a hybrid heuristic-exact method applied to the Pulp Production Scheduling Problem, a difficult real world problem composed of two decision levels.

The third part extends the recently developed concepts and also repairs the limitations identified in the second part. This is done by proposing an incremental search procedure to deal with large RS and a binary tree to structure the exploration of several RS. This structure has several advantages, among which it avoids the double search and provides a natural diversification for the framework. This new extended method is applied to a real-life location-allocation problem using an iterative hybrid heuristic-exact algorithm.

The fourth part generalizes the concepts previously developed and conceives a framework that is flexible, independent of the component methods used, and based on the exchange of information between its components. This final part also offers my conclusions and possible future research topics related to this thesis.

KEYWORDS - Generic Framework, Hybrid Methods, Heuristic Methods, Exact Methods, Real-Life Applications, Pulp Production Scheduling Problem, Location - Allocation Problem

Resumo

Em pesquisa operacional, tratando-se de problemas de grande escala, a exploração completa do espaço de soluções é um objetivo praticamente inalcançável. Além disso, observa-se que quanto maior o tamanho do problema, mais sofisticado deve ser o método para se manter certos níveis de eficiência, levando ao desenvolvimento de ferramentas específicas, como por exemplo os métodos híbridos. Apesar de todos os esforços e avanços no desenvolvimento dos métodos híbridos, onde a maioria dos trabalhos foca na adaptação de dois ou mais métodos específicos, compensando os pontos fracos de um, com os pontos fortes de outros, e.g., a abordagem de intensificação/diversificação, pouco foi feito para se desenvolver um "framework" que seja genérico, independente dos métodos componentes e que tenha como base uma sólida troca de informações.

Esta tese explora esta abertura no desenvolvimento dos métodos híbridos, propondo um framework, nomeado "Iterative Restricted Space Search" (IRSS), onde a principal idéia é a identificação e exploração sucessivas de subespaços restritos do espaço de soluções. Duas características são necessárias à definição destes subespaços: além de terem uma alta possibilidade de conter boas soluções devem ser pequenos o bastante para serem completamente explorados. O IRSS é um algoritmo de busca genérico baseado na interação de duas fases de busca com objetivos complementares. A primeira fase faz uma ampla exploração do espaço de soluções identificando sub-regiões que sejam promissoras e a segunda fase explora essas sub-regiões. O algoritmo alterna entre as duas fases por um determinado número de vezes ou até terminar o tempo total.

Os diversos conceitos e idéias relativos a esta pesquisa são apresentados em ordem cronológica. Para este fim, a tese está dividida em quatro partes:

A primeira é dedicada à descrição atual do desenvolvimento em hibridação, apresentando uma revisão dos métodos híbridos atuais e aplicações que utilizam métodos híbridos na sua resolução.

Na segunda parte, os principais conceitos do framework, como a identificação das regiões restritas e as duas fases de busca são apresentados. Estes conceitos são implementados na forma de um método híbrido heurístico-exato aplicado a um problema de alocação de recursos em usinas de celulose (Pulp Production Scheduling Problem), um problema real contendo dois níveis de decisão.

A terceira parte estende os conceitos recentemente desenvolvidos e repara as limitações identificadas na segunda parte, propondo uma busca incremental aplicada na exploração de grandes espaços restritos e uma estrutura em árvore binária para a exploração de diversos espaços. Esta estrutura tem a vantagem de evitar a exploração de um espaço já explorado anteriormente provendo uma diversificação natural ao framework. Este novo método expandido é aplicado a um problema localização e alocação utilizando um método iterativo híbrido heurístico-exato.

A quarta parte generaliza os conceitos previamente desenvolvidos e concebe o framework o qual é flexível, independente dos métodos componentes utilizados, baseado em uma sólida troca de informações entre as fases e pode ser aplicado a uma grande gama de problemas. Nesta parte final encontram-se também as conclusões e possíveis pesquisas futuras utilizando os conceitos relatados nesta tese.

PALAVRAS CHAVE: Framework Genérico, Métodos Híbridos, Métodos Heurísticos, Métodos Exatos, Aplicações à Problemas Reais, Problema de Alocação de Recursos para à Produção de Celulose, Problema de Localização-Alocação

Acknowledgments

COMPLETING a Ph.D. is always a difficult task and it would be unthinkable without the help and support of my professors, colleagues and friends, without whom this thesis would never be concluded.

Without any doubt, the most important person in this quest is my advisor (and friend) Professor Angel Ruiz, who has been a wonderful mentor. His insights and thoroughness were of great value to the development of this research. I sincerely thank him for his great availability, trust and knowing when to listen me as my friend and when to be my advisor. I was also honored to have Professor Patrick Soriano as my co-advisor, a person whose experience is so large that I am sure does not matter how many times we meet and discuss I will always have something to learn from him. I am also grateful to Professors Jacques Renaud and Fayez Boctor who were present at each step of this endeavor. I show all my appreciation also to Professors Alain Martel and Roberto Wolfler Calvo for their suggestions which helped to enhance the main aspects of this thesis.

Quoting Vinicius de Moraes – Brazilian composer, poet and bohemian – "We don't make friends, we recognize them". I thank Walid and Fabien, not only for the great friends they became, but also for the help in so many different occasions, from cheering up to philosophical discussions, drinking coffee and all the times we played pool together. I could not forget to sincerely thank all the friends who I met along the way, in particular, Alysson, Gisele and Luis.

The administrative personnel from the Ph.D program at Laval University deserves all my gratitude, in especial Marie-Claude Beaulieu, Karine Barquin and Johanne Nadeau. My warmest thanks also to Brigitte Riverin and Louise Doyon from Operations and Decision Systems Department and Cirrelt respectively.

I want to offer heartfelt thanks to my parents, Eladir and José Eduardo, and my family Eloá, Vernihuzão, Vernihuzinho, Roldão and Maria Lúcia for being there for me (us) and for giving all the long distance support they could throughout this long journey.

The financial support provided by CAPES-MEC in Brazil and NSERC in Canada are gratefully acknowledged too.

A very special place, both in this acknowledgement and in my heart, goes to Ana Tereza, Giovanna and the Baby. For being on my side at each step of this odyssey, supporting, encouraging and smiling to me when I was in most need of it. My time in Quebec would not have been half as good without you.

Finally, I thank God to make everything possible.

Foreword

THIS thesis is the capitalization of four years of work as a Ph.D. student at CIRRELT *Centre interuniversitaire sur les réseaux d'entreprise, la logistique et le transport* under the supervision of Professors Angel Ruiz and Patrick Soriano. My contribution was to propose the research orientation to the development of a generic framework for hybrid methods and the conception of the hybrid solution methods based on the framework for the two problems considered in this document. Plays also an important role in my contribution, the programming work, conception and validation of data tests and examples, a first draft of each paper and the preliminary interpretations of the computation results obtained.

This project is composed of three main parts, the first is dedicated to the description of the Pulp Production Scheduling Problem and also the hybrid solution method applied to it, in the second part the proposed solution approach plus an extension is applied to a Location-Allocation problem and the third part is dedicated to the generalization of these hybrid approaches into a framework.

The article *"Minimization of the wood density variation in pulp and paper production"* presents the Pulp Production Scheduling Problem. This work was presented in "6ème Congrès International de Génie Industriel", Besançon, France, 2005. It is published in INFOR, Vol 45(4), 2007, pp 187-196.

The article *"Restricted Space Search Heuristic: Application to a Pulp Production Scheduling Problem"* is the application of proposed solution method into the Pulp Production Scheduling Problem, described in the first article. It was presented in the MIC05 - "6th Metaheuristic International congress", 2005, Vienna, Austria. It should be ready for submission this Winter.

The article *"Iterative Restricted Space Search to solve a real-life location-allocation problem"*, extends the ideas of the second paper to a generic iterative solving approach and applies them to a location-allocation problem. It was presented in the MIC07 -

"7th Metaheuristics International Conference", 2007, Montreal, Canada; META 08 - "International Conference on Metaheuristics and Nature Inspired Computing", 2008, Hammamet, Tunisia; and it should be ready for submission this Winter.

The article *"Iterative Restricted Space Search: A Framework for designing efficient search algorithms"* is a generalization of these solution methods. It was presented in the MIC07 - "7th Metaheuristics International Conference", 2007, Montreal, Canada; META 08 - "International Conference on Metaheuristics and Nature Inspired Computing", 2008, Hammamet, Tunisia; This article should be ready in Summer 2009.

To Giovanna our "estrelinha" and all other "estrelinhas" yet to come that will make our sky even brighter!

"There are no nations! There is only humanity. And if we don't come to understand that right soon, there will be no nations, because there will be no humanity."

Isaac Asimov (1920-1992)

"You must not lose faith in humanity. Humanity is an ocean; if a few drops of the ocean are dirty, the ocean does not become dirty. "

Mahatma Gandhi (1869-1948)

Contents

List of Tables

List of Figures

List of Abbreviations

B&B	Branch and Bound
CCA	Cooperative Coevolutionary Algorithms
GA	Genetic Algorithms
GRASP	Greedy Randomized Adaptive Search Procedure
GTS	Granular Tabu Search
IRSS	Iterative Restricted Space Search
MIP	Mixed Integer Programming
OR	Operational Research
PR	Path Relinking
RINS	Relaxation Induced Neighborhood Search
RS	Restricted Space(s)
SA	Simulated Annealing
SS	Scatter Search
TS	Tabu Search
VND	Variable Neighborhood Descent
VNS	Variable Neighborhood Search
VRP	Vehicle Routing Problem

Introduction

Solving very large, real-life problems to optimality remains probably one the most important challenges in Operational Research (OR). The formidable computational resources currently available have greatly contributed in pushing the limits of exact methods but, as computation performance increases, so too does the size or complexity of the problems we seek to solve. Two approaches have shown impressive advances to tackle these large problems in recent years: (1) classical mathematical manipulations aiming at transforming or reducing the original problem into smaller and simpler problem(s) to solve – e.g. decomposition, linearization, relaxation and projection procedures, among others – and (2) improving or refining solution methods.

Exact methods have been improved with pre-processing techniques, cuts and column generation and Mixed Integer Programming (MIP) heuristics. Heuristic methods have also made important contributions: Tabu Search (TS), Genetic Algorithm (GA) and Variable Neighborhood Search (VNS) among others have been successfully applied to problems whose size makes them intractable by means of exact methods. Despite this progress, as the size of the problems increases, more sophisticated solving methods are required to maintain a certain level of efficiency. This has lead to the development of specific tools such as hybrid methods.

Hybrid methods have been successfully applied to several problems in Operational Research during the last few decades. This success is mostly due to the adaptability and flexibility inherent to this class of solution approaches. *A priori*, using different methods to solve a single problem has the advantage of exploring the solution space from different perspectives that complement each other, e.g. in an intensification/diversification paradigm. Therefore, most of the works in the literature that propose hybrid algorithms focus on how to explore the strengths and weakness of two specific methods adapting them to work together. However, little has been done to develop a generic framework independent of the specific methods to be merged. Despite the component methods used in the hybridization, some information must be exchanged or shared between them. Information exchange is a key issue when designing solving methods encompassing any form of hybridizing but, again, little has been done to develop and assess the performance of generic information sharing schemes.

The objective of this thesis is to propose a framework for the design of efficient search algorithms that is independent of its components methods and has a foundation built on the exchange of information.

Available search methods are very efficient when applied to small regions of the solution space. Thus, if one is able to reduce the scope of the search by reducing or constraining the solution space, a thorough yet fast search can be achieved. In other words, an efficient search strategy should consist in defining one or several *promising subspaces*, small enough to be thoroughly explored in reasonable time but with a high likelihood of containing near-optimal (hopefully the optimal) solutions. This diversification/intensification paradigm led us to design a hybrid algorithmic structure encompassing two main phases. The first phase performs a "macro" search covering the solution space as much as possible and identifies the *promising subspaces*, while in the second phase a "micro" search or the thorough exploration of the *promising subspaces* is performed.

A more formal definition of these *promising subspaces*, called Restricted Space (RS) in the remaining of this thesis, is:

> **Retricted Space (RS)** is a subspace of the universal set of solutions which has two highly desirable characteristics: (1) it should be small enough to be thoroughly explored and (2) it should have a high possibility of containing near-optimal solutions.

When dealing with complex OR problems, it is very difficult to have quantitive measure of *a high possibility of containing near-optimal solutions* without explore the restricted space to optimality. Therefore we use all the information available – including a set of local optima points, bounds and the history of search – to construct a RS which we believe that has *a high possibility of containing near-optimal solutions.* As will be detailed latter, the RS guarantee by construction always contains a set of local optimal solutions. Thus it always provides in the worst case a solution of the same quality as the past solutions visited.

The identification of the RS, done by the "macro" search, is one of the main challenges of this work. Although several alternative methods to identify the RS are possible, the following general features are expected in the method used to find it: (1) The method should be able to explore as much of the solution space as possible. Indeed, such a method should probe or should sample the solution space, and provide a rough idea of the quality of the solutions that could be expected in different regions of the solution space. (2) The method should focus on identifying the common characteristics of good solutions rather than on searching for the best solution. (3) The method should be able to rank these potential regions by their expected (or estimated) objective function. (4) The method should be efficient, meaning that it should evaluate the potential of a given region accurately in reasonable time. To our knowledge there are no heuristics or exact methods able to fully satisfy these requirements on their own. Therefore, we propose the use of a hybrid approach, which the main concepts are developed through the several chapters of this thesis and it is completely described and formalized in Chapter 5.

The search strategy depicted in the previous paragraphs requires the identification and exploration of several RS, performed by the macro and micro search respectively. Among several available possibilities to structure the relationship between the macro and the micro searches, an iterative structure was used to maximize the efficiency of the overall search. In this way, an iteration consists in the identification and exploration of an RS. From one iteration to the next, all information gathered concerning regions already explored (local optima, bounds, etc...) is used to guide the next searches. Therefore, instead of proposing a multi-start approach that simply generates one RS at each iteration, we chose to structure the search procedure using a binary tree that, among other advantages, excludes any previously searched region from the target solution space. This avoids double search and helps to provide a natural diversification for the search.

Two problems were chosen to illustrate the implementation of the Framework. The first, called the Pulp Production Schedule Problem, Chapters 2 and 3, is a real-life problem which arises in the context of the transportation of wood logs for pulp production. The second problem, described in Chapter 4, is a version of the classical Location-Allocation Problem in the context of an international transportation company that exploits an import/export network. Although the two problems are clearly different, they share several characteristics that made them interesting for use in our framework. First, both problems model complex real-life situations. Second, both present two different decision levels. Finally, they are formulated as large scale Mixed Integer Programs difficult to solve.

This thesis is the result of almost four years of work and developments. Therefore, we believe that it is worth presenting our work according to its chronological evolution, from the original idea to the current algorithmic scheme. This approach leads to a natural increase in the complexity of the concepts and also allows the reader to understand the problems that we encountered and how the solutions were conceived and implemented.

The thesis is divided into four parts. Each part consists of one chapter except for part two, which includes two chapters. Part I reviews the most recent hybrid solution approaches and also includes different approaches which use the concept of space restriction. This literature review does not pretend to cover all existing hybrid algorithms, but to evoke the most important relevant concepts. However, it is important to note that problem specific literature reviews can be found in their respective part of the thesis.

The second part presents two articles coping with the Pulp Production Scheduling Problem. The first article, Chapter 2, describes the industrial context of the problem, proposes a mathematical formulation and a heuristic solution approach. The second article, Chapter 3, introduces some of the concepts of the Framework, such as the RS, the macro and micro searches and applies them to the Pulp Production Scheduling Problem in the form of a hybrid heuristic-exact method.

The third part, Chapter 4, has two objectives. First, it introduces a real-life location-allocation problem. Second, it extends the ideas underlying the hybrid method used to solve the Pulp Production Scheduling Problem and removes its drawbacks by proposing an incremental search procedure to deal with large RS, and a binary tree to structure the exploration of several RS, leading to a new iterative method called Iterative Restricted Space Search - IRSS.

The fourth part, Chapter 5, generalizes the concepts previously developed and presents the full framework. The complete framework, including all its extensions, is applied again to the two proposed problems in order to assess the contribution of the different enhancements. This chapter also discusses important issues concerning the implementation of the framework, for example, the trade-off between exploring a few large RS or many smaller RS.

Finally, Chapter 6 offers our conclusions and outlines several future possible research avenues related to this thesis.

Part I

The Literature

Considerate la vostra semenza: fatti
non foste a viver come bruti, ma per
seguir virtute e canoscenza.

Dante Alighieri (1265-1321)
La divina commedia
Canto XXVI, lines 118-120

Chapter 1

Literature Review

This literature review presents the theoretical foundation of this thesis. To this end, it is divided into two sections, the first is dedicated to the hybrid solution approaches and the second talks about the space restriction methods.

Every hybrid method is composed of more than one component method (n.b. or one single solution method used several times, such as the parallel algorithms). The interaction between these component methods differentiates hybrid methods from multi-start approaches. Usually, different methods are associated with one another to complement the weakness of one with the strengths of the other and this association is made through any kind of exchange of information. Therefore, studying and proposing hybrids which focus on a solid exchange of information will intuitively lead to more efficient solution approaches. To this end, a section in this literature review depicts the information exchange protocol of the hybrid methods present in the recent literature. As we chose tow real-life models to illustrate the IRSS, a section describing hybrid methods applied to real-life problems is included.

As the literature on hybrid methods is very rich and vast, it is beyond the scope of this thesis to make an exhaustive review of all hybrid approaches or to formalize and classify them. For such matters, the interested reader is referred to the works of Talbi (2002), which provides an extensive classification of hybrid methods using heuristics and Puchinger and Raidl (2005) on hybrid heuristic-exact methods.

Within the operational research literature, the term "hybrid solution approach" is generally applied to the mixing of two or more different solution methods. The goals of this approach are to encourage the exploration of new search regions, escape the local attraction of optimal points, and generate cuts and/or columns, among others.

However, it can be argued that the fundamental principle of the hybrid framework proposed in this thesis differs somewhat from the principles of the majority of hybrid approaches have been based on up to now. As briefly explained in the introduction we use a hybrid approach to identify a potential region of the space, which we call Restricted Space. As the restriction of the search space is a fundamental idea in the framework proposed by this thesis, a section that introduces the various approaches that use any kind of space restriction is also included.

Both fields are fundamental in the development of this thesis. First, hybrid methods will be explored, followed by approaches that use any kind of space restriction during their execution.

1.1 Hybrids

Hybrid solution methods belong to a relatively new, but very broad domain in Operational Research. Essentially, any two solution methods can be put together to create a hybrid method. The first articles about hybrid methods can be tracked to the beginning of the 1980s. One of the first published articles which explicitly uses the expression *hybrid* is the work of Langston (1982) where two simple heuristic methods – one greedy and one improvement – are used sequentially to assign jobs to a set of identical machines to minimize the finish time.

Traditionally, hybrid approaches have merged heuristic and/or exact algorithms (Gallardo et al., 2007). Exact methods are a special class of solution approaches that guarantee optimality, but they can become computationally intractable for many combinatorial optimization problems, especially NP-Hard problems (Manber, 1989). Exact methods however can be very efficient when dealing with small to medium sized problems. Examples of these methods are: Branch and Bound (Land and Doig, 1960), and its variations, Branch and Cut, Branch and Price, Branch and Cut and Price (Barnhart et al., 2000 ; Nemhauser and Wolsey, 1988) and Dynamic Programming (Wolsey, 1998). If we remove the optimality condition, we arrive in the domain of heuristic methods. These methods are known to be very efficient, even when dealing with NP-Hard problems. On the other hand, heuristics are able to find good solutions in a relatively short computational time and they can be specially designed to profit from special structures of a problem. Examples of heuristics methods belonging to the constructive and neighborhood based class are: the descent local search (Papadimitriou and Steiglitz, 1982), greedy heuristic (Lawler, 1976), simulated annealing (SA) (Kirkpatrick et al., 1983), tabu search (TS) (Glover and Laguna, 1997), examples belonging to the population based class are: genetic algorithms (GA) (Holland, 1975), ant colonies (AC) (Dorigo et al., 1996), scatter search (SS) and path relinking (PR)(Glover, 1977) among others. Glover and Kochenberger (2003), Blum and Roli (2003) and Voβ et al. (1999) present interesting overviews of the main metaheuristics developed in recent years.

Recent research on hybrid methods involve simulation (Peng et al., 2006), constraint programming (Correa et al., 2004 ; Hooker, 2006), neural networks (Sahoo and Maity, 2007 ; Pendharkar, 2005) and multi-agent (Yan and Zhou, 2006) approaches. Also, this special class of solution methods has been applied to a wide range of operational research problems. These have achieved very good results in classical problems, such as the knapsack problem (da Silva et al., 2007), the Traveling Salesman Problem (Nguyen et al., 2007), the p-median location problem (Resende and Werneck, 2004) as well as on real world applications such as, health care (Bertels and Fahle, 2006 ; Pécora, 2002), the optimization of credit portfolios (Schlottmann and Seese, 2004) and aircraft scheduling, (Gronkvist, 2006) among others.

The next subsection presents some recent articles dealing with hybrid methods emphasizing the communication protocol used to connect the component methods.

Communication protocols within the hybrid methods

Despite the number of possible combinations all hybrid methods have one thing in common, the component methods communicate with each other using a protocol. To structure this review, we propose a communication protocol that defines the type of information (Homogeneous/Heterogeneous and Raw/Preprocessed) and how this information is passed through the methods (Pipeline/Cyclic/Arbitrary). These communication protocols are defined as: **pipeline**, where each component method is executed only once and the information is passed in sequence from the first to the last without feedback; **cyclic**, where the information is passed in sequence from the first to the last method which returns the information to the first method completing the cycle; **arbitrary**, where there is no fixed order to transfer the information, usually all methods send information to a single pool which is available and visible to all. The information can be **homogeneous**, when all methods share the same type of information, (e.g. the best solution found) or **heterogeneous**, where the type of information differs from method to method. Also, the information can be **preprocessed** (e.g., sorted, classified, modified) before being transferred or can be transferred in its **raw** format.

Pipeline / Cyclic / Arbitrary

Examples of pipeline communication are: Langston (1982), where two heuristics – one constructive and one improvement – are used in sequence to assign jobs to parallel machines. The constructive heuristic generates a partial solution that is passed as information to the improvement heuristic, which completes the solution. Perez et al. (2005) proposed a Hybrid GRASP and Path Relinking (PR) with a pipeline information flow. The GRASP is used to generate an initial population, then all the individuals belonging to this initial population are combined two by two using the PR, and the algorithm finishes returning the best solution found. Delmaire et al. (1999) presented a GRASP and Tabu Search hybrid for the single-source capacitated facility location problem where the GRASP generates an initial population that is improved by the TS. Alvim et al. (2004) proposed a hybrid for the one-dimensional bin packing problem, their algorithm has five phases (i.e., reduction, bounds, construction, redistribution and improvement) passing information sequentially in a pipeline structure. Despite being called several times, using multi-start approach the algorithm is classified as a pipeline information exchange algorithm instead of cyclic as there is no information going from the last to the first phase.

Lapierre et al. (2004) used a cyclic exchange of information between a Tabu Search and a Variable Neighborhood Search (VNS) hybrid to solve a location-allocation problem. The neighborhood developed for this application uses a pseudo-random sampling technique to reduce the number of solutions to be evaluated but retains the possibility of concentrating the shipment in a specific hub. Their hybrid algorithm is essentially a Tabu Search, in which the VNS chooses the neighborhood to be explored based on the search status.

Almost every hybrid that uses parallelization has arbitrary communication. Examples of this type of communication can be found in Crainic and Gendreau (2002) for the fixed charge capacitated multi-commodity network design problem, where a series of Tabu Search metaheuristics are executed in a parallel cooperative scheme. Each Tabu Search updates a pool of solutions every time it reaches a local optimal point and can request information from the pool at a moment determined by a series of six import criteria. Gendron et al. (2003) presented a parallel hybrid heuristic for the multi-commodity capacitated location problem. This hybrid, which combines Variable Neighborhood Descent (VND) (Hansen and Mladenović, 2001) and Slope Scaling (Kim and Pardalos, 2000), is based on adaptive memories that are updated by the two metaheuristics in running time and is also responsible for sending new starting solutions whenever the solution method needs one. An example of arbitrary communication that is not a parallel algorithm is the core and shell framework introduced by Jain and

Meeran (2002). Their approach is a hierarchical search method based on the tradeoff between intensification and diversification. The innermost search method, the core, is responsible for intensification while the several shell algorithms have an increased degree of diversification. The communication between the several shells and the core is made through a set of elite solutions available to all search algorithms.

Heterogeneous / Homogeneous

Homogeneous information exchange is very common, e.g., every hybrid that only exchanges solutions is classified as homogeneous.

The Cooperative Coevolutionary Algorithms (CCA), (Potter and De Jong, 1994 ; Wiegand, 2003), are good examples of heterogeneous information exchange. The CCA imitates biological coevolution using a series of reciprocal changes between two or more distinct populations. In CCA a complete solution is an aggregation of several parts of solutions, where each part is defined as a species that is tackled by one evolutive method. Therefore, each species evolves separately and the interaction between the species is performed when they are aggregated to become the whole solution. These solution methods have heterogeneous information because each evolutive method contributes with a different part of a solution. French et al. (2001) proposed a heterogeneous exchange of information between a GA and a Branch and Bound (B&B) for the MAX-SAT problem. The hybrid starts with the B&B that introduces potential solutions in a GA population. When the control of the hybrid is passed to the GA it is executed for several generations to improve the initial population. When the GA finishes, the control of the hybrid is passed back to the B&B but the decisions to choose the branching node, branching variable and branching direction are made using the information extracted from the individual solutions generated during the GA phase. Subsequently, the algorithm alternates between the two approaches until the stop criterion is reached. Budenbender et al. (2000) proposed a hybrid TS - B&B for the Direct Flight Network Design Problem. The intuition of this method is to use the information in the tabu list and a set of rules to find a subset of solutions to be explored by the B&B. In this hybrid the information exiting the TS is the tabu list and the returning information is the best solution found inside the given region.

Raw / Preprocessed

Passing the information in its raw format is the most common approach, all the examples listed above, except for Budenbender et al. (2000), fall into this category.

Budenbender et al. (2000) proposed a hybridization between a TS and a B&B, where the TS sends the tabu list which is **encoded** as a prohibition of branching in certain variables and the B&B sends back the best solution found. In this way the information exiting from the TS is modified before it arrives at the B&B. Another preprocessed information exchange can be found in Prins et al. (2007). They developed a Hybrid containing a Lagrangean Relaxation (LR) and a Granular Tabu Search (GTS) (Toth and Vigo, 2003) for the capacitated location routing problem (LRP). The GTS tackles the original capacitated LRP sending the best solution found to the LR which deals with a location problem. Therefore, the current LRP solution is **transformed** into a location problem by aggregating the customers into super-customers, only then the LR is executed to solve the problem. The solution resulting from the LR is disaggregated and passed back to the GTS. The GTS and the LR use a cyclic protocol communication limited by a number of cycles. The RINS method (Danna et al., 2005) consists in calling for a heuristic search during a Branch and Bound tree. At a given node the current relaxed solution and the incumbent (best known) solution are **compared** which define a neighborhood around the incumbent solution. This neighborhood is explored by RINS, which returns the best solution found to the B&B. Velarde and Laguna (2004) presents a hybrid between a linear method and TS for a scenario based approach for the Robust capacitated international sourcing problem. The linear relaxation of each decomposed scenario is solved and the shadow prices of the capacity constraints are **averaged**. Using this information a decision is made regarding the potential moves (i.e., open, close, swap). The most promising moves are implemented in the TS procedure that returns the best known solution to the linear method, restarting the cycle.

Hybrids applied to real-life problems

There are a number of examples of real life problems touching different fields in Operational Research. These problems range from aircraft scheduling to health care and also production problems, energy and water distribution and petrol production, to cite a few. All of these problems have several points in common. Real world problems are among the most difficult problems to solve in Operational Research, not just for their great size and high complexity, which usually is NP-Hard, but also because this class of problems has very specific constraints and modeling needs. For all these reasons, generic and straightforward solution approaches may not perform well for such problems. To efficiently solve them, the specific characteristics of the problems need to be exploited and the chosen solution approach needs to be adapted to it. The following paragraphs discuss some recent real-life problems that use hybrids as a solving procedure.

Ahuja et al. (2005) proposed a hybrid solution approach for the locomotive-assignment problem. This problem consists in assigning locomotives into a pre-planned train schedule, while assuring enough locomotive power to pull the train from the start point to the end destination. Their model incorporates real-life features such as train-to-train connections and consist bustings and consistency. A consist is said to be busted when a set of locomotives arriving with a train is broken into subsets to be reassigned to two or more trains. A solution is consistent when the same locomotive is assigned to the same train for a period of at least one week. In order to solve this problem the authors use a combination of decomposition techniques, integer programming and very-large scale neighborhood search (Ahuja et al., 2002) using a pipeline–homogeneous–raw protocol. The work of Vaidyanathan et al. (2008) extended this formulation by adding new constraints to the planning problem, such as incremental locomotive planning which helps to achieve a good solution with minimum changes to the original planning.

Atkin et al. (2007) proposed a hybrid metaheuristic based on TS and a constructive heuristic, named Path Assignment Heuristic, using a cyclic–heterogeneous–raw protocol to tackle runway scheduling at London Heathrow airport. Only one runway is used for all aircraft departures at Heathrow airport. A separation time – depending on the aircraft routes, weights and speeds – between two departures is required for each pair of aircraft at take-off to safety reasons. Also, the geometry of the runway at Heathrow airport imposes physical constraints not found in the academic literature. Their approach proposes a solution that guides the runway controllers in their decisions and can also be used to anticipate future problems in the schedule. Also in the domain of the aircraft scheduling, Gronkvist (2006) proposed a hybrid of Constraint Programming and Column Generation to the Tail Assignment Problem which consists in assigning each aircraft (identified by its tail number) to a flight, while considering real life constraints such as maintenance and landing restrictions and prohibited flight times for certain aircrafts at certain airports. Their method starts with a constraint programming that reduces the original search space, and then the linear relaxation is used to generate columns for the relaxed problem. Once the relaxed optimal is found a heuristic method is applied to find an integer solution. The tree methods exchange information using a Pipeline – Heterogeneous – Raw protocol.

Bertels and Fahle (2006) solved the home health care problem – i.e., visiting and nursing patients at their home – using a combination of constraint programming, tabu search and simulated annealing. This hybrid approach has two phases. In the first phase a set of initial solutions is generated by constraint programming and saved in a pool. In the second phase a heuristic method is chosen randomly, between a TS and SA, and used to improve a random solution from the pool. The improved solution takes the place of the original solution in the pool and a new pair (heuristic and solution) is chosen when the algorithm restarts. Despite using a pool of solutions and restarts, this algorithm is classified as a pipeline because the constraint programming is used only once to generate the initial solution. Also in the healthcare field, the physician scheduling problem (Carter and Lapierre, 1999), consists in determining the sequence of shifts which each physician will work considering several constraints such as, contracts, ergonomic rules and hospital demand. Pécora (2002) presented a combination of GA and TS, using a cyclic–homogeneous–raw protocol, to tackle a real-life instance arising in the General Jewish Hospital - Montreal. In this approach, after each GA crossover the offsprings are improved by a TS before being part of a new generation.

Borraz-Sanchez and Rios-Mercado (2005) tackled a real-life problem for the distribution of natural gas. When dealing with natural gas distribution through pipeline 3% to 5% of the total natural gas transported is spent to maintain pressure in the pipeline. As the quantity of natural gas that is transported is huge, even a marginal improvement can have a significant positive impact in the whole system. The main decisions for this problem are the flow of gas to meet the demand requirements and at which point of the pipeline to use natural gas to keep the right pressure inside the whole gas pipeline. The authors decompose the main problem into a sub problem containing just the pressure variables, which is solved with dynamic programming, while the second subproblem deals with the flow variables and is solved by a Tabu Search. The hybrid method provides improved results for all 11 instances given in the article and this improvement is more than 2% in 8 out of 11 instances. It is implemented using cyclic–heterogeneous–raw protocol.

Budenbender et al. (2000) presented a real world instance of the facility location problem based on the German postal service. This problem consists in transporting letter mail through airport terminals by deciding which airports will be used and how the freight is transported among the terminals. The authors propose a Hybrid Tabu Search and Branch and Bound, where the exact method is used to explore a neighborhood given by the metaheuristic. The protocol used here is Cyclic and Heterogeneous because the TS and the B&B iteratively exchange different types of information. The TS sends the tabu list and the B&B send the best solution found. The protocol is also preprocessed because the information exiting the TS – the tabu list – is transformed in a constraint before arrives to the B&B.

The water distribution problem consists in transporting clean water from the treatment plant to individual customers. This multi-objective problem has many decisions to make. It has to maintain adequate water pressure, maintain the disinfection level, ensure minimum pipe bursts and also minimize the costs of material, excavation and maintenance. Keedwell and Khu (2006) deal with this problem by applying a hybrid including a cellular automaton (von Neumann and Burks, 1966) and a GA; in this case, the cellular automata is responsible for seeding – to generate an initial population of solutions – and the GA – used as an intensification procedure – combines the solutions in the initial population with the goal of defining the Pareto optimal set of solutions. The solution method uses a pipeline–homogeneous–raw information exchange.

Tarantilis et al. (2008) presented a variant of the Vehicle Routing Problem often met in real-life problems. Their work aims at identifying optimal routes for a fleet that can have their capacity recharged at intermediary replenishment stations. The authors used a combination of four heuristics to tackle this problem. In the first phase a constructive heuristic is used to find an initial solution. In the second phase this solution is improved using a combination of Tabu Search and Variable Neighborhood Search. Finally, in the third phase a guided local search is used to eliminate low quality features from the final solution. This algorithm is tested with a benchmark improving the best known solution for several instances and it uses a pipeline–homogeneous–raw exchange of information.

As previously mentioned, the Framework proposed in this thesis has two hybridizations; one between the "macro" and "micro" searches and a second inside the "macro" search. Chapter 5 will demonstrate that this hybridization keeps the same information exchange protocol independently of the component methods implemented.

Table 1.1 shows the complete protocol for information exchange within hybrid methods for each article cited here. Note that in the case when the article uses more than two solution methods, only the two that we judge the most important are represented in this Table.

Reference	Method 1	Method 2	Information Protocol*
Ahuja et al. (2005)	MIP Method	Very Large Scaled Neighborhood	Pip-Hom-Raw
Alvim et al. (2004)	Constructive Heuristic	Improvement Heuristic	Pip-Het-Raw
Atkin et al. (2007)	TS	Constructive Heuristic	Cyc-Het-Raw
Bertels and Fahle (2006)	Constraint Programming	TS + SA	Pip-Hom-Raw
Borraz-Sanchez and Rios-Mercado (2005)	TS	Dynamic Programming	Cyc-Het-Raw
Budenbender et al. (2000)	TS	MIP method	Cyc-Het-Pre
Crainic and Gendreau (2002)	TS	TS	Arb-Hom-Raw
Danna et al. (2005)	MIP method	Truncated MIP method	Arb-Het-Pre
Delmaire et al. (1999)	GRASP	TS	Pip-Hom-Raw
French et al. (2001)	GA	Branch and Bound	Cyc-Het-Raw
Gendron et al. (2003)	VND	Slope Scaling	Arb-Hom-Raw
Gronkvist (2006)	Constraint Programming	Column Generation	Pip-Het-Raw
Keedwell and Khu (2006)	GA	Cellular Automaton	Pip-Hom-Raw
Langston (1982)	Constructive Heuristic	Improvement Heuristic	Pip-Hom-Raw
Lapierre et al. (2004)	TS	VNS	Cyc-Het-Raw
Jain and Meeran (2002)	Core	Shell	Arb-Hom-Raw
Pécora (2002)	GA	TS	Cyc-Hom-Raw
Perez et al. (2005)	GRASP	PR	Pip-Hom-Raw
Potter and De Jong (1994)	Evolutionary Algorithm	Evolutionary Algorithm	Cyc-Het-Raw
Prins et al. (2007)	GTS	Lagrangean Relaxation	Cyc-Het-Pre
Repoussis et al. (2006)	GRASP	TS + VNS	Pip-Hom-Raw
Tarantilis et al. (2008)	TS + VNS	Guided Local Search	Pip-Hom-Raw
Vaidyanathan et al. (2008)	MIP Method	Very Large Scaled Neighborhood	Pip-Hom-Raw
Velarde and Laguna (2004)	TS	Linear Method	Cyc-Het-Pre

*Information Protocol: Pip = Pipeline, Cyc = Cyclic, Arb = Arbitrary;
Hom = Homogeneous, Het = Heterogeneous; Raw = Raw, Pre = Preprocessed

Table 1.1: Information exchange protocol in hybrid methods

1.2 Space restriction approaches

The idea of restricting the search space is not a new idea in OR. As the size of problems that can be solved to optimality is mostly determined by the power of the computational resources available, researchers have been proposing several procedures to remodel large scaled problems into a solvable sized problem, among others figures, mathematical manipulations, cutting planes algorithms, column generation and constraint propagation techniques. Geoffrion (1970a) mention several manipulations to deal with large scaled mathematical programming such as, decomposition, linearization, relaxation and projection. These manipulations can be seen as one of the pillars that hold the development of solution approaches up to today. Cutting plane algorithms aim at reducing the search space by proposing valid cuts with the goal of identifying the integer convex envelope. Column generation approaches follow the contrary path, they start with a very small space and look for profitable new variables to add to the restricted problem. There are also modeling issues used to restrict the solution space, such as the aggregation of variables. This approach has the drawback of not dealing with the exact problem, but rather an approximation of it, but this may be necessary and useful, especially when dealing with real-life problems. Constraint propagation techniques also aim at reducing the solutions space by identifying possible inconsistencies that cannot belong to any feasible solution. Also, neighborhood based heuristic methods explores the solution space by means of a well defined sequence of restricted spaces, called neighborhoods.

The rest of this subsection is dedicated to outline and exemplify these space restriction approaches to better place the IRSS into the current literature. It is not the goal of this section to provide a classification or a taxonomy for these approaches, for such matters specific reviews are provided. This section is divided into three parts: mathematical programming, constraint propagation and neighborhood based heuristic methods.

Mathematical programming

Dealing with large NP-hard problems using pseudo-enumerative methods can be very costly. In order to cope with large scale mathematical programs, approximations or manipulations (Geoffrion, 1970a,b) are used to transform the main problem into smaller and more manageable/solvable subproblems. Actually, these manipulations have the goal to reduce the solution space to a reasonable size, one which can be handled with known solution methods. For example, Benders Decomposition (Benders, 1962), fixes the values of the "complicating variables" reducing the main problem to a parameterized linear problem, (n.b. for the generalized form of Benders' Decomposition see Geoffrion, 1972), which is used to iteratively approximate the convex envelope of the solution space by identifying cutting planes. Costa (2005) presents a survey on Benders decomposition applied to fixed charge network design problems, Rei et al. (2006) used local branching to accelerate the classical Benders decomposition algorithm, improving simultaneously the lower and upper bounds for a series of network design problems. The Dantzig-Wolfe decomposition (Dantzig and Wolfe, 1960) method for linear problems, considered the basis of the column generation approaches, is based on the interaction of two models. A reduced instance, with fewer variables (columns) than the main model is solved and a sub model is used to identify the variable that should be added to improve the objective function. Gilmore and Gomory (1961) applied the Dantzig-Wolfe decomposition to the unidimensional cutting problem. Their algorithm iteratively solves the linear relaxation main model and uses the shadow prices in sub model that generates profitable cut patterns, added as new columns in the main model. The algorithm cycles between the main and the sub model until no cut pattern can be added to improve the objective function.

An integer programming technique used to reduce the solution space is cutting plane algorithms. The rationale behind these algorithms is to try to identify the facets – especially the ones in which the optimal solution belongs – of the convex envelop of the integer solutions. Because if this convex envelop is completely identified, any linear method will return the optimal integer solution. Usually such approaches are not able to identify the integer convex envelop – which is as difficult as solving the main problem to optimality – but instead, the cutting plane algorithms provide a good approximation for the convex envelope improving the convergence of the solution method. There are a number of cutting plane techniques which can be applied to generic MIP problems and for more information about this theory we refer the reader to the books (Wolsey, 1998 ; Nemhauser and Wolsey, 1988). Problem-specific cuts also can be added to strengthen the model e.g. Cavallet et al. (2000) uses a knapsack model to generate valid cuts that are based on the passing flow through the hubs for a location-allocation problem.

A restriction in the solution space can also be achieved by aggregating similar variables. This manipulation changes the structure of the problem to make it a simpler and easier one to solve. However, this simplification does not usually provide the optimal solution for the original problem but this technique is largely used when the original problem is very large and complex. Johnson and Wang (1998) tackled a problem of scheduling disassembly operations for recycling that is modeled as a two-commodity network problem. Disassembly operations can be very complex due to the sequence-dependent process leading to a very large mathematical program. Therefore, to reduce the final search space they use a technique that aggregates compatible materials and also similar assembly operations. In addition, the non-profitable disposal nodes are identified and eliminated from the search space. The final model, which is a relaxation of the real model, is then solved using a branch and bound algorithm. Prins et al. (2007) described a location routing problem (LRP). During one phase of their hybrid algorithm some customers have their demand aggregated into a super-customer, as called by the authors. Using this aggregation the LRP is transformed into a location allocation problem, which is solved by a Lagrangean relaxation.

It can be argued that the IRSS has its foundations on the mathematical programming approaches, for example the definition of a RS is related with the concept of projection, the structural variables, Chapter 5, is related with relaxation and decomposition. The IRSS uses these manipulations and the integer programming methods, such as Branch and Bound and Local Branching, to propose a framework for hybrid methods that is based on the successive definition and exploration of restricted spaces.

Constraint propagation

Constraint propagation techniques (Mackworth, 1977) are used in artificial intelligence to strengthen constraints and reduce the domain of variables leading to a reduction of the search space. Constraint propagation is also a method for reducing the search space of a problem instance based on the interdependence between the variables induced by the set of constraints. Its goal is to detect inconsistent assignments that cannot be part of any feasible solution. Dorndorf et al. (2000a) proposed a space reduction procedure based on constraint propagation and a time-oriented branching sequence, applied to the Resource-Constrained Project Scheduling with Generalized Precedence Constraints. Hybrid approaches using constraint propagation, together with branching techniques, may prove to be very effective when dealing with highly constrained mathematical programs. Artigues and Feillet (2008) and Sadykov (2008) used the constraint propagation techniques embedded in a branch and bound. In these works, the main goal of constraint propagation is to reduce the search space and generate infeasibility cuts to speed up the branching. The former minimizes the makespan, while the latter minimizes the weighted number of late jobs. Dorndorf et al. (2000b) provides a survey on constraint propagation applied to the disjunctive scheduling problem. The reader is referred of Rossi et al. (2006) to details and a list of references in constraint propagation.

The IRSS restricts the space by probing the solutions space and looking for common characteristics of good solutions. Constraint propagation techniques proposes to reduce the search space by identifying inconsistencies.

Neighborhood based heuristics

There several modern metaheuristics that use the concept of neighborhood, eg.: Simulated Annealing (Kirkpatrick et al., 1983), Multi Neighborhood (Boctor, 1993), Tabu Search (Glover and Laguna, 1997), Variable Neighborhood Search (Hansen and Mladenović, 2001) and Very Large Scale Neighborhood Search (Ahuja et al., 2002).

A neighborhood can also be seen as restriction of space, differently of the valid cut, column generation and mathematical manipulations techniques, neighborhood based heuristics does not transform the solution space. Instead, it selects a region around a reference solution to perform the search procedure, due to this characteristic, most of the neighborhood based heuristics act locally thus they need some procedure to diversify the search, like tabu list in TS, the control parameter in SA, multi neighborhoods or even all these artifices together in a hybrid method. For more on neighborhood heuristics we suggest the work Glover and Kochenberger (2003) to the reader.

Differently from neighborhood based metaheuristics, the Iterative Restricted Space Search proposes an algorithm structure to explore the solution space which naturally provides the diversification. As will be described in the next chapters two restricted spaces have not points in common, which naturally leads the search to new regions of the solution space.

The idea of reducing the solution space to be explored is a natural way to deal with large scale problems and it has been explored in much different ways during the last years. In this thesis, this concept is used in the framework both inside the "macro" search as exemplified in Chapter 5 and also in the information passed from the "macro" search to the "micro" search, the RS. In addition, the IRSS proposes an intelligent way to define the restricted space based on the information gathered during the "macro" search.

References

Ahuja, R. K., Ergun, O., Orlin, J. B., and Punnen, A. P. (2002). A survey of very large-scale neighborhood search techniques. *Discrete Applied Mathematics*, 123(1-3):75–102.

Ahuja, R. K., Liu, J., Orlin, J. B., Sharma, D., and Shughart, L. A. (2005). Solving real-life locomotive-scheduling problems. *Transportation Science*, 39(4):503–517.

Alvim, A. C. F., Ribeiro, C. C., and Glover, F. (2004). A hybrid improvement heuristic for the one-dimensional bin packing problem. *Journal of Heuristics*, 10(2):205–229.

Artigues, C. and Feillet, D. (2008). A branch and bound method for the job-shop problem with sequence-dependent setup times. *Annals of Operations Research*, 159(1):135–159.

Atkin, J. A. D., Burke, E. K., Greenwood, J. S., and Reeson, D. (2007). Hybrid metaheuristics to aid runway scheduling at london heathrow airport. *Transportation Science*, 41(1):90–106.

Barnhart, C., Hane, C. A., and Vance, P. H. (2000). Using branch-and-price-and-cut to solve origin-destination integer multicommodity flow problems. *Operations Research*, 48(2):318–326.

Benders, J. (1962). Partitioning procedures for solving mixed-variables programming problems. *Numerische Mathematik*, 4:238–252.

Bertels, S. and Fahle, T. (2006). A hybrid setup for a hybrid scenario: combining heuristics for the home health care problem. *Computers & Operations Research*, 33(10):2866–2890.

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

Boctor, F. F. (1993). Discrete optimization and multi-neighbourhood, local improvement heuristics. Technical report, Groupe de Recherche en gestion de la logistic : Université Laval.

Borraz-Sanchez, C. and Rios-Mercado, R. Z. (2005). A hybrid meta-heuristic approach for natural gas pipeline network optimization. In *Hybrid Metaheuristics, Proceedings*, volume 3636 of *Lecture Notes in Computer Science*, pages 54–65. Springer-Verlag Berlin, Germany.

Budenbender, K., Grunert, T., and Sebastian, H. J. (2000). A hybrid tabu search/branch-and-bound algorithm for the direct flight network design problem. *Transportation Science*, 34(4):364–380.

Carter, M. W. and Lapierre, S. D. (1999). Scheduling emergency room physicians. Technical Report 99-23, Centre for Research on Transportation, Montreal, Canada.

Cavallet, A., Malucelli, F., and Wolfler Calvo, R. (2000). A non linear multicommodity network design approach to solve a location-allocation problem in freight transportation. In *ODYSSEUS 2000 Workshop in freight transportation and logistics.* Chania (Greece).

Correa, A. I., Langevin, A., and Rousseau, L. M. (2004). Dispatching and conflict-free routing of automated guided vehicles: A hybrid approach combining constraint programming and mixed integer programming. In *Integration of Ai and or Techniques in Constraint Programming for Combinatorial Optimization Problems*, volume 3011 of *Lecture Notes in Computer Science*, pages 370–379. Springer-Verlag Berlin, Berlin.

Costa, A. M. (2005). A survey on benders decomposition applied to fixed-charge network design problems. *Computers & Operations Research*, 32:1429–1450.

Crainic, T. G. and Gendreau, M. (2002). Cooperative parallel tabu search for capacitated network design. *Journal of Heuristics*, 8(6):601–627.

da Silva, C. G., Figueira, J., and Climaco, J. (2007). Integrating partial optimization with scatter search for solving bi-criteria 0,1-knapsack problems. *European Journal of Operational Research*, 177(3):1656–1677.

Danna, E., Rothberg, E., and Le Pape, C. (2005). Exploring relaxation induced neighborhoods to improve mip solutions. *Mathematical Programming*, 102(1):71–90.

Dantzig, G. B. and Wolfe, P. (1960). Decomposition principle for linear programs. *Operations Research*, 8(1):101–111.

Delmaire, H., Diaz, J. A., Fernandez, E., and Ortega, M. (1999). Reactive grasp and tabu search based heuristics for the single source capacitated plant location problem. *Infor*, 37(3):194–225.

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

Dorndorf, U., Pesch, E., and Phan-Huy, T. (2000a). A time-oriented branch-and-bound algorithm for resource-constrained project scheduling with generalised precedence constraints. *Management Science*, 46(10):1365–1384.

Dorndorf, U., Pesch, E., and Toan, P. H. (2000b). Constraint propagation techniques for the disjunctive scheduling problem. *Artificial Intelligence*, 122(1-2):189–240.

French, A. P., Robinson, A. C., and Wilson, J. M. (2001). Using a hybrid genetic-algorithm/branch and bound approach to solve feasibility and optimization integer programming problems. *Journal of Heuristics*, 7(6):551–564.

Gallardo, J., Cotta, C., and Fernandez, A. (2007). On the hybridization of memetic algorithms with branch-and-bound techniques. *IEEE Transactions On Systems Man and Cybernetics Part B-Cybernetics*, 37-1:77–83.

Gendron, B., Potvin, J.-Y., and Soriano, P. (2003). A parallel hybrid heuristic for the multicommodity capacitated location problem with balancing requirements. *Parallel Computing*, 19:591–606.

Geoffrion, A. M. (1970a). Elements of large-scale mathematical programming part i. *Management Science*, 18(11):652–675.

Geoffrion, A. M. (1970b). Elements of large-scale mathematical programming part ii. *Management Science*, 16(11):676–691.

Geoffrion, A. M. (1972). Generalized benders decomposition. *Journal of Optimization Theory and Applications*, 10(4):237–260.

Gilmore, P. and Gomory, R. (1961). A linear programming approach to the cutting-stock problem. *Operations Research*, 9(6):849–859.

Glover, F. (1977). Heuristics for integer programming using surrogate constraints. *Decision Sciences*, 8:156–166.

Glover, F. and Laguna, M. (1997). *Tabu search.* Kluwer academic publishers.

Glover, F. W. and Kochenberger, G. A., editors (2003). *Handbook of Metaheuristics.* Springer.

Gronkvist, M. (2006). Accelerating column generation for aircraft scheduling using constraint propagation. *Computers & Operations Research*, 33(10):2918–2934.

Hansen, P. and Mladenović, N. (2001). Variable neighbourhood search: Principles and applications. *Euroupean Journal of Operational Research*, 130:449–467.

Holland, J. (1975). *Adaptation in Natural and Artificial Systems.* MIT Press.

Hooker, J. N. (2006). An integrated method for planning and scheduling to minimize tardiness. *Constraints*, 11(2-3):139–157.

Jain, A. S. and Meeran, S. (2002). A multi-level hybrid framework applied to the general flow-shop scheduling problem. *Computers & Operations Research*, 29(13):1873–1901.

Johnson, M. R. and Wang, M. H. (1998). Economical evaluation of disassembly operations for recycling, remanufacturing and reuse. *International Journal of Production Research*, 36(12):3227–3252.

Keedwell, E. and Khu, S. T. (2006). A novel evolutionary meta-heuristic for the multi-objective optimization of real-world water distribution networks. *Engineering Optimization*, 38(3):319–336.

Kim, D. and Pardalos, P. M. (2000). Dynamic slope scaling and trust interval techniques for solving concave piecewise linear network flow problems. *Networks*, 35(3):216–222.

Kirkpatrick, S., Gelatt, C. D., and Vecchi, M. P. (1983). Optimization by simulated annealing. *Science*, 220(4598):671–680.

Land, A. H. and Doig, A. G. (1960). An automatic method of solving discrete programming-problems. *Econometrica*, 28(3):497–520.

Langston, M. A. (1982). Improved 0/1-interchange scheduling. *Bit*, 22(3):282–290.

Lapierre, S. D., Ruiz, A. B., and Soriano, P. (2004). Designing distribution networks: Formulations and solution heuristic. *Transportation Science*, 38(2):174–187.

Lawler, E. (1976). *Combinatorial Optimization: Networks and Matroids.* Holt, Rinehart and Winston, USA.

Mackworth, A. K. (1977). Consistency in networks of relations. *Artificial Intelligence*, 8(1):99–118.

Manber, U. (1989). *Introduction to Algorithms - A Creative Approach.* Addison-Wesley Publishing Company.

Nemhauser, G. L. and Wolsey, L. A. (1988). *Integer and Combinatorial Optimization.* Wiley-Interscience.

Nguyen, H. D., Yoshihara, I., Yamamori, K., and Yasunaga, M. (2007). Implementation of an effective hybrid ga for large-scale traveling salesman problems. *IEEE Transactions on Systems, Man and Cybernetics, Part B*, 37(1):92–99.

Papadimitriou, C. and Steiglitz, K. (1982). *Combinatorial Optimization: Algorithms and Complexity.* Prentice-Hall.

Pécora, J. E. (2002). Alocação de médicos em salas de emergências - uma abordagem híbrida. Master's thesis, Campinas University, Campinas, Brasil.

Pendharkar, P. C. (2005). Hybrid approaches for classification under information acquisition cost constraint. *Decision Support Systems*, 41(1):228–241.

Peng, J., Shang, G., and Liu, H. (2006). A hybrid intelligent algorithm for vehicle routing models with fuzzy travel times. *Computational Intelligence, Pt 2, Proceedings*, 4114:965–976.

Perez, M., Almeida, F., and Moreno-Vega, J. M. (2005). A hybrid grasp-path relinking algorithm for the capacitated p-hub median problem. In *Hybrid Metaheuristics, Proceedings*, volume 3636 of *Lecture Notes in Computer Science*, pages 142–153. Springer-Verlag Berlin, Germany.

Potter, M. A. and De Jong, K. A. (1994). A cooperative coevolutionary approach to function optimization. In *The 3rd Parallel Problem Solving from Nature (PPSN), Proceedings of the Conference.*, pages 249–257. Springer-Verlag.

Prins, C., Prodhon, C., Ruiz, A., Soriano, P., and Wolfler Calvo, R. (2007). Solving the capacitated location-routing problem by a cooperative lagrangean relaxation-granular tabu search heuristic. *Transportation Science*, 41(4):470–483.

Puchinger, J. and Raidl, G. R. (2005). Combining metaheuristics and exact algorithms in combinatorial optimization: A survey and classification. In *Artificial Intelligence and Knowledge Engineering Applications: a Bioinspired Approach, Pt 2, Proceedings*, volume 3562 of *Lecture Notes in Computer Science*, pages 41–53. Springer-Verlag, Berlin.

Rei, W., Cordeau, J., Gendreau, M., , and Soriano, P. (2006). Accelerating benders decomposition by local branchings. Technical report, Center for Research on Transportation - C.P. 6128, succursale Centre-ville Montr´eal QC H3C 3J7 Canada.

Repoussis, P. P., Paraskevopoulos, D. C., Tarantilis, C. D., and Ioannou, G. (2006). A reactive greedy randomized variable neighborhood tabu search for the. vehicle routing problem with time windows. In *Hybrid Metaheuristics, Proceedings*, volume 4030 of *Lecture Notes in Computer Science*, pages 124–138. Springer-Verlag Berlin, Germany.

Resende, M. G. C. and Werneck, R. F. (2004). A hybrid heuristic for the p-median problem. *Journal of Heuristics*, 10(1):59–88.

Rossi, F., van Beek, P., and Walsh, T., editors (2006). *Handbook of Constraint Programming*. Elsevier.

Sadykov, R. (2008). A branch-and-check algorithm for minimizing the weighted number of late jobs on a single machine with release dates. *European Journal of Operational Research*, 189(3):1284–1304.

Sahoo, B. and Maity, D. (2007). amage assessment of structures using hybrid neuro-genetic algorithm. *Applied Soft Computing*, 7-1:89–104.

Schlottmann, F. and Seese, D. (2004). A hybrid heuristic approach to discrete multi-objective optimization of credit portfolios. *Computational Statistics & Data Analysis*, 47(2):373–399.

Talbi, E. G. (2002). A taxonomy of hybrid metaheuristics. *Journal of Heuristics*, 8(5):541–564.

Tarantilis, C. D., Zachariadis, E. E., and Kiranoudis, C. T. (2008). A hybrid guided local search for the vehicle-routing problem with intermediate replenishment facilities. *Informs Journal on Computing*, 20(1):154–168.

Toth, P. and Vigo, D. (2003). The granular tabu search and its application to the vehicle-routing problem. *Informs Journal on Computing*, 15(4):333–346.

Vaidyanathan, B., Ahuja, R. K., Liu, J., and Shughart, L. A. (2008). Real-life locomotive planning: New formulations and computational results. *Transportation Research Part B-Methodological*, 42(2):147–168.

Velarde, J. L. G. and Laguna, M. (2004). A benders-based heuristic for the robust capacitated international sourcing problem. *IIE Transactions*, 36(11):1125–1133.

Voβ, S., Martello, S., Osman, I., and Roucariol, C., editors (1999). *Metaheuristics - Advances and Trends in Local Search Paradigms for Optimization.* Kluwer Academic Publishers.

von Neumann, J. and Burks, A. W., editors (1966). *Theory of Self-Reproducing Automata.* University of Illinois Press.

Wiegand, P. (2003). *An Analysis of Cooperative Coevolutionary Algorithms.* PhD thesis, George Mason University.

Wolsey, L. A. (1998). *Integer Programming.* Wiley-Interscience.

Yan, S. and Zhou, K. (2006). Three-tier multi-agent approach for solving traveling salesman problem. *PRICAI 2006: Trends In Artificial Intelligence, Proceedings*, 4099:813–817.

Part II

Application to a Pulp and Paper Scheduling Problem

"The best friend on Earth of man is the tree. When we use the tree respectfully and economically, we have one of the greatest resources of the Earth."

Frank Lloyd Wright (1867 – 1959)

Introduction

The contribution of this part is twofold. First, it models and introduces a new problem, the Pulp Production Scheduling Problem (PPSP), and second it proposes a new solution approach the Restricted Space Search. To this end it is divided into two chapters, the first one is dedicated to the definition and formalization of the mathematical model related to the Pulp production Scheduling Problem, a real-life problem that arrises in the context of a Brazilian Pulp company. The main objective of this problem is homogenize the feed of wood chips in the digester – a pressure vessel where the pulp is produced – in order to optimize the quality of the pulp produced. At this point, as the main goal of this article is not the solution approach, but the formalization of the problem, a simple constructive heuristic is proposed. This article was published in INFOR Volume 45, Number 4 in November 2007.

The second chapter of this part proposes a new solution approach, the Restricted Space Search, for the same problem. It lays the basis and develops the first concepts of this thesis, as the two phase search, structural variables, restricted spaces and direction vectors. These concepts are extended in Part III and generalized in Part IV.

Despite dealing with the same problem, the reader will notice that the models in these two chapters are not exactly the same, this is due to the development of the ideas and a better understanding of the problem. The first difference is that in Chapter 2 the demands are modeled in volume of wood, due to the fact that the Brazilian pulp has a very good strategic planning all the harvest areas have almost the same size. Implying that the total volume of wood logs produced by each harvest area is very similar. Therefore, in the second chapter we modeled the demand by numbers of harvest areas consumed per period instead of total volume of wood. Leading to a new model that avoids the partition of the same harvest area into different digesters, without the need of binary variables or new constraints, an undesirable operational constraint for the pulp plant. The second difference is that the constraints imposing a smooth changing in the density of the wood chips fed in the digester, (constraints 2.6) does not belong to the second mathematical model. There are two main reasons for that, first as the range of possible basic densities are small, thus even a great change in the feeding will not affect the final pulp quality substantially. Second is that the wood chips waiting to be fed in the digester are stored in heaps which mixes them. Operationally the better solution for this problem is the implementation of a new storage facilities in which the different chips could eventually be separated, a solution that the pulp company considered reasonable but inviable for the moment. Therefore to better adequate our model to the reality of this pulp plant these constraints where subtracted of the model.

Despite being two chapters they are complementary and both contribute for this research. The first by modeling a new real-life problem and the second by proposing a new solution approach for this problem.

Chapter 2

Introduction to Pulp Production Scheduling Problem

Résumé - La pâte kraft, matière première nécessaire à la fabrication du papier, s'obtient par cuisson de copeaux de bois dans des lessiveurs. Les paramètres physico-chimiques du processus (température, pression, temps de cuisson et quantités de produits chimiques), établis selon la densité basique du bois à transformer, ont un impact majeur sur la qualité de la pâte. En sélectionnant des parcelles dont les densités basiques sont similaires, on optimise à la fois le processus de production et la qualité du produit obtenu. La quantité de bois disponible dans chaque parcelle étant limitée et les densités pouvant être significativement différentes d'une parcelle à l'autre, la planification opérationnelle de la production consiste essentiellement à combiner, pour chaque lessiveur, les parcelles à exploiter, de telle manière que les densités soient les plus homogènes possible. Ce problème est modélisé comme un problème d'ordonnancement dans lequel chaque parcelle est affectée à un lessiveur et à une période, en minimisant la variance des densités des copeaux de bois à transformer pour chaque période de l'horizon de planification. Nous proposons une solution heuristique basée sur cette formulation et discutons de son efficacité à l'aide des résultats numériques obtenus.

MOTS CLÉS : ordonnancement, foresterie, modélisation mathématique, heuristique constructive

Minimization of the wood density variation in pulp and paper production

José Eduardo Pécora Jr [†▽], Angel B. Ruiz[†▽], Patrick Soriano[‡▽]

[†]Dép. opérations et systémes de décision, Faculté des sciences de l'administration, Université Laval
Québec (Qc) Canada G1K 7P4
eduardo.pecora@centor.ulaval.ca, angel.ruiz@ulaval.ca

[‡]Service des Méthodes quantitatives de gestion, HEC-Montréal
3000, ch. Côte-Sainte-Catherine , Montréal (Qc) H3T 2A7, Canada
patrick@crt.umontreal.ca

[▽]CIRRELT - Centre interuniversitaire de recherche sur les reseaux d'entreprise, la logistique et le transport, Université de Montréal
CP 6128 Succursale Centre-ville, Montréal (Qc) H3C 3J7 Canada

Abstract

Kraft pulp, the raw material in paper manufacturing, is obtained by exposing pieces of wood to a cooking process. The parameters of this process (temperatures, times, chemicals, etc.) depend strongly on the density of the different woods in the cooker and have an influence on the quality of the pulp that is obtained. In order to optimize both the production process and the pulp quality, one wishes to cook together woods having similar densities. However, given that the harvest areas contain limited quantities of trees and that wood densities vary significantly from one area to another, deciding how harvested wood from different areas should be mixed for processing is a significant operational decision in pulp production planning. This situation is modeled as a scheduling problem where one has to decide which harvested area goes to each available processing units in order to minimize the variance of wood densities within each cooker for each period of the planning horizon. We also present an approximated solution approach based on the formulation proposed. Some results are reported and the efficiency of the method is discussed.

KEY WORDS: scheduling, forestry, mathematical modeling, constructive heuristic

2.1 Introduction

Despite the promises of several end-of-century gurus, we are still far away from a paperless world. Pulp, the base for paper and cardboard production, is still a healthy industry. The world consumption of paper and cardboard attained 339 million tons in 2003, and grows by about 3% per year (see http://www.copacel.fr/anglais.htm). The large industrialized countries make up the lion's share of this consumption. Production per capita in these countries is estimated to be between 179 kg per year (Spain) and 301 kg per year (United States). The consumption of paper and cardboard is linked to economic activity, but also to wood availability at the different stock points present at the different levels of the value chain.

The process for the manufacture of pulp and paper basically consists in transforming wood into a fibrous material, known as paste, pulp or industrial pulp. However, industrial models vary from one country (or region) to another, according to climatic conditions, harvested tree species as well as the specific local supplying and transformation practices. In this paper, we will focus on the particular model observed in the Brazilian industry. This model differs from others – the Canadian model, for example – in several aspects concerning mainly the forest management and characteristics (species), supply issues (logistics and transportation), the transformation process and the distribution and fulfillment strategies. The particularities of the Brazilian pulp and paper production model will be discussed in the next paragraphs.

In contrast to the Nordic and Canadian production models, Brazilian pulp and paper production is almost exclusively based on the transformation of eucalyptus rather than a mix of different species. Eucalyptus trees have a much shorter live cycle than Nordic species (five to seven years to maturity instead of 35 to 40) and they are cultivated in plantations where genetically selected species are exclusively dedicated to pulp and paper production. Farms are divided into harvest areas, which are homogeneous parcels of land producing enough wood to feed a pulp digester for one day. This sylviculture approach leads to other major operational differences with respect to the exploitation of natural forests. First of all, as the area of these farms is significantly smaller than the ones dedicated to Nordic species, traveling distances to the processing factories are shorter. In fact, a Brazilian tree will travel around 40 km to reach the transformation facility while a Canadian tree will easily travel 400 km or even more. Secondly, the access to parcels is on permanent routes. Thus, in the studied case, we will exclusively concentrate on transportation scheduling, neglecting other operational issues such transportation modes, transportation delays or even route construction.

Once an area has been harvested, cut wood is left to dry naturally for a period of 4 to 6 weeks and then transported to the mills. This waiting time allows the wood to loose part of its humidity, reducing its weight and therefore the transportation cost. However, if the wood becomes too dry further operations in the transformation cycle might be hindered. Therefore, taking into account the moist Brazilian climate, plant engineers recommend wood transportation within a time window ranging from the fourth to the 12th week after harvest. Note that this is in contrast with Nordic countries, where wood humidity loss is an important concern due to their dryer climates. Transportation is mostly done by truck, one harvest area requiring several trucks.

Pulp is extracted from wood chips by mechanical, chemical or thermal methods, but the most common one in the Brazilian pulp industry is a thermo-chemical process which is known as the *kraft process* (see Kennedy et al., 1989 ; Biermann, 1996). This cooking consists in submitting the chips to the chemical action of strong *white liquor* (composed of caustic soda – i.e. sodium hydroxide – and sodium sulphite) and steam inside a digester in order to separate the lignin that binds the fibre in the wood. The liberated fibre is industrial pulp. The digester is a pressure vessel where the chips and white liquor are continuously introduced through an opening at the top. Total cooking time of the wood chips is around 120 minutes, carried out from the top to the center of the digester. From the center down, a washing operation is conducted to remove the residual solution, weak black liquor (the strong white liquor used in cooking plus lignin removed from the wood), which is later used as fuel for the recovery boilers.

The industrial setting that we observed includes a single factory with three different cookers, each with its own wood pile in the factory backyard. Wood logs from the piles are sent through chippers to be slashed into wood chips. These are then fed to the digester area, where the cooking process begins. Figure 2.1 schematizes such a pulp production process and the particular industrial setting considered (within the dashed box).

Pulp produced from wood chips with different densities are used to manufacture distinct types of paper. Pulp coming from high density chips are more suitable for

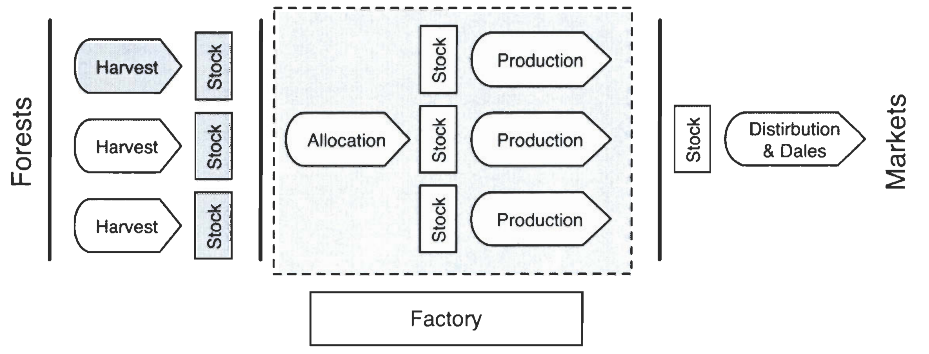

Figure 2.1: Pulp production process and the decisions considered

packing grades where high tear strength is important. On the other hand, pulp produced from low density chips are preferred to manufacture fine papers (Kennedy et al. 1989).

Cooking wood chips with different basic densities produces necessarily a mix of both overcooked and undercooked pulp (Biermann, 1996 ; Kibblewhite and Uprichard, 1996 ; Williams, 1994). Homogeneous wood chips are desired when submitting them to a kraft process, because one will have a greater control of the final pulp quality and will avoid the losses in production due to under or over-cooking.

In order to optimize production, one needs to carefully set the digester parameters according to the characteristics of the wood to be processed and, in particular, to the wood *basic density*. To this end, production planners have to deal with two difficulties.

- Firstly, chips coming from harvest areas with different basic densities clearly require different cooking times, different temperatures, but especially different doses of chemicals which motivate engineers to "group" similar woods to be cooked together.

- Secondly, pulp production is a continuous process which means that the process cannot be stopped while changing the cooking parameters. Thereby, a second goal when planning production seeks to avoid (or minimize as much as possible) the number and the size of the changes in cooking parameters. Since wood fed into the digester comes from a large and heterogeneous set of chips, production planners need to choose – schedule – from which of the available forests one should transport wood in order to feed the cooking processors with similar density woods at the different factories and minimize the changes in their cooking settings.

Unfortunately, there is no known economical function relating the homogeneity of the wood being cooked together and the processing costs and performance. However, plant engineers from our industrial partner estimate that savings of up to 10 US$ per ton of produced pulp may be achieved if (1) the digester is fed with a good mix of wood – i.e. wood with similar basic densities – and (2) changes in the cooking parameters are minimized. The impact of such savings is extremely important for an industry that, though still growing, shows strong price-driven behaviour.

Taking into account the above considerations this paper proposes a mathematical model that aims at helping managers to schedule wood transportation to feed the plant digesters in order to minimize the wood density variance within each of them.

Introducing density variance within the objective function naturally leads to a non-linear mathematical model. We therefore apply a linearization strategy, based on the discretization of the continuous density values into discrete density levels. The mathematical model accounts for classical constraints such as wood availability and system capacities, but also considers engineering plant constraints such as specific restrictions that aim at smooth digester operation by avoiding radical changes in the digester settings between consecutive production periods.

The paper is organized as follows; the next section presents a review of the related literature. Section 3 introduces some notation before presenting the mathematical formulation. A real-life industrial application is presented in Section 4. Finally, section 5 concludes the paper.

2.2 Research areas within pulp production literature

As the scientific production in forestry is vast, our literature review concentrates exclusively in two research streams. The selected issues are: 1) the application of operational research techniques to general forestry problems and wood supply chain management and 2) the chemical process involved in pulp production and, in particular, the key role of wood density in the transformation processes. In the next paragraphs we review the scope of these respective research domains and we explain how the present study fits within the research framework of the forestry discipline.

Several research works have concentrated in applying OR techniques to the forestry context. Weintraub et al. (2000) discuss the combinatorial nature of some forest management problems and the OR algorithms that have been proposed to tackle them. This survey also introduces some imposed environmental constraints, which lead to the adjacency problem (see also Murray and Church, 1995, 1996a,b). In short, these environmental restrictions state that, in order to protect wildlife, adjacent areas should not be harvested in the same time period thus preventing the destroyed habitat zones from being too large and preserving sufficient movement corridors between unharvested forest units for wildlife.

The concept of supply chain management (SCM) arose in the early 90s. SCM has allowed many companies to achieve important improvements in almost every kind of logistic system (see Slats et al., 1995 ; Geoffrion and Powers, 1995). It attempts to coordinate the logistic activities around the different echelons of the production chain, but it can also improve the supply coordination between vertically related companies or even competing companies. SCM is particularly appealing for the forestry industry due to the large distances that generally separate forests and production centers from customers. Increasingly, these are more spread out all around the world with the growing globalization of the economy. Besides, wood is a heavy and dense material which implies costly transportation, handling and storage activities.

Ronnqvist (2003), studies the wood supply chain and production planning. Forest management, as well as transportation and routing issues, are considered within strategic, tactical and operational planning. At the strategic level – a horizon ranging from 20 to 25 years – the transportation problem relates to road building and upgrading in order to guarantee future timber flow. In tactical planning, usually with a horizon of 2 to 4 years, decision makers are interested in selecting the set of areas which will be harvested in the next period, to meet expected market demand. The operational level – 1 or 2 months – is mostly concerned by the choice of transportation modes and the target becomes minimization of the costs. Karlsson et al. (2003) discuss transportation issues and other operational questions concerning the Swedish forest industry such as crew scheduling or road construction and clearing (snow removal).

The second research stream concerns the pulp production process itself. White liquor, the base for pulp and paper production, may be produced by several different mechanical, chemical and thermal processes. In particular, this study is concerned by the thermo-chemical process production. Roughly, this process submits the pieces of wood to precise temperature and pressure conditions in the presence of chemical agents. The optimal receipt for such a cooking process requires a correct balance between these parameters and the physical characteristics of the wood (Biermann, 1996 ; Kennedy et al., 1989).

A good comprehension of the relationships between raw material and the final product is mandatory for any pulp and paper industry which would like to face the current competitive and demanding market. The impact of wood characteristics on the paper process is studied by Williams (1994). In his work the author states that a classification of wood into density ranges is beneficial for the final quality of the pulp. Furthermore, Foelkel et al. (1992), demonstrate that density is a critical physical property of wood in the eucalyptus cooking equation. Basic density appears as a key factor during the delignification – wood impregnation with the cooking liquor – procedure. In general, wood chips with high density present greater resistance to liquor flow during cooking, leading to a deficient degree of impregnation. Thus, the intensity of the delignification reactions is reduced and the required cooking times, as well as the chemical doses, need to be readjusted. Therefore, programming the thermo-chemical digester parameters under a mix of heterogeneous wood fibres (different densities) represents a difficult but worthy problem for the plant engineers.

Several other studies have concentrated on how to measure the values of the wood physical properties in practice better and faster. Schimleck et al. (2005) use infrared spectroscopy to determine basic density and pulp yield properties. Molteberg (2004) describes measurement methods to determine several wood properties like basic density, dry density, fibre length, kraft pulp yield and others, using small wood samples. Duffy and Kibblewhite (1989) present a method based on the suspension of fibres to infer the wood parameters and their importance in the quality of the final product.

However, despite the proved impact of basic density management in the pulp production process, a thorough bibliographic search did not find any previous research that dealt explicitly with the variance in basic wood density within pulp production planning or wood procurement and transportation schedules. To the best of our knowledge, this study presents the first mathematical model integrating wood transportation and production decisions within a wood density variance optimization objective. The next section first presents the different parameters and decisions variables required by proposed model followed by a formal mathematical formulation.

2.3 Problem description and mathematical model

The notation needed to describe the mathematical model is as follows. Let W be the set of wood to be transported. Note that not all areas are available to transport at the same time: some areas are too young, some are too old, and others do not meet the minimum required quality to produce pulp. Each individual area $w \in W$, is characterized by the volume of wood available, v_w, and the average basic density, δ_w. Also, each harvested area has its own transportation window. In practice lower and upper bounds of 4 and 12 weeks respectively, are imposed for the wood transportation window. We model these transportation time windows by binary parameters τ_{wt}, which take value 1 if the area w may be transported within period t and 0 otherwise. The average transportation time is small when compared to the planning period of one week, so we have no significant reason to consider transportation times between the forest and the mill.

On the manufacturing side, we assume that a single mill is equipped with several digesters identified by index $f \in F$. Wood arriving from the harvested areas is temporarily stockpiled in the mill stockyard, which has a limited total storage capacity $MStock$. The pulp demand for every digester and period, denoted by d_{ft}, is deterministic and known in advance. They are also assumed to be less than the corresponding digester production capacity.

As we mentioned in the previous section, to the best of our knowledge there is neither a theoretical nor empirical formula providing a relationship between the homogeneity of the wood and cooking costs or pulp quality. Nevertheless, reducing the basic wood density variability has been identified by plant engineers as their primary goal when planning wood transportation and their allocation to the cookers. Therefore, we have translated the engineering wishes into a mathematical objective function which aims at minimizing the wood basic density variation for each cooker and period.

Unfortunately, the variance measure introduces a non-linear term in the objective function. In order to overcome this drawback, we propose a linearization strategy based on the definition of a set of discrete density ranges and two sets of auxiliary variables. The linearization starts by separating the continuous range of density values into a finite set of working levels $l \in L$, each level characterized by an average density δ_l. Using this density classification, we define σ_{wl} as the average density deviation of each harvested area w with respect to the median density of each of the proposed working levels $l \in L$. The average density deviation is computed as $\sigma_{wl} = (\delta_w - \delta_l)^2$.

Based on the previous definition of the metric deviation, we propose a set of auxiliary variables $VDev_{flt}$ which give, for each period t, the distance between the average density of the wood-mix within each digester f, and each of the proposed working levels l. Lastly, we propose a second set of auxiliary variables, $VSlack_{flt}$, whose purpose will be explained later in this section.

Although discrete working levels are used to describe the digester settings, pulp production is a continuous process. Therefore, changing digester parameters at the end of a given period is somewhat more difficult than could be guessed a priori, so changes should be avoided or at the least minimized. In order to account for these operational constraints, the model includes equations that limit the change in working levels observed from one period to the next to changes from one level to either of its two adjacent levels (i.e. from l to either $l-1$ or $l+1$). The next paragraphs summarize the different sets of indexes, parameters and variables used in the model formulation that follows.

Thus, the Pulp Production Scheduling Problem (PPSP) can be written as follows:

	Index
w	Index for harvested wood waiting for transportation; $w = 1, ..., \|W\|$
f	Index for pulp processing units (digesters); $f = 1, \ldots, \|F\|$
l	Index for the digester working level; $l = 1, \ldots, \|L\|$
t	Index for period number; $t = 1, \ldots, \|T\|$
	Forest data and Transportation windows
v_w	Available wood volume in area w;
δ_w	Average wood density of area w;
σ_{wl}	Deviation of the average density of area w w/r to the median density of working level l;
$W\prime$	Set of harvested areas that must be consumed within the planning horizon due to the expiration of their transportation window;
τ_{wt}	Time window parameter which equals 1 if area w may be transported at period t, 0 otherwise;
	Factory and demand data
$MStock$	Stockyard capacity (in volume);
VS_{w0}	Volume of wood from area w Stored at the factory at the beginning of the planning horizon;
d_{ft}	Volume of wood to be transformed in digester f during period t (i.e. demand);
	Binary decision variables
Z_{flt}	Equals 1 if digester f is set to work at level l during period t, 0 otherwise;
	Continuous and non-negative variables
VT_{wt}	Volume of wood from harvested unit w Transported at period t;
VC_{wft}	Volume of wood from harvested unit w Consumed (processed) by digester f at period t;
VS_{wt}	Volume of wood from harvested unit w in Stock at the end of period t;
	Auxiliary, continuous and non-negative variables
$VDev_{flt}$	Deviation of the average wood density within digester f with respect to working level l at period t;
$VSlack_{flt}$	Deviation slack of digester f with respect to working level l at period t;

$$Min \sum_f \sum_l \sum_t VDev_{flt} + p_1 \sum_{w \in W\prime} (v_w - \sum_f \sum_t VC_{wft}) + p_2 \sum_w \sum_t VS_{wt} \tag{2.1}$$

Subject to:

$$\begin{aligned}
\sum_l Z_{flt} &= 1; && \forall f, t; && (2.2)\\
VDev_{flt} + VSlack_{flt} &= \textstyle\sum_w VC_{wft}\sigma_{wl}; && \forall f, l, t; && (2.3)\\
VSlack_{flt} &\leq M(1 - Z_{flt}); && \forall f, l, t && (2.4)\\
VDev_{flt} &\leq VSlack_{fl't}; && \forall f, t, l, l' \in \{L : l \neq l'\} && (2.5)\\
Z_{fl-1t+1} + Z_{flt+1} + Z_{fl+1t+1} &\geq Z_{flt}; && \forall f, l, t \in \{1, ..., |T| - 1\} && (2.6)\\
\sum_f VC_{wtf} &\leq v_w \tau_{wl}; && \forall w, t && (2.7)\\
\sum_w VC_{wft} &= d_{ft} && \forall f, t && (2.8)\\
\sum VS_{wt} &\leq MStock && \forall t && (2.9)\\
\sum_t VT_{wt} &\leq v_w && \forall w && (2.10)\\
VT_{wt} + VS_{wt-1} - \sum_f VC_{wft} &= VS_{wt} && \forall w, t && (2.11)
\end{aligned}$$

As previously mentioned, PPSP is a linear integer program. The first term of the objective function (2.1) computes the sum, over each period, of the deviation of the wood densities within each cooker with respect to the target working level. Clearly, minimizing this sum leads to maximizing the homogeneity of the wood density, the goal pursued by the plant engineers. The second term in (2.1) penalizes the wasted wood, i.e. the wood that could not be processed before the end of its allowed time window. The third term aims at reducing the amount of wood stocked at the factory by imposing a penalty on it. It is desirable that this penalized term in the objective function reaches zero. Therefore, the values for penalty weights p_1 and p_2 are positive and given the relative importance of each term, we impose $p_1 > p_2$. The numerical values for these parameters will be adjusted empirically after preliminary experimental tests.

The model contains three sets of constraints. The first one serves to linearize the model. First, we have to ensure that the variables $VDev_{flt}$ compute only the deviation of the harvested area average density with respect to the target working level of the digester for the given period. To this end, constraints (2.2) to (2.5) are required. Constraints (2.2) state that one and only one working level per factory and period will be chosen (i.e. the target level). Then, note that constraints (2.4) allow variables $VSlack_{flt}$ to take any positive value bounded by M. However, $VSlack_{flt}$ are forced to zero for the selected working levels. Finally, constraints (2.3) compute the deviation of the wood densities within each cooker with respect to the working level, putting the result in $VDev_{flt}$ - for the selected factory working level - or in $VSlack_{flt}$ – for the other working levels. Constraints (2.5) ensure that the model chooses the level which brings about minimum density deviation. This constraint is needed because, in particular situations, the model could try to select a level satisfying the consecutive level constraints (2.6) even if they do not match the real level in the digester.

A difficult part of this model is the definition of parameter M used in Equation (2.4). Its value should be big enough to make (2.4) a valid equation but cannot be too big in order to avoid floating point errors and a bad scaled matrix. Therefore, this parameter is computed for each factory and period multiplying the greatest deviation σ_{wl} by the demand for the given factory and period.

The second set of equations contains the consecutive level constraints (2.6) which prevent the model to change more than one level in two consecutive periods. This constraint is needed to force the model to make smooth changes in the working level of each digester. The third set of constraints is rather classical. Equations (2.7) ensure that harvested areas can only be transported in the allowed periods (i.e. during their transportation window) and demand satisfaction constraints (2.8) force the model to process the predetermined volume of wood at each digester for each planning period. Equations (2.9) and (2.10) ensure, respectively, that the inventory for each period is below the stockyard capacity and that the availability of wood at each area is respected. Finally, inventory balance is kept by equations (2.11).

Our preliminary results showed that standard branch and bound software could only solve small sized instances of PPSP (up to 4 periods). However, real instances of PPSP require from 12 to 16 planning periods. In order to overcome this important drawback, a rolling horizon heuristic based on the previous model is presented in the next section.

Rolling horizon heuristic

A natural decomposition approach to the PPSP consists in solving each period – or group of periods – individually. In the following, we propose a α/β decomposition approach which divides the initial problem into a series of smaller sub problems which are sequentially solved by a standard branch and bound algorithm. Parameters β and α designate the size of each sub- problem (i.e. the number of periods to solve at each iteration) and how many of them are kept as part of the solution ($\alpha \leq \beta$), respectively. In particular, the heuristic begins solving the first sub problem containing the β first periods. Then, the first α periods of the solution are kept and the $(\beta - \alpha)^{th}$ period becomes the first one in the next sub problem. This procedure is repeated until all the blocks have been scheduled. The following pseudo code illustrates the decomposition heuristic.

Algorithm 1 Rolling Horizon Heuristic

```
1: t = 1
2: repeat
3:    Solve PPSP containing periods $[1 + \alpha(t-1)]$ to $[min(\beta + \alpha(t-1), |T|)]$
4:    Add periods $[1 + \alpha(t-1)]$ to $[\alpha + \alpha(t-1)]$ to the solution
5:    t = t + 1
6: until $[\alpha + \alpha(t-1)] \geq |T|$
7: End
```

At the end, at most $\lfloor T/a \rfloor$ sub problems will have been solved, where $\lfloor x \rfloor$ is the largest integer smaller than x. Also, note that the last sub problem may include less than β periods. Yet the choice of parameters β and α remains a nontrivial issue, because a delicate trade-off needs to be made between accuracy, which increases with β, and the computational effort required by the method. Additional discussion on this difficult issue is provided in the next section.

2.4 Practical application and numerical results

The purpose of this section is twofold. First, we intend to assess the performance of the proposed heuristic approach to solve PPSP in terms of both the numerical value of the solutions obtained and the computational effort required. This is an important issue because, as we were not able to solve to optimality any real sized instance, further analyses concerning the practical application of our model – the second goal of this section – will be made using approximated solutions. In particular, in the second part of the section we solve a real instance using the heuristic approach and we compare it to a solution provided by the plant engineers. This analysis evaluates how our solution fits the engineers production objectives – maximize wood homogeneity and minimize level changes between periods – rather than their numerical objective values.

Assessing the performance of the approximated method

Our test data is based on real information provided by one of the largest pulp producers in Brazil. All the instances solved here consider a three digester setting, each with production ranging from 5000 m^3 to 7000 m^3 per day, operating on a 24 hours per day, 7 days per week basis. Inside the planning horizon, the company manages around 400 harvest areas, each with a volume of about 6000 m^3. Since the quantity of available wood varies in time according to the number of harvested forests – which is an exogenous decision to the model – we defined the parameter ρ as the ratio between the total demand and the total volume of wood available. Therefore, setting ρ to different values during the instances generation allows us to reproduce different supply scenarios. We observed that the value of ρ strongly influences the difficulty of the instance to solve. If ρ is close to 1, the problem is very tight and almost every piece of available wood will have to be transported in order to meet demand. On the other hand, when this ratio reduces to 0, the number of feasible solutions increases considerably as more wood is available. Finally, we modeled the wood basic density of these forests as a random variable with a normal distribution which parameters were obtained from historical observations.

The discretization of density values into working levels is an important issue, since the number of levels directly affects the quantity of binary variables in the model and therefore the computing effort required to solve the model. However, one can expect that the larger the number of levels, the better the approximation of the real problem. Taking this into account and after discussions with the plant engineers, we concluded that 5 levels was a good trade-off between both criteria.

Therefore, randomly generated instances were grouped according to the number of periods considered (4 or 8), the number of available harvested units (200 or 400), and different ρ values (29, 54, 78 or 96%). For each category, three instances were generated according to the volume of wood available (uniformly distributed between 5000 and 7000) and average wood density (normally distributed with average basic density 530 and deviation σ of 50 or 100) for a total of 96 instances. The weights $p1$ and $p2$ in the objective (2.1) were set to 100 and 10 respectively after some preliminary tests. All the tests were conducted on a SUN Ultra-5 400MHZ CPU with 256 MB of RAM, using Cplex 8.1 and the maximum allotted computation time was limited to 36000 seconds (10 hours) and an optimality gap equal to 0.1%.

Our computational experiments seek two main goals. First, they aim at identifying the range of problem sizes for which PPSP can be solved to optimal efficiently and therefore substantiate whenever the call for heuristic methods, as the one presented in

Section 2.1, is justified. The second goal deals with the selection of the parameters for the rolling horizon heuristic. To this end, all possible combinations of values for which $\beta \leq 3$ and $\alpha < \beta$ were tested. The value of the upper bound for β was empirically set to ensure that each sub-problem could be solved within a reasonable time. Computational results over the test bed for 4 and 8 periods are presented in Table 2.1.

Instance				Exact Algorithm				$\alpha = 1; \beta = 2$		$\alpha = 1; \beta = 3$		$\alpha = 2; \beta = 3$	
$\|T\|$	$\|W\|$	σ	ρ	GAP	Δ	Time	#OPT	Δ	Time	Δ	Time	Δ	Time
04	200	050	29	0.08%	0.01%	3261	3	0.00%	98	0.01%	586	0.01%	574
04	200	050	54	0.10%	0.00%	9654	3	0.05%	181	0.06%	1418	0.12%	1310
04	200	050	78	0.10%	0.00%	6924	3	0.17%	263	0.10%	1576	1.27%	1375
04	200	050	96	0.10%	0.00%	9330	3	0.56%	402	0.32%	1916	0.64%	1648
04	200	100	29	0.10%	0.02%	11055	3	1.04%	156	0.09%	1661	0.11%	1644
04	200	100	54	0.10%	0.00%	7928	3	0.37%	340	0.14%	2879	0.18%	2852
04	200	100	78	0.10%	0.00%	14150	3	2.44%	576	0.70%	3989	1.69%	3869
04	200	100	96	62.93%	0.00%	36007	0	6.51%	805	1.28%	6329	3.32%	5930
04	400	050	29	0.10%	0.00%	11467	3	0.03%	269	0.01%	1999	0.01%	1970
04	400	050	54	0.10%	0.00%	11631	3	0.67%	460	0.03%	3170	0.07%	3043
04	400	050	78	0.09%	0.00%	16063	3	4.85%	752	0.39%	4005	0.81%	3437
04	400	050	96	0.09%	0.00%	27632	3	3.73%	1277	0.24%	7541	0.46%	6772
04	400	100	29	0.08%	0.00%	23990	3	0.28%	344	0.02%	3785	0.00%	3711
04	400	100	54	0.10%	0.00%	10511	3	0.10%	877	0.08%	5108	0.08%	5039
04	400	100	78	24.77%	0.00%	29815	1	1.15%	1643	0.63%	9010	1.57%	8343
04	400	100	96	91.16%	0.00%	36004	0	3.18%	2403	2.37%	14677	3.83%	14268
08	200	050	54	96.50%	0.72%	36012	0	0.02%	313	0.28%	1782	0.13%	1533
08	200	050	78	97.38%	0.27%	36009	0	0.10%	545	0.00%	4297	0.07%	3285
08	200	050	96	96.98%	1.08%	36008	0	1.51%	1097	0.29%	10105	0.94%	6930
08	200	100	54	99.63%	0.67%	36010	0	0.09%	392	0.04%	4196	0.22%	3051
08	200	100	78	99.24%	0.99%	36009	0	0.10%	959	0.19%	8142	0.27%	7137
08	200	100	96	99.47%	2.08%	36007	0	3.49%	1411	0.27%	12341	1.02%	12391
08	400	050	54	99.44%	0.18%	36008	0	0.34%	1108	0.16%	9589	1.08%	7971
08	400	050	78	97.35%	0.49%	36007	0	0.14%	2098	0.05%	13647	0.10%	11267
08	400	050	96	98.37%	2.03%	36005	0	1.17%	4405	0.27%	22506	0.74%	20103
08	400	100	54	99.82%	0.48%	36008	0	0.44%	1405	0.03%	11807	0.45%	12447
08	400	100	78	99.32%	0.72%	36007	0	0.41%	3750	0.01%	16806	0.25%	16238
08	400	100	96	99.69%	2.74%	36005	0	3.70%	7156	0.04%	25523	1.00%	22114
Average				48.69%	0.44%	24911	1.4	1.31%	1267	0.29%	7514	0.73%	6795
Average $\|T\| = 4$				11.26%	0.00%	16589	2.5	1.57%	678	0.40%	4353	0.89%	4112
Average $\|T\| = 8$				98.60%	1.04%	36008	0.0	0.96%	2053	0.14%	11728	0.52%	10372

Table 2.1: Computational results

The left section of Table 2.1 identifies the characteristics of the instances considered in terms of the number of periods $|T|$, the number of harvested areas $|W|$ and the values of parameters σ and ρ. For each combination of T, W, σ and ρ we generated, 3 different instances. Thus, each line in Table 2.1 reports the average results obtained by the CPLEX standard branch-and-bound and three different settings of the rolling horizon heuristic ($\alpha = 1$, $\beta = 2$; $\alpha = 1$; $\beta = 3$ and $\alpha = 2$; $\beta = 3$). The column GAP reports the average gap, where each gap instance is computed as (best integer solution – best lower bound) / best lower bound) obtained by CPLEX after 10 hours of computing. Columns Δ also compute average gaps, but this time with respect to the best integer solution over all the methods, and the #OPT column shows the number of instances solved to optimality by Cplex, from a total of 3 instances. Finally, the Time columns report the average computational time in seconds for each of the four methods.

As illustrated in column #OPT of Table 2.1, one may expect to solve to optimality (in fact, near optimal solutions) only the smallest sized instances of PPSP i.e. four periods and ρ ¡ 96%. Therefore, solving real sized instances (12 to 16 periods) would require either improvements in the proposed formulation and the development of specialized exact algorithms (e.g. Branch & Cut) or the use of heuristic methods as the one presented in Section 2.3. It can also be observed from Table 2.1 that the instances having the largest value of ρ (96%) seem to be more difficult to prove optimality.

When comparing the performance of the four proposed methods for the 8 periods instances, one may observe that the exact algorithm is dominated by the heuristic methods which provide solutions of comparable quality to the standard branch and bound in a third of the time. These heuristics can be divided into two main groups concerning their β values. The results provided by the heuristic set with $\beta = 3$ are slightly better than the $\beta = 2$. On the other hand, $\beta = 2$ can find, for most of the test instances, similar results in a fraction of the time used by $\beta = 3$.

In the same Table 2.1, the values in bold are the best solution when comparing the four solving approaches. One can notice that for $|T| = 4$ instances Cplex finds the best known solution for 15 of 16 instances. While for $|T| = 8$, the heuristic method set with ($\alpha = 1$; $\beta = 3$) finds the best solutions for 10 out 12 instances.

As the results seem more interesting for the largest instances, additional test problems were generated and solved with CPLEX and the proposed heuristic. These tests basically differ by the number of periods in the planning horizon, in this case 12 periods. Table 2.2 resumes the results obtained for these new instances where, due to the larger size, one instance of each category was solved instead of three as in Table 2.1.

Table 2.2 also reports very large values in column GAP, confirming the very slow convergence of the exact method unable to prove the optimality for any of the tests. One can also remark that in this case the Rolling Horizon heuristic performs better than the exact algorithm – achieving improvements ranging from 2.8% up to 54% - using considerably less computational effort.

Now let us consider again the three versions of the heuristic approach in order to discuss the performance of the proposed sets of α and β values. Not surprisingly, the results produced by the ($\alpha = 1$, $\beta = 3$) configuration dominates the other settings finding the best solution for 8 out 12 instances, but the one with ($\alpha = 1$, $\beta = 2$) obtains

Instance				Exact Algorithm			$\alpha = 1; \beta = 2$		$\alpha = 1; \beta = 3$		$\alpha = 2; \beta = 3$	
$\|T\|$	$\|W\|$	σ	ρ	GAP	Diff. Best	Time	Diff. Best	Time	Diff. Best	Time	Diff. Best	Time
12	200	050	63	98.68%	21.42%	36016	0.34%	170	0.00%	726	0.29%	493
12	200	050	78	99.71%	04.03%	36008	0.01%	599	0.00%	5192	0.08%	3296
12	200	050	96	99.09%	09.98%	36008	0.00%	875	0.61%	5977	1.03%	4798
12	200	100	63	98.77%	54.26%	36013	7.74%	197	0.00%	947	3.23%	653
12	200	100	78	98.94%	43.35%	36009	27.27%	320	0.00%	1516	0.04%	1222
12	200	100	96	99.35%	16.17%	36007	0.00%	714	1.29%	5279	0.05%	3730
12	400	050	73	98.88%	41.25%	36010	2.71%	543	0.00%	2539	2.71%	1835
12	400	050	78	99.67%	02.86%	36008	0.00%	1237	0.14%	10698	0.20%	6730
12	400	050	96	99.37%	10.58%	36006	0.00%	2764	0.44%	21321	0.36%	12919
12	400	100	66	99.50%	47.54%	36008	0.32%	1041	0.00%	6171	0.00%	4117
12	400	100	78	99.76%	15.99%	36008	0.50%	2171	0.00%	14255	0.02%	11893
12	400	100	96	99.36%	17.23%	36007	1.57%	3661	0.00%	22626	2.10%	11717
Average				99.26%	23.72%	36009	3.37%	1191	0.21%	8104	0.84%	5283

Table 2.2: Computational results for the 12 period test bed

solutions of almost the same quality in a fraction of the time. The third configuration ($\alpha = 2$, $\beta = 3$) can be discarded. These results confirm that the PPSP is a very difficult problem, with a very weak linear relaxation, and the constructive method proposed here performs well even for the largest instances.

Practical application

In order to evaluate the practical contribution of the research presented, this section considers a particular instance provided by our industrial partner and compares the solution reached by the ($\alpha = 1$, $\beta = 3$) heuristic to the one obtained when applying the ad-hoc method (to be described later) used by the plant engineers. Solutions are compared in terms of the two goals identified by the plant managers: the homogeneity of the wood processed in the digester at each period, and the minimization in the digesters set point variations along the planning horizon. To support such a discussion, Figures 2.2 and 2.3 show bubble graphics where the x and the y-axis respectively indicate the period and the basic density. Each bubble represents one allocated area, the bubble's size being proportional to the area volume. To keep Figures 2.2 and 2.3 readable, only a part of the solutions – the one corresponding to the first digester – is presented.

Figure 2.2 is a typical solution produced by the constructive procedure used currently by the factory managers which allocates greedily the harvested areas with the least available time windows first. The density aspect is taken into consideration by allocating areas with high, medium and low densities into different factories. Figure

2.3 illustrates the solution provided by the Rolling Horizon with parameters$\alpha = 1$ and $\beta = 3$.

As one may observe, Figure 2.2 shows bubble dispersion greater than in the Rolling Horizon solution. Therefore, the solution shown in Figure 2.3 is clearly preferred by plant managers according to the wood homogeneity goal. Needless to say, a similar behaviour is observed when comparing the solutions for digesters 2 and 3.

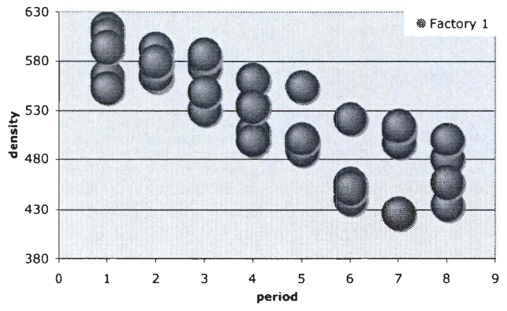

Figure 2.2: Greedy constructive solution

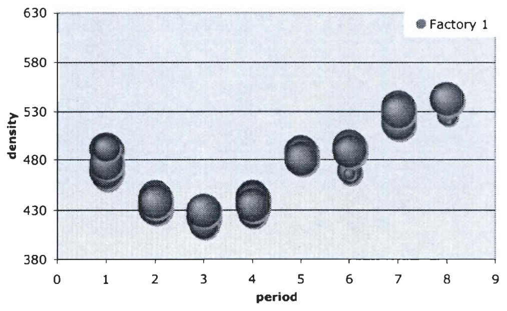

Figure 2.3: Rolling Horizon Solution

Figures 2.4 and 2.5 illustrate the evolution of the digester's set points along the time horizon. In both Figures, each line corresponds to a digester, the y-axis being the average wood density inside the digester for each period (x-axis). Comparing these Figures, one can notice how the average density for each digester evolves along the time. Both cases present smooth variations most of the time, without big jumps from low to high densities or vice-versa, satisfying the manager's requirement.

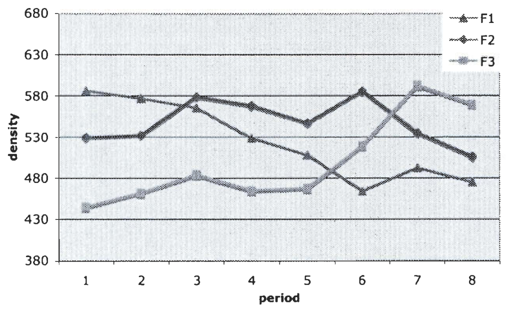

Figure 2.4: Average densities for the greedy solution

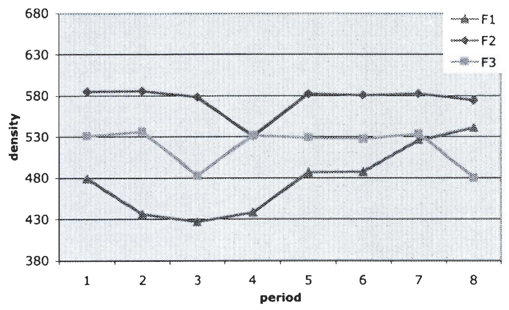

Figure 2.5: Average densities for the Rolling Horizon (3,1) solution

2.5 Conclusion

This paper focuses on the coordination of wood transportation decisions in order to minimize the wood density variance within pulp digesters. This production objective has an influence on the quality of the final pulp that is obtained but can also allow companies to achieve considerable savings, including less consumption of chemicals a better setup of the digester pressure and temperature to the raw product consumed. However, introducing density variance within the objective function naturally leads to a non-linear mathematical model. A linearization strategy, based on the discretization of the continuous density values into discrete density levels, allows us to formulate a linear integer mathematical model. This model accounts for classical constraints such as wood availability and system capacities, but also considers specific engineering plant constraints such as consecutive level restrictions that aim to smooth digesters operation by avoiding drastic changes on the digesters settings between consecutive production periods. A decomposition approach that uses the formulation proposed was also developed for solving large, real sized, instances. Based on data provided by one of the largest pulp and paper producers in Brazil, random testing instances were generated. The heuristic method shows a very good performance solving the whole test bed accurately and efficiently.

References

Biermann, C. J. (1996). *Handbook of Pulping and Papermaking.* Academic Press, second edition.

Duffy, G. G. and Kibblewhite, R. P. (1989). A new method of relating wood density, pulp quality and paper properties. *Appita Journal*, 42(3):209–214.

Foelkel, C. E. B., Mora, E., and Menochelli, S. (1992). Densidade básica: sua verdadeira utilidade como índice de qualidade da madeira de eucalipto para produção de celulose. *O Papel*, 53:35–40.

Geoffrion, A. M. and Powers, R. F. (1995). 20 years of strategic distribution-system design - an evolutionary perspective. *Interfaces*, 25(5):105–127.

Karlsson, J., Ronnqvist, M., and Bergstrom, J. (2003). Short-term harvest planning including scheduling of harvest crews. *International Transactions in Operational Research*, 10:413–431.

Kennedy, J. F., Philips, G., and Williams, P., editors (1989). *Wood processing and utilization*. Ellis Horwood Limited.

Kibblewhite, R. P. and Uprichard, J. M. (1996). Kraft pulp qualities of eleven radiata pine clones. *Appita*, 49(4):243–250.

Molteberg, D. (2004). Methods for the determination of wood properties, kraft pulp yield and wood fibre dimensions on small wood samples. *Wood Science and Technology*, 37(5):395–410.

Murray, A. T. and Church, R. L. (1995). Measuring the efficacy of adjacency constraint structure in forest planning-models. *Canadian Journal of Forest Research-Revue Canadienne De Recherche Forestiere*, 25(9):1416–1424.

Murray, A. T. and Church, R. L. (1996a). Analyzing cliques for imposing adjacency restrictions in forest models. *Forest Science*, 42(2):166–175.

Murray, A. T. and Church, R. L. (1996b). Constructing and selecting adjacency constraints. *Infor*, 34(3):232–248.

Ronnqvist, M. (2003). Optimization in forestry. *Mathematical Programming*, 97(1-2):267–284.

Schimleck, L., Payne, P., and Wearne, R. (2005). Determination of important pulp properties of hybrid poplar by near infrared spectroscopy. *Wood Fiber and Science*, 37(3):462–471.

Slats, P. A., Bhola, B., Evers, J. J. M., and Dijkhuizen, G. (1995). Logistic chain modeling. *European Journal of Operational Research*, 87(1):1–20.

Weintraub, A., Church, R. L., Murray, A. T., and Guignard, M. (2000). Forest management models and combinatorial algorithms: analysis of state of the art. *Annals of Operations Research*, 96:271–285.

Williams, M. F. (1994). Matching wood fiber characteristics to pulp and paper processes and products. *Tappi Journal*, 77(3):227–233.

Chapter 3

Restricted Space Search Heuristic: Application to a Pulp Production Scheduling Problem

Résumé - Nous proposons un algorithme basé sur la synergie, la complémentarité et l'échange multi-directionnel d'informations entre plusieurs méthodes de résolution dans un problème d'ordonnancement rencontré dans la production de pâte à papier. La méthode hybride proposée s'articule en deux phases. Dans la première phase, deux méthodes heuristiques interagissent pour identifier des régions prometteuses de l'espace des solutions ensuite explorées par une méthode exacte. Ces espaces restreints de recherche (Restricted Spaces) se caractérisent par une taille suffisamment petite pour être complètement explorée en un temps raisonnable, et par une forte probabilité d'y trouver un optimum globale, sinon une bonne solution optimale locale. Cet algorithme a été testé de manière à en déterminer ses paramètres et à mesurer l'utilité des différentes procédures de recherche. Ces tests montrent l'efficacité et la robustesse de l'algorithme qui permet d'obtenir de très bonnes solutions sur des instances de grandes tailles avec des temps de calculs compétitifs.

Restricted Search Space Heuristic: Application to a Pulp Production Scheduling Problem

José Eduardo Pécora Jr [†▽], Angel B. Ruiz[†▽], Patrick Soriano[‡▽]

[†]Dép. opérations et systémes de décision, Faculté des sciences de l'administration, Université Laval
Québec (Qc) Canada G1K 7P4
eduardo.pecora@centor.ulaval.ca, angel.ruiz@ulaval.ca

[‡]Service des Méthodes quantitatives de gestion, HEC-Montréal
3000, ch. Côte-Sainte-Catherine , Montréal (Qc) H3T 2A7, Canada
patrick@crt.umontreal.ca

[▽]CIRRELT - Centre interuniversitaire de recherche sur les reseaux d'entreprise, la logistique et le transport, Université de Montréal
CP 6128 Succursale Centre-ville, Montréal (Qc) H3C 3J7 Canada

Abstract

We propose a hybrid algorithm, which exploits the synergy and the complementarity of several solution methods, to tackle a difficult real world problem arising in the context of pulp and paper production. The proposed hybrid method is built around two distinct phases: first, two heuristic algorithms interact to identify a promising reduced search space that will then be thoroughly explored in the second phase. Highly desired characteristics of this Restricted Space (RS) are: 1) it should be small enough so that the second phase may perform a thorough search in reasonable time, and 2) it should have a high potential of containing the global optimum or at least a good local optimum. Extensive computational experiments have been conducted in order to identify the usefulness of the different search procedures used in the global method. The computational tests show the robustness and the efficiency of the method, which produces high quality solutions especially for the largest test instances in very competitive computation times.

3.1 Introduction

The Pulp Production Scheduling Problem (PPSP) considered here arises in the context of a large Brazilian pulp and paper producer. The process for the manufacture of paper pulp is based on the transformation of wood into a fibrous material, known as paste, pulp or industrial pulp. This process begins in the forests, where trees are harvested and then wait for a transportation window. Logs are then transported to production plants and sent through chippers to be slashed. The wood chips are then fed to the digester area, where the cooking process (i.e. transformation or Kraft process) begins. The digester is a pressure vessel where the chips and chemicals, known as white liquor, are continuously introduced through an opening at the top. The Kraft process consists basically in submitting the chips to the combined action of sodium hydroxide, sodium sulfite and steam inside the digester in order to separate the lignin that binds the fibre in the wood, for more information on the Kraft process see Biermann (1996). This liberated fibre is the industrial pulp.

During the Kraft process the basic density of the chips plays an important role, as described by Foelkel et al. (1992) and Williams (1994). If the range of the basic densities of the chips present in a digester is wide, setting the parameters for the thermochemical process (i.e. quantities of chemicals, pressure and temperature) to appropriate values will not be possible. Consequently the digester will contain a mix of under and over cooked wood chips and the transformation will therefore be inefficient: lower percentages of lignin will be extracted from the wood while requiring increased energy consumption and generating a lower quality pulp. In addition, pulp produced with high density wood has a totally different use than pulp produced with low density wood: the former is mostly used to manufacture cardboards while the later is used for tissues, see Kennedy et al. (1989). Therefore, the homogeneity of the basic density of wood chips is a highly desired factor in the pulp and paper industry.

Fisher (1958), studied maximum homogeneity problems in the context of partitioning data samples into groups in statistics. He divided them into two classes. The first class deals with what he calls *unrestricted problems*, i.e. problems where no conditions are imposed *a priori* on the observations to be grouped. These problems can be easily solved by approaches based on sorting and contiguous partition of the sorted observations. The second class deals with *restricted problems*, i.e. problems where conditions are imposed on the observations to be grouped, and for these, no simple general solution method is available. The PPSP falls within this second class. Of course, maximizing the homogeneity of a partition is equivalent to minimizing the sum of the variance of its elements.

As was mentioned above, in order to attain pulp quality requirements while minimizing production costs one needs to use the most homogeneous mix of wood chips possible. Unfortunately, to the best of our knowledge there are no published works which provide a function measuring the economic impact of variations in the homogeneity or variance of wood chips basic density. Therefore, we will minimize this variance, a non-linear function, as optimization criterion for the PPSP. The main decision of the PPSP basically consists in selecting which harvested areas should be used in a given period at a given production facility in order to maximize the homogeneity of the basic density of wood chips inside each digester. The PPSP planning horizon starts at the end of the short-term forest planning, as described in Ronnqvist (2003), and does not account for transportation issues such as those presented in the work of Karlsson et al. (2003). In fact, the PPSP uses the output of the short-term harvest planning as its input (the available harvested areas waiting for transportation) and provides the data for transportation models detailing in which week each harvested area should be transported and to which facility it should be fed.

A complete description of the PPSP can be found in the work of Pécora et al. (2007) where the authors provide a mathematical model and propose a constructive heuristic to solve it. As they describe, the exact methods applied to the PPSP have a very limited efficiency, mostly due to the slow convergence of the linear relaxation. In order to overcome this drawback, we propose a new hybrid heuristic based on the synergy and complementarity of informations provided by several solution approaches that are applied to the problem. The first phase of the proposed algorithm starts by hybridizing two heuristic methods which will iterate and cooperate in order to cut out uninteresting regions of the solution space, identifying in the process what we will refer to as a promising Restricted Space (RS). The aim of this first phase is to reduce the solution space to be searched later on so it can be handled with a reasonable effort, but to do it in a way that maintains a high potential of keeping the global optimal solution or at least very good local optima within the RS. Once a satisfactory RS is obtained, the second phase is called to thoroughly explore this subregion of the initial solution space and return the best solution found within it.

Within the operational research literature, the term “hybrid solution approach” is generally applied to the combination of two (or more) different solution methods with the goal of encouraging the exploration of new search regions, of escaping the local attraction of optimal points or of generating cuts, among others. Clearly, the approach proposed here falls within this rapidly growing field. However, it can be argued that its fundamental principle differs somewhat from the ones that have underlined the majority of hybrid approaches up to now. The number of possible hybrid approaches is huge, basically any two solution methods can be put together to create a hybrid method.

Therefore, the literature on hybrid algorithms is rather large and growing fast. This special class of solution methods has been applied on a wide range of operational research problems achieving very good results both on classical ones such as the knapsack problem (da Silva et al., 2007), the TSP (Nguyen et al., 2007), the p-median (Resende and Werneck, 2004) as well as on real world applications such as health care (Bertels and Fahle, 2006 ; Pécora, 2002), the optimization of credit portfolios (Schlottmann and Seese, 2004), and aircraft scheduling (Gronkvist, 2006), to cite just a few. Traditionally hybrid approaches merged exact and/or heuristic algorithms (Gallardo et al., 2007), but more and more work is carried out on hybrid methods involving other approaches such as simulation (Peng et al., 2006), constraint programming (Correa et al., 2004 ; Hooker, 2006), neural networks (Sahoo and Maity, 2007 ; Pendharkar, 2005) and multi-agent approaches (Yan and Zhou, 2006). It is beyond the scope of this paper to make an exhaustive review of all hybrid approaches or to formalize or classify them. For such matters, the interested reader is referred to the works of Talbi (2002), which provides an extensive classification of Hybrid Methods using heuristics, and of Puchinger and Raidl (2005), which focuses on hybrid heuristic-exact methods.

The remainder of this paper is organized as follows. Section 3.2 will introduce the two mathematical models used in this work. Section 3.3 will describe the overall structure of the proposed hybrid algorithm putting the emphasis on the multi-directional exchange of information between the methods and on how to determine the RS. Section 3.4 will detail the Space Restriction Phase of the proposed algorithm. Computational results will be presented and analyzed in Section 3.5 and, finally, Section 3.6 will conclude the paper.

3.2 Mathematical formulation

This section presents two mathematical models that schedule known amounts of available wood into different digesters and periods, with the goal of minimizing the chips basic density variance within each digester-period pair.

First, using the density variance as the objective function naturally leads to a nonlinear mathematical model. Let T be the planning horizon, where each time period t is one week long, and W be the set of harvested areas waiting for transportation. Note that not all harvest areas are available for transportation in a given planning horizon T: some forests are very young, some are very old, and others do not meet the minimum required quality to produce pulp. To each harvest area $w \in W$ is associated a transportation window identified by $\tau_{wt} \in \mathbb{B}$, which takes value 1 when the harvest area w is available to transport at period t and zero otherwise. Each harvest area w is also characterized by the volume of wood available v_w and the average wood basic density δ_w within that area.

On the manufacturing side, we consider a single plant equipped with several digesters, or pulp processing units, identified by index $f \in F$. The pulp demand for every digester and period, denoted by d_{ft}, is deterministic, assumed to be less than the corresponding digester production capacity and is given as a number of harvest areas. The volume of available wood for each harvest area v_w is not explicitly considered in the demand satisfaction constraints because, in the particular context modeled here, all harvest areas are planted specifically for pulp generation and their volumes are basically the same. Thus one can easily simplify these constraints by considering the demands as a fixed number of harvest areas without much loss of precision. However, the volume plays an important role when computing the average densities, $\hat{\delta}_{ft}$. As will be explained in Section 3.3, this parameter is the one which will drive the search and we want to keep it as accurate as possible. We will hence consider the exact harvest area volumes when computing them.

The PPSP can therefore be formulated as the following non-linear integer model where the only decision variables are the $x_{wft} \in \mathbb{B}$ which take value 1 when the harvest area w is consumed by factory f at period t, and zero otherwise.

NL-Model

$$\text{Minimize} \quad Z^{NL} = \sum_w v_w \sum_t \sum_f (\delta_w - \hat{\delta}_{ft})^2 x_{wft} \tag{3.1}$$

$$\begin{aligned}
\text{Subject to:} \quad \sum_f x_{wft} \leq& \quad \tau_{wt} && \forall\, w, t; && (3.2)\\
\sum_f \sum_t x_{wft} \leq& \quad 1 && \forall\, w; && (3.3)\\
\sum_f \sum_t^{t'} x_{wft} =& \quad 1 && \forall\, w \in W_{t'}, \forall t' \in T; && (3.4)\\
\sum_w x_{wft} =& \quad d_{ft} && \forall\, f, t; && (3.5)\\
\frac{\sum_w v_w \delta_w x_{wft}}{\sum_w v_w x_{wft}} =& \quad \hat{\delta}_{ft} && \forall\, f, t; && (3.6)\\
x_{wft} \in& \quad \mathbb{B}; \hat{\delta}_{ft} \in \mathbb{R}_+ && && (3.7)
\end{aligned}$$

The objective function (3.1) is the sum of the squared differences between the allocated wood and the average basic density, $\hat{\delta}_{ft}$, for each factory and period. Constraints (3.2) state that each harvest area (or forest) can only be used within its time windows, while constraints (3.3) ensure that the same forest cannot be used more than once. In the real life application modeled here, certain harvest areas are required to be used before the end of the current planning horizon, generally because they have been harvested for some time already and must be transported and processed or will become unfit for processing. Set $W_{t'} \subset W$ identifies such forest units and constraints (3.4) forces the model to use all such harvest areas (i.e. those that must absolutely be used before the end of the period t'). Demand satisfaction is ensured through constraints (3.5) while constraints (3.6) compute the average basic density for each factory and period combination. Finally, constraints (3.7) define the variable ranges.

However, due to operational constraints the digesters cannot really be set to work at any precise density value within the continuous range of basic densities, but rather within a specific range of densities. We therefore apply a linearization strategy, based on the discretization of the continuous basic density range in $|L|$ different states or *levels* of production, as described in Pécora et al. (2007). We will denote by δ_l the center value of each of these density production levels, $l \in L$, and introduce a new binary decision variable, y_{ftl}, which will take value 1 if the digester f is scheduled to operate at density level l in period t and zero otherwise. We also have two new sets of real and positive auxiliary variables: σ_{ftl} which serve to compute the deviation between the basic densities of the wood on the allocated harvest areas and the target basic density used by the digester, and s_{ftl} which act as explicit slack variable. The resulting linear integer model is formulated hereafter.

L-Model

$$\text{Minimize} \quad Z^L = \sum_t \sum_f \sum_l \sigma_{ftl} \tag{3.8}$$

$$\text{Subject to:} \quad \sum_l y_{ftl} = 1 \qquad \forall f, t; \tag{3.9}$$

$$s_{ftl} \leq M(1 - y_{ftl}) \qquad \forall f, l, t; \tag{3.10}$$

$$\sum_w v_w(\delta_w - \delta_l)^2 x_{wft} = \sigma_{ftl} + s_{ftl} \qquad \forall f, l, t; \tag{3.11}$$

$$x_{wft} \in [0,1],\ y_{ftl} \in \mathbb{B},\ \sigma_{ftl} \in \mathbb{R}_+,\ s_{ftl} \in \mathbb{R}_+ \tag{3.12}$$

and equations (3.2), (3.3), (3.4) and (3.5).

One needs to ensure that the σ_{ftl} variables compute only the deviation of the forest average density with respect to the target working level of the digester for the given period. To this end, constraints (3.9), (3.10) and (3.11) are required. Constraints (3.9) state that one and only one working level per digester and period will be chosen (i.e. the target level). Then, note that constraints (3.10) allow variables s_{ftl} to take any positive value bounded by M when digester f is not schedule to operate at level l in period t. However, s_{ftl} are forced to zero for the selected working levels. Constraints (3.11) compute the deviation of the wood densities within each digester with respect to the working level, putting the result either in σ_{ftl} — for the selected digester working level — or in s_{ftl} — for the other working levels. Finally, we find the non-negativity constraints (3.12) and the basic model constraints such as described in the NL-model.

An important remark at this point is that variables x_{wft} become continuous in this second model which has a major impact on the model by decreasing the number of binary variables. Unfortunately, even with fewer binary variables this model is still very difficult to solve to optimality, mainly because of the "Big M" constraints (3.10), which results in a very poor lower-bound: it can take hours for commercial Branch and Bound packages to prove optimality.

One may also notice that the NL-Model does not have the level variables y_{ftl}, which are only present in the L-Model. The former deals directly with the allocation variables x_{wft}, as a consequence, a solution generated by the NL-Method may not be integer feasible for the L-Method.

Therefore, these two mathematical models may be viewed as complementary because in the NL-Model the allocation variables are integers and it has a continuous basic density range. On the other hand, in the L-Model we have continuous allocation variables and a discrete basic density range. The hybrid approach described in this paper is designed to exploit this complementarity.

Note that $\hat{\delta}_{ft}$ is defined as a decision variable of the NL-Model. However, a simple substitution in the objective function will eliminate it. We chose to keep it in the model for explanation purposes. For the same reason, we will introduce a new, but redundant, variable in the L-Model; which is the basic density value associated to each density level, its definition is:

$$\delta^L_{ft} = \sum_{l \in L} \delta_l \, y_{ftl} \tag{3.13}$$

where δ_l is the basic density target value associated with each discrete working level $l \in L$.

The variable $\hat{\delta}_{ft}$ defined in the NL-Model plays the same role for its respective model.

$$\delta_{ft}^{NL} = \hat{\delta}_{ft} \qquad (3.14)$$

These two variables give us the density's working level at each digester and period for their respective models. The comparison between δ_{ft}^{L} and δ_{ft}^{NL} will identify the next search direction, as will be explained in the next section.

3.3 The hybrid algorithm

This section provides a macro view of the proposed hybrid algorithm, detailing the main structure, the protocol that manages the exchange of information and how the RS is generated. Description of each of its component methods is provided in the next section.

Overall structure

The hybrid algorithm presented here is comprised of three solution methods. It can be divided into two phases, the space restriction phase and the search phase. The first phase includes two solution methods, where each one is designed to tackle one of the models described in the previous Section; they are correspondingly named L-Method and NL-Method. These two solution methods will interact by exchanging information, aimed at generating not just good solutions, but also a Restricted Space (RS) which although being small enough, will also have a high possibility of containing near optimum solutions. This RS will then be thoroughly explored in the second phase. Figure 3.1 shows the components within the overall algorithm.

Each solution method has a specific role and therefore requires different inputs and produces different outputs, as shown in Figure 3.1. The procedure starts with the L-Method, that receives the whole search space as input. The NL-Method will receive the information generated by the L-Method as input and will explore not just better solutions but will also try to determine interesting search directions to be passed back

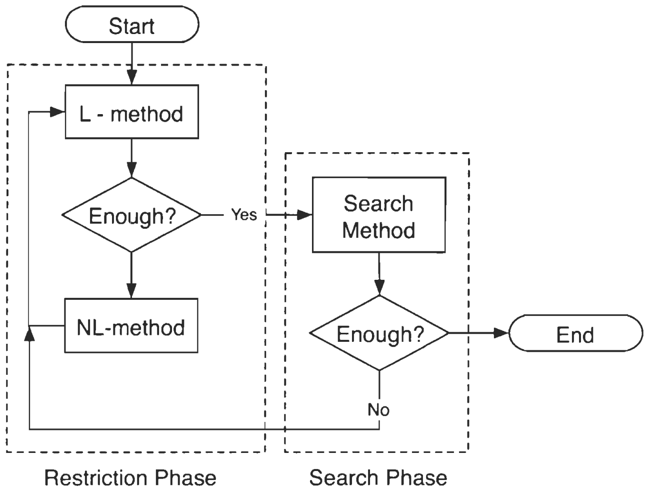

Figure 3.1: Cooperative algorithm macro schema

as input to the L-Method. This iterative process is repeated a number of times, once it is finished an RS is generated based on the information gathered. The RS is then passed on to the search procedure which will thoroughly explore this RS. Table 3.1 summarizes the expected inputs, outputs as well as the objectives of each solution method.

	Method	Input	Output	Objectives
Phase I Restriction	L-Method	Search direction	A solution following a given direction	Find a good solution inside a bounded search space following a given search direction
	NL-Method	An initial solution containing the starting basic densities	An improved solution and a search direction	Provide an improvement of the given solution and a new search direction to be explored
Phase II Search	Search Method	RS	The best solution found within the RS	Explore the RS thoroughly

Table 3.1: Summary of the solution methods

Exchange of information within the restriction phase

This subsection will discuss several variants of how, where and when to generate the information inside the restriction phase. While the next subsection will detail the generation of information from the restriction to the search phase, the RS.

The multidirectional exchange of information is a key characteristic in this hybrid algorithm. The two methods, L-Method and NL-Method, will exchange information in an iterative way with the main objective of defining an RS.

Independently the solution approach used as L-Method, any feasible solution to the L-Model provides a unique level for each $(digester, period)$, through the variable y_{ftl}, and this level has a unique density δ^L_{ft} associated. This resulting density, called *target density*, will be the information passed to the NL-Method. This one will allocate the forests as close as possible from the target density in order to minimize the total variance. Therefore, any solution method for the NL-Model will provide us with a feasible allocation of the forests for each $(digester, period)$, and the density variables δ^{NL}_{ft} can be easily computed by means of Equation 3.14.

Let Δ_{ft} be defined as

$$\Delta_{ft} = \delta^{NL}_{ft} - \delta^L_{ft}. \tag{3.15}$$

The Δ_{ft} measures the difference between the solutions provided by the two solution methods. When different from zero, Δ_{ft} can be viewed as a search direction, in fact it indicates that the two methods **do not agree** on the value that the working density should take and further search needs to be done to solve this issue. If the value of Δ_{ft} is positive, this means that the NL-Method proposes a higher average basic density then the L-Method. A symmetrical analysis can be done if the value of Δ_{ft} is negative.

As previously stated, for each $(digester, period)$ one has $|L|$ binary variables, y_{ftl}, and each one is associated with one density working level. In Figure 3.2 each box is a density working level that corresponds to a unique y_{ftl}, the arrow represents the Δ_{ft}, where the starting point is δ^L_{ft} and the ending point is δ^{NL}_{ft}. In the same figure one can also see the 5 possible alternatives for the values and directions of Δ_{ft}. The first two, $F1T1$ and $F1T2$ show the direction vector in the middle of the density range pointing up and down respectively. This may be interpreted as a possible region to explore, for example in $F1T1$ levels $L3, L4$ and $L5$ and for $F1T2$ levels $L3, L2$ and $L1$. For the next two, $F1T3$ and $F1T4$, the initial solution is on one of the bounds and the direction vector points outside the valid density range, this can occur because some forests have a very high or very low basic density, in this case the possible search region is restricted to a single level $L5$ and $L1$ respectively for $F1T3$ and $F1T4$. The last case $F1T5$ shows a very small direction vector, this happens when the two methods **agree** on the value for the basic density for some $(digester, period)$. This can also be used as information by constraining the search to solutions where only this level can be active, as in the example by fixing $F1T5$ at level 4.

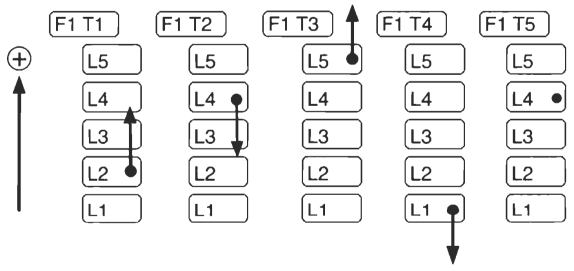

Figure 3.2: Example of relationship between the methods

As shown in Figure 3.2 there are three classes of direction vectors, the one which points to the interior of the density range and has a considerable size, the one that points outside of the density range, and finally the one which is so small that one can say that it points nowhere. In the rest of this document they will be called Direction Vector Type 1, Type 2 and Type 3 respectively.

Direction Vector Type 1: this is where the greatest source of information lies, not just because we have a direction vector which points to some part of the density range, but also because it has a minimum considerable size. In this case the decision which should be taken is to forbid the L-Method to search within the density levels located in the direction opposite to the direction vector as well as in the current density level. It is important to forbid the current density level to prevent the method from returning to the same solution. This is done by generating a constraint of the type:

$$\sum_{(ftl)\in G'_1} y_{ftl} = 0 \tag{3.16}$$

where G'_1 is the set of levels l, which should be forbidden for $(digester, period)$. If we take $F1T1$ as example the constraint will become $y_{(1,1,1)} + y_{(1,1,2)} = 0$.

Direction Vector Type 2: in this case the direction vector points outside the valid density range, here there is not much to do but to forbid all other levels except the present one. Note that this kind of decision does not contribute to diversify the next solution, since it sets the variable to its current value. In this case a constraint similar to Equation 3.16 but with a different set G_1'' of variables is generated. Taking F1T3 as example in Figure 3.2 one will have an equation like: $y_{(1,3,1)}+y_{(1,3,2)}+y_{(1,3,3)}+y_{(1,3,4)} = 0$

Direction Vector Type 3: this case is very similar to **Direction Vector Type 2** in the sense that it does not specifically drive the method towards new solutions. When the direction vector is close to zero, which means that the two methods agree on the variable value, one can set this value to 1 by forcing all others level variables to zero. Then a new constraint similar to constraint Equation 3.16, defined with the set G_1''', can also be used in this case. In our example it is characterized by the column F1T5 and the equation $y_{(1,5,1)} + y_{(1,5,2)} + y_{(1,5,3)} + y_{(1,5,5)} = 0$ will be generated.

As all the constraints have the same structure, we can aggregate them in just one large constraint,

$$\sum_{(ftl)\in G_1} y_{ftl} = 0 \tag{3.17}$$

where $G_1 = G_1' \bigcup G_1'' \bigcup G_1'''$.

These three types of Direction Vectors can be classified into two classes, the diversification (Direction Vector 1) and intensification (Direction Vector 2 and 3). This is because the first drives the solution procedure to new regions of the space, while the other two keep the solution procedure into the same region of the space.

Identifying the next search direction

The identification of the next search direction will follow two steps: first we will identify all the components of the direction vector which are *close enough* to zero (Direction Vector 3) and the components which points outside the valid range (Direction Vector 2). Then, they are fixed at their current value using Equation 3.17, this will act as an intensification procedure keeping the search into a bounded region of the space. Second, the components with *the greatest module* are used as Direction Vector 1, driving the search procedure towards a new region of the space acting as a diversification procedure.

There are two pending questions which will lead to the setting of our parameters:

- When is a component close enough to zero?
- How many components of the Direction Vector should be used as diversification?

Several different computational tests were made in order to find the best combination of these three types of direction vectors, as well as the values of their parameters. This issue deserves a whole section, and will be discussed further in Section 3.5.

Defining the Restricted Space - RS - for the search phase

In order to define the RS, we keep track of the different local optima solutions visited during the restriction phase of the hybrid algorithm. Let S be the set of the density variables δ^i_{ft} for these local optima solutions found during the cooperation L-Method $\Longleftrightarrow$ NL-Method. If the number of cycles in the restriction phase is N one will have $2N$ elements in S, half coming from each solution method.

There could be several ways of defining a restriction of the original search space based on the information gathered by the interaction between the L and NL-Methods. We propose to define our restricted search space by using the bounds:

$$\underline{\delta}_{ft} = Min\{\delta^i_{ft}, \forall i \leq 2N\}, \quad \forall ft \in F\text{x}T \tag{3.18}$$
$$\bar{\delta}_{ft} = Max\{\delta^i_{ft}, \forall i \leq 2N\}, \quad \forall ft \in F\text{x}T \tag{3.19}$$

and by imposing the following restriction on the basic density levels $\underline{\delta}_{ft} \leq \delta_{ft} \leq \bar{\delta}_{ft}$. Where δ^i_{ft} is the density variable component of solutions in S.

This region was chosen, because it contains all the visited solutions, including the best one, but it is larger than the convex envelop of these solutions.

Bounding the average basic density in the L-Model is the same as forcing the levels outside these bounds to be ZERO. This can be done by applying a constraint like Equation 3.17. Mathematically it can be defined as:

$$\sum_{(flt)\in K} y_{ftl} = 0 \tag{3.20}$$

where $K = \{(f,t,l)/\delta_l \leq \underline{\delta}_{ft} \text{ or } \delta_l \geq \bar{\delta}_{ft}, \forall (f,t,l)\}$.

The chosen region will encapsulate the density variables inside a box, making the search space much smaller than the original one, and as all the visited local optima solutions are inside the box this region contains feasible solutions. In its worst case it will return an already visited solution.

It is obvious that the greater the RS the greater the chances of finding the optimum solution, for this reason it is not trivial to estimate a good size for the RS. At this point we have a trade off between the size of the RS and the value of the best solution inside it. In an extreme case one can see the complete set of solutions as one possible RS, it is obvious that if we have enough time to solve this RS it will give us the optimum solution. On the other hand, an RS of just one solution can be solved in no time, but it may not add anything useful to our research. An ideal, but unlikely, RS contains just one solution, the global optimum.

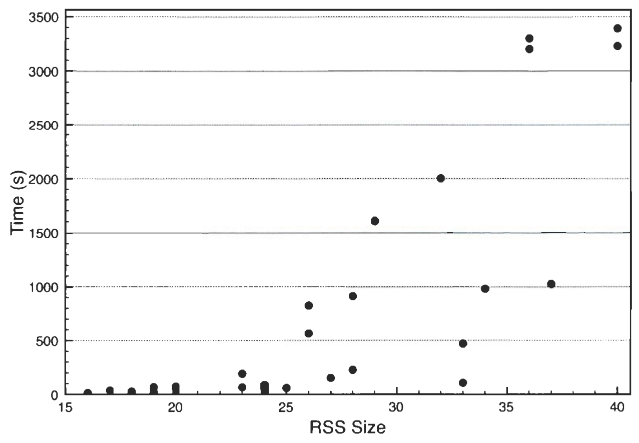

Figure 3.3: Tradeoff between the size of the RS and CPU time to solve

The tradeoff between the size of the RS and the total available time to solve it is a question of great importance in this study. In order to measure the RS size, we use the quantity of free integer variables in the RS, which has been proven as a good metric. Preliminary tests were done in order to find a good size for the RS. These results are summarized in Figure 3.3, where the x-axis is the number of free integer variables and the y-axis is the time until the optimality is proven in seconds. After these preliminary tests, it was found, that a size in the range $[17, 30]$ is the ideal for our algorithm in all instances of tests. This is the size that we will use during the computational tests as the stopping criteria for the first phase.

Search phase

The exact method used in the search phase is the commercial solver Cplex 10 set with an integrality gap of 0.1% and all other options are the default.

An illustration of the hybrid framework for the PPSP

An illustrative example is presented in Figure 3.4, where the dashed rectangles are always a solution (feasible or not). The table inside this rectangle represents each (*digester*, *period*) in this example we have 2 Factories (rows) and 3 periods (columns). One can notice that the main procedure starts with an empty (infeasible) table which after the L-Method becomes our first solution. Then this *information* goes to the NL-Method where it checks for the search directions, reallocating the forests into a NON-discrete basic density space, generating another allocation. After that, it builds the direction vector, and some of its components will become our new search directions. This information goes back to the L-Method and the algorithm restarts, stopping when the stop criterion, a given size of RS, is achieved.

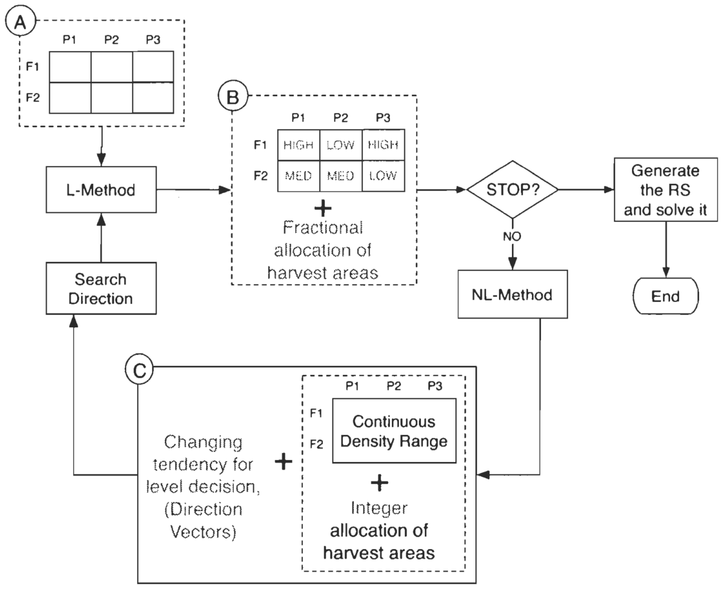
A
P1 P2 P3
F1
F2
L-Method
Search Direction
B
P1 P2 P3
F1 HIGH LOW HIGH
F2 MED MED LOW
+ Fractional allocation of harvest areas
STOP?
NO
Generate the RS and solve it
End
NL-Method
C
Changing tendency for level decision, (Direction Vectors)
+
P1 P2 P3
F1
F2
Continuous Density Range
+ Integer allocation of harvest areas

Figure 3.4: Illustration of the RS heuristic

3.4 Space restriction phase

It is not the main goal of this work to propose a single heuristic method which performs greatly for the PPSP. Instead we want robust methods which will help themselves probe the solution space in order to identify an RS, which will be solved by an experienced method in the search phase. This section will discuss the methods chose to play the role of the L-Method and NL-Method cited above. There are a number of methods which will fit the requirements for the hybridization, these two where chosen because they are fast enough and provide good solutions.

The constructive method (L-Method)

This method deals with the L-Model, it has been observed that this model has a very poor linear relaxation. In fact, the BIG-M constraints, Equation 3.10, are useful only when the variables are integer. In consequence the linear relaxation always adjusts itself to get a null objective function, taking in this way, too many iterations to increase the lower bound. Therefore, large and real world instances cannot be solved in an interesting computational time. But this problem can be heuristically solved by disaggregating the periods – each period can be solved independently of the others. Pécora et al. (2007) solved this problem by decomposing it into several subproblems each containing a small number of periods. The final integer solution is obtained by aggregation of the several partial solutions. It is clear that this procedure, as it is a relaxation of the main problem, does not guarantee global optimality but, as described by the authors, this constructive method can perform very well for real size instances. The constructive method proposed here is very similar to the one in Pécora et al. (2007), but here we choose to make the construction faster, then we solve only one period at a time having a complete solution in $|T|$ iterations.

In order to fit our needs, the L-Method must receive a search direction as input and provide a solution in this direction as output. This search direction is given by a constraint similar to Equation 3.17. The Algorithm 2 describes the constructive heuristic (L-Method) in detail.

Algorithm 2 Constructive heuristic for the PPSP - L-Method

```
Require: $G \neq \emptyset$
Ensure: The type of all the variables is set to real
1: Set $G$ as the search direction, by adding the respective constraint
2: for $t = 1$ to $t = |T|$ do
3:     Set the type of variables of period $t$ at Binary
4:     Solve the L-Model
5:     Fix the variables of period $t$ at current value
6: end for
7: Remove the constraints relative to direction $G$
8: Return the solution $\delta^i$
```

To illustrate this algorithm, consider a small problem with 3 periods. Given G as a search direction, then a relaxation of the L-Model – given by the *Period 1* set as an integer and *Periods 2 and 3* as real (relaxed) – is solved. In the next iteration, the variables in *Period 1* are fixed at their current value, while the variables in *Period 2* become integers and in *Period 3* continue to be real. Finally in the last iteration, the variables in *Period 2* are also fixed at their current values and the ones in *Period 3* set as integer, solving this last iteration provides a feasible integer solution. Note that, as a relaxation of the complete problem is solved at each iteration, the constraint defining G as the search direction is respected in each step of the algorithm.

The allocation method (NL-Method)

This second heuristic deals with the NL-Model, which is as difficult to solve as its linear homologue. Following the same idea used in the first method, we will not tackle the whole problem at once. Instead, we propose an iterative method which fixes the values of the variables $\hat{\delta}_{ft}$ and solves this projected problem – easier than the original. The values $\hat{\delta}_{ft}$ are iteratively updated and a new projected problem is solved. The algorithm cycles until convergence or for a number of iterations. The objective function to be minimized at each iteration is defined by Equation 3.1. In the first iteration the target densities, $\hat{\delta}_{ft}$, come from the solution for the L-Method and it is successively updated in the NL-method as explained bellow.

Algorithms 3 and 4 describe the allocation heuristic in detail. They take the target densities δ_f, t as input and to avoid infeasibilities we need to ensure that we have more forests, $|W|$, to allocate than the quantity of forests to be scheduled at each (digester, period), $nPos$ multiplied by $|T||F|$.

The main algorithm, starts calling the Greedy Allocation Algorithm, which will receive the target densities $\delta_{f,t}$ as input and will provide an allocation as a solution. A feasibility check is performed and any unfeasibility is corrected. Then a local search algorithm will explore the surrounding of this solution using a 2-OPT neighborhood search. When there is no improved solution the new target densities are computed as described in STEP 6, and if the stop criteria (a minimum change in the target densities) is not met, the whole algorithm starts again.

The greedy algorithm starts allocating the *Must Set forests*, i.e. the ones which due to the time windows must be allocated before the period t' in order not to be lost, ($w \in W_{t'} \subset W$)starting with the most urgent ones. After this, it will greedily allocate the other forests by minimizing the total deviation. Due to the structure of the

Algorithm 3 Allocation heuristic for the PPSP

```
Require: Target densities - $\delta_{ft}$
Ensure: $|W| \geq nPos|F||T|$
1: repeat
2:     Greedy Allocation($\delta_{ft}$)
3:     Feasibility Check
4:     Local Search
5:     for each $(f,t)$ in $F$x$T$ do
6:         $\delta_{ft} \leftarrow$ average density of allocated forests in $(f,t)$
7:     end for
8: until Stop Criteria
9: Return updated target densities $\delta_{ft}$
```

Algorithm 4 Greedy Allocation (δ_{ft})

```
1: $a_{ft} \leftarrow 0, \forall(f,t) \in F$x$T$
2: Sort the Must Set, starting from the most urgent forests
3: for each $w \in$ Must Set do
4:     $(f,t) \leftarrow Min_{f,t}\{(\delta_w - \delta_{ft})^2\}$
5:     if Number of allocated forests in $(f,t) < a_{ft}$ then
6:         Allocate $w$ in $(f,t)$
7:         $a_{ft}++$
8:     end if
9: end for
10: while $Min_{(f,t)} a_{ft} < |nPos|$ do
11:     for each $(f,t)$ in $F$x$T$ do
12:         if Number of allocated forests in $(f,t) < a_{ft}$ then
13:             $w \leftarrow Min_w\{(\delta_w - \delta_{ft})^2, \tau_{wt} = 1\}$
14:             Allocate $w$ in $(f,t)$
15:             $a_{ft}++$
16:         end if
17:     end for
18: end while
```

time windows, it is possible that this procedure leaves some *Must forests* aside. This infeasibility is corrected in Step 3 of Algorithm 3. A nuance in this allocation is that for the *Must Set* forests we will look for the pair (*digester*, *period*) with the minimum deviation and allocate the forests in it. For the other forests the contrary is done; for each (*digester*, *period*) we look in the set of possible forests to allocate and chose the one which corresponds to the minimum deviation. The algorithm is conceived in this manner to oblige the allocation of the *Must Set* forests first.

3.5 Computational experiments

This section is divided into two main parts, the first aims at assessing the potential of the RS heuristic by comparing the quality of the solutions obtained with respect to the one produced by its component methods and to a commercial Branch & Bound software. The second part will be dedicated to the parameters setting and analysis of the results.

Our test bet is composed by 52 test instances generated based on real data provided by the pulp company. These instances are divided in 4 groups concerning the number of periods considered in the planning horizon $|T| = \{4, 8, 16, 32\}$, with 16 test instances for $|T| = 4$ and 12 for each of the others. The test instances also differ by the number of available forests to allocate $|W|$ as described in Table 3.2. All test instances have a very diversified availability of wood, time windows and basic density distribution in order to provide a very wide and robust test bet. All tests were performed on dual *AMD Opteron 250 2.4 GHz* computers with Linux operating system.

$\|T\|$	4	4	8	8	16	16	32	32
$\|W\|$	200	400	200	400	200	400	400	600
#instances	8	8	6	6	6	6	6	6

Table 3.2: Tests instances description

As we lack an optimum solution, or a past best know solution, for these test problems, we will use the solution for the L-Model given by the commercial solver *Cplex 10.0* after 10 hours of CPU time as our best known solution. Inside this running time Cplex has proved optimality with a *Gap* $\leq 0.1\%$ for just 6 out of 52 test instances all belonging to the class $|T| = 4$.

There are two parameters to set, the number of components of Direction Vector type 1 and when to consider a component *close enough* to zero – Direction Vector Type 3. After empirical tests we chose the values of $\lambda = \{10\%, 25\%, 50\%\}$ of the total components of the Direction Vectors ,$|F||T|$, as potential values for the first parameter and the values $\beta = \{5, 15, 25\}$ for the second parameter. The Direction Vector is considered as null when its norm is less or equal to β. These values were chosen taking into consideration the amplitude of one discrete density level which is 50, they are equivalent to 10%, 30% and 50% of the size of the discrete level.

Preliminary tests

This subsection is dedicated to the overall tests of efficiency and optimality. To this end, we are not concerned about the parameter settings. This will be done in the next two subsections. Therefore all tests presented in this subsection were done with $\lambda = 10\%$ and $\beta = 5$.

Table 3.3 compares the RS heuristic with each of its components as well as with *Cplex 10.0* after 1 and 10 hours of CPU time. The first column of Table 3.3 names the number of periods considered in the tests. Each of the four next columns are the average normalized difference of the RS heuristic comparing with the L-Method, NL-Method and the exact method after 1 and 10 hours of CPU time. This normalized difference is given by the equation below.

$$Value = \frac{Z_{RS} - Z_{Method}}{Z_{RS}} 100\% \tag{3.21}$$

Where Z_{RS} is the objective function returned by the RS heuristic and Z_{Method} is the value given by the each of the comparing solution methods {L-Method, NL-Method, Cplex10h, Cplex1h}. The last column is the total time spent by the RS heuristic in seconds, where the maximum alloted time is 3600 seconds (1 hour) for the second phase and no limit was imposed on the first phase. Each of the following rows shows the average results for the number of instances described in Table 3.2 while the last row is the average for all the 52 test instances.

$\|T\|$	L-Method	NL-Method	Cplex 10h	Cplex 1h	Time(s)
4	-13.3%	-26.0%	1.00%	0.95%	64
8	-5.9%	-16.8%	1.05%	0.37%	563
16	-13.6%	-32.8%	0.50%	-7.08%	3188
32	-24.7%	-26.4%	-1.69%	-21.47%	3683
total	-14.3%	-25.5%	0.3%	-6.2%	1735

Table 3.3: Averages results

Looking at the two first columns one can easily see that the NL and L-Methods perform poorly if they are applied alone, being easily beaten by RS heuristic. On the other hand Cplex performs very well for the smallest instances, $T = \{4, 8\}$ arriving at better results than RS heuristic even after 1 hour of CPU time. For the $T = 16$ instances we can say that the RS heuristic, using on average 3188 seconds, arrives almost at equality with Cplex after 10 hours (36000 seconds), beating by more than 7% Cplex 1h. For the largest test instances, the RS heuristic finds, on average, solutions 1.69% better than Cplex 10h using 1/10 of its alloted time.

At this point one can see that, on average, the proposed hybrid algorithm shows robustness, giving better results than each of its components. While comparing with the commercial solver, it is able to provide better solutions for the greatest test classes in a fraction of the total alloted time for Cplex.

Table 3.4 shows the six test instances in which Cplex were able to prove the optimality. The first three columns are the number of periods, forests and instance's number. The column "GAP CPX" is the gap given by Cplex, the column "Cplex 10h" is the normalized difference of the solution returned by the RS heuristic and the optimal one. The last two columns are the time spent by Cplex to prove optimality and the total time spent by RS heuristic, both in seconds. As we can see the RS heuristic is able to identify an RS which contains the optimal solution in 5 out 6 instances using a very short computational time. However being a heuristic method, it is unable to prove the optimality.

P	W	Instance #	GAP CPX	Cplex10h	Cplex time	RS time
4	200	1	0.1%	**0.0%**	25509	13.20
4	200	2	0.1%	**0.0%**	7252	84.04
4	200	3	0.1%	**0.0%**	25455	25.01
4	200	4	0.1%	**0.0%**	23041	27.71
4	400	1	0.1%	**0.0%**	34405	29.51
4	400	2	0.1%	1.1%	13344	178.33

Table 3.4: Test instances with known optimal solution

The stopping criteria for the first phase were set at $|RS| \geq 17$ after preliminary tests, (Figure 3.3 in Section 3.3), or a maximum of four cycles. In order to verify if the size of RS used as stopping criteria is adequate we re-ran the tests which had a $|RS| < 35$ increasing the stopping criteria to $|RS| \geq 35$ or a maximum of four cycles. Table 3.5 summarizes the results found in this comparison. The three first columns of this table are the instance description, the columns $\uparrow RS$ and $\uparrow Time$ are the increase in the RS size and time spent respectively and the column $\downarrow OF$ is the decrease in the objective function.

As we can see by the values in the $\downarrow OF$ column there is little or no improvement in the value of the objective function, except for two results with 3.5% and 1.2%. On the other hand the increase in time is substantial, arriving at almost one hour in for certain instances. This results came to verify the tests made in Section 3.3, which predicted an RS of size 17 as a good compromise between quality of the solution obtained and computational time.

T	W	Intance #	↑RS	↑Time	↓OF
04	200	4	12	240	0.0%
04	200	5	4	11	0.0%
04	200	7	3	15	0.0%
04	400	14	5	48	0.0%
04	400	16	5	345	-3.5%
08	200	1	9	80	-1.2%
08	200	2	12	3486	-0.3%
08	200	3	15	3562	0.0%
08	200	4	8	1553	0.0%
08	200	5	5	48	0.0%
08	200	6	17	1693	-0.6%
08	400	8	13	2559	0.0%
08	400	9	4	2048	0.0%
08	400	12	15	3664	-0.3%
16	200	5	39	3584	0.0%
		Average	11	1529	-0.4%

Table 3.5: Testing the increase of the size of RS

Direction vector 1

In this subsection, we will explore the impact of changing the number of components of direction vectors of type 1 in the size of the RS and the quality of the final solution provided by the final exact method. After a careful preliminary tests, were chosen the values, $\lambda = \{10\%, 25\%, 50\%\}$ of the total number of components $|T||F|$, to be tested. While varying the value of λ we keep the value of our second parameter $\beta = 5$. The Direction Vector type 2 is always considered and, on the contrary of the of the others, it is not bounded in its quantity because empirical results prove that it is quite rare, as described further in the analyze of Table 3.8.

Table 3.6 follows the same structure as the precedent ones, having the first two columns dedicated to the instance identification. Next each group of four columns describes the data for the given set of parameters shown in the first row. Columns RS% stands for the normalized RS size – the number of free binary variables inside the RS divided by the quantity of binary variables in the model – columns *Cpx 10h* and *Cpx 1h* are the comparative values with the solution given by Cplex after 10 and 1 hours respectively. Finally, columns Z^* describes how many instances were solved until the optimality inside the proposed RS in one hour or less of CPU time. Also, this table is divided in three parts concerning its rows, each one is dedicated to one set of test instances, having 8, 16 and 32 periods respectively, the 4 periods instances were excluded from this analysis for being too small and not representative. The row presents average results of the 6 instances belonging of each class, all values in Table 3.6 are in [%] except for the ones in column Z^* and the description of the test instances.

		$\lambda = 10, \beta = 5$				$\lambda = 25, \beta = 5$				$\lambda = 50, \beta = 5$			
$\|W\|$	$\|T\|$	RS%	Cpx10h	Cpx1h	Z^*	RS%	Cpx10h	Cpx1h	Z^*	RS%	Cpx10h	Cpx1h	Z^*
200	8	18.47	0.82	0.21	6/6	18.61	0.72	0.11	6/6	16.81	0.71	0.10	6/6
400	8	20.00	1.28	0.53	6/6	17.36	1.19	0.43	6/6	19.86	1.02	0.26	5/6
200	16	18.06	0.70	-3.80	2/6	16.94	-0.03	-4.56	4/6	18.26	0.45	-4.06	3/6
400	16	21.32	0.30	-10.36	2/6	14.38	-1.03	-11.88	4/6	16.46	-0.74	-11.54	4/6
400	32	22.15	1.31	-13.99	0/6	19.24	0.53	-14.90	0/6	18.89	1.09	-14.19	0/6
600	32	15.45	-4.69	-28.95	2/6	17.95	-4.41	-28.65	0/6	17.43	-5.73	-30.41	0/6

Table 3.6: Results for direction vector 1

Taking as comparison the solution provided by Cplex after one hour of CPU time (column *Cpx1h*), we observe that the RS heuristic are able to find solutions around the same quality as *Cpx1h* for the smallest test class. This because, as said before, the exact method performs very well when dealing with small test problems. On the other hand, for the $|T| = 16$ and $|T| = 32$ test classes there are a considerable increase performance of the RS heuristic. Finding solutions up to 28.95% better than the exact method. For some test classes the RS heuristic is better even than Cplex after 10 hours of CPU time.

Concerning the parameters one can notice that the $\lambda = 50, \beta = 5$ setting on average performs better for the 8 and 32 periods instances while the setting $\lambda = 25, \beta = 5$ performs better for the 16 periods instances and these two settings completely dominates the $\lambda = 10, \beta = 5$ setting. As the setting $\lambda = 50, \beta = 5$ gives us the best solutions for 2 out 3 test classes and the size of the RS are very likely for all of them we chose this setting to continue our computational tests.

Direction vector 3

Another important setting in the RS heuristic is when to consider that the two methods in the restriction phase agree on the value of the basic density. This is expressed by the parameter β meaning the upper bound in the Direction Vector module to which we will consider the agreement. In this case any Direction Vector smaller than β will be considered as zero. Considering that the amplitude of a basic density level is $50kg/m^3$, the values $\beta = \{5, 15, 25\}kg/m^3$ where chosen for our test. Following the same structure as the precedent one, Table 3.7 describes the results obtained when we change the value of β. All tests where performed using $\lambda = 50$.

		$\lambda = 50, \beta = 5$				$\lambda = 50, \beta = 15$				$\lambda = 50, \beta = 25$			
W	T	RS	Cpx10h	Cpx1h	Z^*	RS	Cpx10h	Cpx1h	Z^*	RS	Cpx10h	Cpx1h	Z^*
200	8	16.81	0.71	0.10	6	17.22	0.80	0.19	6	2.92	0.94	0.32	6
400	8	19.86	1.02	0.26	5	16.53	1.25	0.49	6	3.89	0.89	0.12	6
200	16	18.26	0.45	-4.06	3	14.03	0.86	-3.63	6	8.40	0.66	-3.84	6
400	16	16.46	-0.74	-11.54	4	13.68	-0.40	-11.18	3	11.60	0.82	-9.78	4
400	32	18.89	1.09	-14.19	0	15.17	1.76	-13.41	0	8.23	1.87	-13.28	1
600	32	17.43	-5.73	-30.41	0	15.00	-5.13	-29.69	0	7.92	-5.10	-29.69	1

Table 3.7: Results for direction vector 3

Analyzing the results provided by Table 3.7 one can notice three general rules: first, the size of the RS decreases when the value of β increases. This can be explained by the *intensification* behavior of this parameter. Because, the greater is its value the greater is the number of (*digester*, *period*) which will be kept in its current value, in consequence the heuristic method will probe a smaller region resulting in a smaller RS. Second, the smaller is the RS the worse is the solution which it contains. Third, smaller RS are more suitable to be solved inside the one hour alloted time.

Examining these results, one can notice that $\beta = 25$ provides on average the smallest RS, in some cases even an RS of size zero and most of these RS are solvable in the 1 hour time limit. As expected for the test instance $|T| = 32$, the quantity of the RS solved inside the 1 hour time limit is smaller than the other classes. Because, despite having a normalized RS value around 8% on average, it implies an RS size around $|T||F||L| * 0.08 = 38.4$ which is outside of the solvable range of $[17, 30]$ described in Section 3.3.

Considering the other two parameter settings $\beta = 5$ and $\beta = 15$ one can notice that there is not a great change in the size of the RS for these two settings, however $\beta = 5$ provides slightly better solutions for most of the test instances.

		$\lambda = 50, \beta = 05$					$\lambda = 50, \beta = 15$					$\lambda = 50, \beta = 25$				
W	T	C	DV1	DV2	DV3	Free	C	DV1	DV2	DV3	Free	C	DV1	DV2	DV3	Free
200	8	2.0	11.5	4.9	5.1	2.5	2.5	6.7	4.1	13.2	0.0	4.0	0.7	3.5	19.9	0.0
400	8	2.0	11.5	5.3	5.0	2.3	2.2	7.7	4.8	11.5	0.0	4.0	0.7	3.4	19.9	0.0
200	16	2.0	24.0	1.7	9.3	13.0	2.0	21.0	1.3	25.3	0.3	2.3	8.8	0.2	39.0	0.0
400	16	2.0	24.0	2.3	10.0	11.7	2.0	18.2	0.5	29.3	0.0	3.2	4.2	0.0	43.7	0.0
200	32	2.0	48.0	3.5	18.5	26.0	2.0	40.3	2.3	52.8	0.0	2.0	19.3	1.0	75.7	0.0
400	32	2.0	48.0	4.3	20.0	23.7	2.0	38.8	2.2	55.0	0.0	2.0	18.0	0.8	77.2	0.0

Table 3.8: Components of the direction vectors

In order to completely understand these tests Table 3.8 presents the quantity of each Direction Vector found in each test. This table has the same structure as the others, with the description of the instance on the left and the next three blocks are the results found. Columns C are the number of cycles L-Method $\leftrightarrow$ NL-Method performed until the stop criteria in the restriction phase, columns DV1, DV2 and DV3 are the average number of each Direction Vector found per cycle. The last column named 'Free" are the quantity of $(digester, period)$ which does not have a direction vector of either type associated with. For example in the first row, we have 24 possible combinations of digester and period (3 factories and 8 periods), from these we have 11.5 $(digester, period)$ associated with a DV1, 4.9 with DV2 and 5.1 with DV3, leaving 2.5 $(digester, period)$ free. These free $(digester, period)$ are very important for the performance of the heuristic, because they leave some space for the method adapt itself avoiding a completely guided solution procedure.

As observed in Table 3.7 the setting $\lambda = 50, \beta = 5$ provides solutions slightly better than $\lambda = 50, \beta = 15$ using an RS of equivalent size. Actually, this can be explained by comparing the column "Free" in Table 3.8 for both settings. In setting $\lambda = 50, \beta = 15$ one can observe that these values are on the great majority zero, meaning that there is not enough space to the method adapt itself. On the other hand, the same column for the setting $\lambda = 50, \beta = 5$ have all the values different from zero. Actually, these free spaces are another factor responsible for the diversification of the search leading to better RS.

A good setting of parameters will respect the tradeoff between intensification and diversification and will consider the total available time to solve the final RS as well as the size of the problem. The value of $\lambda = 50$ seems to be a good compromising in terms of diversification, providing near-optimal solutions for all values of β tested. The setting of this second parameter is dependent of the goal, finding a good solution or finding an RS which may be solvable inside the alloted time. If we are interested in finding an RS which is solvable at optimality large values of β are advised, on the other hand, if we prime for a good solution rather than time a value of β between 5 and 15 are advised.

3.6 Conclusion

In this work we proposed a hybrid heuristic method based on the synergy of its components based on the exchange of information to deal with a real world problem, the Pulp Production Scheduling Problem (PPSP). The proposed solution method can be divided into two phases. In the first, two heuristic methods interact exchanging information about the searched space and visited solutions in order to find a suitable Restricted Space (RS), which is explored by an exact algorithm in the second phase. There are two main objectives pursued in this work, find an RS which may be small enough to be solved in a reasonable time and which contains near-optimum solutions. It is clear that these objective are contradictory, when we completely satisfy one we penalize the other. Despite that, the computational tests showed that it is possible to find a combination of parameters providing a good compromise between a small enough RS and near-optimal solutions. The computational tests performed also showed efficiency of the RS Heuristic, finding the optimal solution in 5 of the 6 instances where it is known.

Acknowledgements

Financial support for this research was provided by the Canadian Natural Sciences and Engineering Research Council (NSERC) and Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES) - Brazil. This support is gratefully acknowledged.

References

Bertels, S. and Fahle, T. (2006). A hybrid setup for a hybrid scenario: combining heuristics for the home health care problem. *Computers & Operations Research*, 33(10):2866–2890.

Biermann, C. J. (1996). *Handbook of Pulping and Papermaking*. Academic Press, second edition.

Correa, A. I., Langevin, A., and Rousseau, L. M. (2004). Dispatching and conflict-free routing of automated guided vehicles: A hybrid approach combining constraint programming and mixed integer programming. In *Integration of Ai and or Techniques in Constraint Programming for Combinatorial Optimization Problems*, volume 3011 of *Lecture Notes in Computer Science*, pages 370–379. Springer-Verlag Berlin, Berlin.

da Silva, C. G., Figueira, J., and Climaco, J. (2007). Integrating partial optimization with scatter search for solving bi-criteria 0,1-knapsack problems. *European Journal of Operational Research*, 177(3):1656–1677.

Fisher, W. D. (1958). On grouping for maximum homogeneity. *Journal of American Statistical Association*, 53(284):789–798.

Foelkel, C. E. B., Mora, E., and Menochelli, S. (1992). Densidade básica: sua verdadeira utilidade como índice de qualidade da madeira de eucalipto para produção de celulose. *O Papel*, 53:35–40.

Gallardo, J., Cotta, C., and Fernandez, A. (2007). On the hybridization of memetic algorithms with branch-and-bound techniques. *IEEE Transactions On Systems Man and Cybernetics Part B-Cybernetics*, 37-1:77–83.

Gronkvist, M. (2006). Accelerating column generation for aircraft scheduling using constraint propagation. *Computers & Operations Research*, 33(10):2918–2934.

Hooker, J. N. (2006). An integrated method for planning and scheduling to minimize tardiness. *Constraints*, 11(2-3):139–157.

Karlsson, J., Ronnqvist, M., and Bergstrom, J. (2003). Short-term harvest planning including scheduling of harvest crews. *International Transactions in Operational Research*, 10:413–431.

Kennedy, J. F., Philips, G., and Williams, P., editors (1989). *Wood processing and utilization*. Ellis Horwood Limited.

Nguyen, H. D., Yoshihara, I., Yamamori, K., and Yasunaga, M. (2007). Implementation of an effective hybrid ga for large-scale traveling salesman problems. *IEEE Transactions on Systems, Man and Cybernetics, Part B*, 37(1):92–99.

Pécora, J. E. (2002). Alocação de médicos em salas de emergências - uma abordagem híbrida. Master's thesis, Campinas University, Campinas, Brasil.

Pécora, J. E., Ruiz, A., and Soriano, P. (2007). Minimization of the wood density variation in pulp and paper production. *INFOR*, 45(4):187–196.

Pendharkar, P. C. (2005). Hybrid approaches for classification under information acquisition cost constraint. *Decision Support Systems*, 41(1):228–241.

Peng, J., Shang, G., and Liu, H. (2006). A hybrid intelligent algorithm for vehicle routing models with fuzzy travel times. *Computational Intelligence, Pt 2, Proceedings*, 4114:965–976.

Puchinger, J. and Raidl, G. R. (2005). Combining metaheuristics and exact algorithms in combinatorial optimization: A survey and classification. In *Artificial Intelligence and Knowledge Engineering Applications: a Bioinspired Approach, Pt 2, Proceedings*, volume 3562 of *Lecture Notes in Computer Science*, pages 41–53. Springer-Verlag, Berlin.

Resende, M. G. C. and Werneck, R. F. (2004). A hybrid heuristic for the p-median problem. *Journal of Heuristics*, 10(1):59–88.

Ronnqvist, M. (2003). Optimization in forestry. *Mathematical Programming*, 97(1-2):267–284.

Sahoo, B. and Maity, D. (2007). amage assessment of structures using hybrid neuro-genetic algorithm. *Applied Soft Computing*, 7-1:89–104.

Schlottmann, F. and Seese, D. (2004). A hybrid heuristic approach to discrete multi-objective optimization of credit portfolios. *Computational Statistics & Data Analysis*, 47(2):373–399.

Talbi, E. G. (2002). A taxonomy of hybrid metaheuristics. *Journal of Heuristics*, 8(5):541–564.

Williams, M. F. (1994). Matching wood fiber characteristics to pulp and paper processes and products. *Tappi Journal*, 77(3):227–233.

Yan, S. and Zhou, K. (2006). Three-tier multi-agent approach for solving traveling salesman problem. *PRICAI 2006: Trends In Artificial Intelligence, Proceedings*, 4099:813–817.

Part III

Application to a Location-Allocation Problem

"If in other sciences we should arrive at certainty without doubt and truth without error, it behooves us to place the foundations of knowledge in mathematics."

Roger Bacon (1214-1294)

Introduction

The main contribution of this part is the proposal of a solution procedure which complements the concepts developed in the Part II. The solution procedure proposed to tackle the PPSP has basically two drawbacks: 1) The size of the RS cannot be precisely controlled and 2) It searches only one RS. The solving method proposed here counts with an incremental search procedure – named embedded RSS – to intelligently explore large RS. It counts also with a binary tree to structure the exploration of several RS. This new iterative method – named Iterative Restricted Space Search, IRSS – has its foundations on the intensification / diversification paradigm, where the intensification is performed by the search phase and the tree structure where it is implemented provides a natural diversification. This diversification is due to the fact that, by construction, the intersection of any two RS is always the empty set. Leading to the exploration of new regions of the solution space at each iteration.

The IRSS is applied to a real-life Location-Allocation problem in the context of an international transportation company. The two decision levels present in this problem makes it a very good choice to illustrate the IRSS, also the non-linearities preset in economies of scale make it a very difficult problem where exact methods can only solve a small size instances. We used the model provided in Cavallet et al. (2000) to be able to compare the results provided by the IRSS with the prior results. In this chapter only a compact version of this model is presented, the reader can find the original model in Appendix A.

Chapter 4

Iterative Restricted Space Search to solve a real-life location-allocation problem

Résumé - Nous traitons dans ce travail d'un cas réel de localisation-affectation rencontré par une société de transport suisse. Le problème intègre des coûts linéaires et non-linéaires, et exploite le transport inter-hub avec l'objectif de profiter d'économies d'échelle. Si des algorithmes exacts ont été utilisés pour résoudre ce problème, leur efficacité se heurte à la taille des problèmes réels. Nous proposons donc un algorithme hybride reposant sur la synergie et l'échange multi-directionnel d'informations. Il s'articule autour de deux phases : une phase de recherche itérative exploitant la structure des coûts et des flux pour délimiter un espace restreint de recherche, et une phase de résolution exacte sur cet espace. La solution optimale générée par la phase exacte sur l'espace restreint de recherche est alors utilisée comme solution initiale par la première phase afin de délimiter un nouvel espace restreint de recherche. Une structure de branchement exploitant l'interaction de ces deux phases permet d'autre part d'interdire toute nouvelle recherche sur l'espace déjà exploré. Cet algorithme a fait l'objet de résultats encourageants montrant sa capacité à restreindre fortement l'espace de recherche permettant de générer plus rapidement des solutions comparables à celles obtenues par des méthodes exactes.

Iterative Restricted Space Search to solve a real-life location-allocation problem

José Eduardo Pécora Jr $^{\dagger\bigtriangledown}$, Angel B. Ruiz$^{\dagger\bigtriangledown}$, Patrick Soriano$^{\ddagger\bigtriangledown}$

†Dép. opérations et systémes de décision, Faculté des sciences de l'administration, Université Laval Québec (Qc) Canada G1K 7P4
eduardo.pecora@centor.ulaval.ca, angel.ruiz@ulaval.ca

‡Service des Méthodes quantitatives de gestion, HEC-Montréal
3000, ch. Côte-Sainte-Catherine , Montréal (Qc) H3T 2A7, Canada
patrick@crt.umontreal.ca

$^{\bigtriangledown}$CIRRELT - Centre interuniversitaire de recherche sur les reseaux d'entreprise, la logistique et le transport, Université de Montréal
CP 6128 Succursale Centre-ville, Montréal (Qc) H3C 3J7 Canada

Abstract This paper proposes a hybrid exact-heuristic algorithm, called Iterative Restricted Space Search (IRSS), for solving an uncapacitated location-allocation problem with non-linear costs, inspired by the case of an international transportation company based in Switzerland. Exact algorithms have been applied to this problem with only partial success, especially for real world problems. The strategy of our algorithm is to iteratively define and explore restricted regions of the global solution space that have a high potential of containing good (hopefully, optimal) solutions. The algorithm encompasses two phases: the restriction phase, which defines a restricted region in the solution space (the RS); and the search phase, during which this RS is explored thoroughly. The best solution found inside the RS is passed back to the restriction phase, which defines a distinct new RS. The algorithm alternates between the two phases a fixed number of times or until the allotted time is expired. The interaction between the two phases is supported by a branching structure that prevents duplicate searches. IRSS produces very positive computational results in a reasonable amount of time.

4.1 Introduction

The location-allocation problem described in this document is based on the real-world context of a Swiss transportation company, which operates an import/export network with its own consolidation centers, or hubs, located inside and outside Switzerland. Commodities to be exported from national customers are transported to a national hub and from there to an international hub; imported goods follow the inverse of this path. This multi-commodity uncapacitated location-allocation problem takes into account two sets of non-linear costs: the hub implementation cost and the transportation cost for the goods traveling from or to a hub located outside Switzerland. These non-linear costs are used to represent the economies of scale possible from grouped transportation. Solving this problem involves, first, deciding which national hubs should be opened and their size, and second, selecting the path of each commodity through the network. The international hubs already exist and are thus outside of the scope of our study. Figure 4.1 depicts the existing hub network.

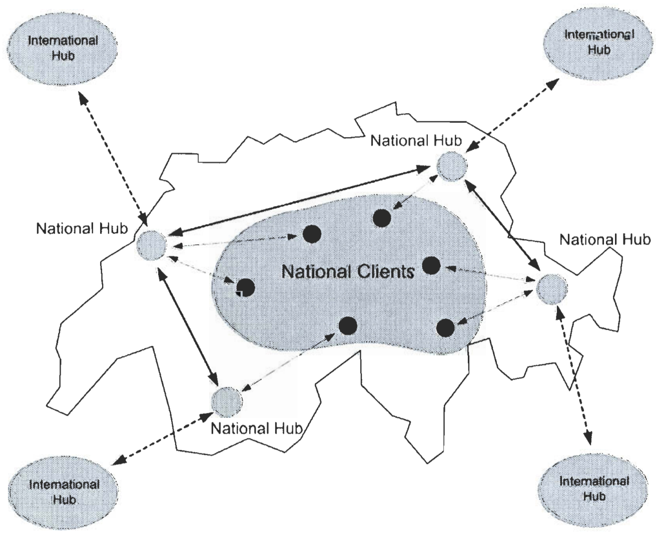

Figure 4.1: The transportation network

Due to the size and the mathematical complexity of real-world problems, pure exact algorithms are inefficient for solving real-world instances. For this reason, this paper proposes the Iterative Restricted Space Search (IRSS), a hybrid exact-heuristic algorithm that extends the RS Heuristic proposed by Pécora et al. (2007).

The IRSS hybrid algorithm has two phases: the restriction phase, which exploits the non-linear cost structures of hub allocation and flows to the external arcs in order to define a Restricted Space (RS); and the search phase, during which this RS is thoroughly explored. The IRSS alternates between them in order to generate and then explore several distinct RS. An RS is a subset of the solution space that is small enough to be explored thoroughly in a reasonable amount of time but large enough to maximize the probability of finding good (hopefully, optimal) solutions. The best solutions found inside the RS and its complementary search space, RS^C, are sent back to the restriction phase in order to define a new RS. The algorithm stops after exploring a given number of RS or when the allotted computational time is expired.

The remainder of this paper is structured as follows. The next section (4.2) reviews the recent advances in hybrid algorithms applied to discrete location problems. Section 4.3 describes the mathematical model used in this study and explains how the non-linear cost structures were modeled. Section 4.4 introduces the hybrid algorithm and its components, as well as the protocol of the information exchange between them. Section 4.5 presents our computational experiments and their results, and section 4.6 offers our conclusions.

4.2 Recent works in discrete location problems

This section focuses on recent papers about discrete location problems, especially those studies that have used hybridization as a solution method. Since the literature about hybrid algorithms is quite extensive, this review focuses on hybrid methods based on heuristics or integer programming, though a few surveys about general location problems are also mentioned for the interested reader.

Several annotated bibliographies and reviews have been published in the field. ReVelle et al. (2008) have recently published an updated bibliography of the recent advances in discrete location problems and their variants. Klose and Drexl (2005) published another bibliography, in which the location problems are grouped in three main categories according to the modeling approach used: continuous, network and MIP. ReVelle and Eiselt (2005) also published a survey of location analysis studies, which classifies the problems based on type (e.g., discrete and continuous allocation) and objective function (e.g., "Pull" functions to define depot locations, "push" functions to locate landfills, facilities that most of the people would like to have far from their homes, and "equity" functions to insure that the distances between customers and facilities are as equal as possible.) Marin et al. (2006) considered the possible formulations of the uncapacitated multi-hub location allocation problem, proposing a formulation with inter-hub transportation for up to two hubs that outperforms the previous formulations for small and medium problems. The authors show that clique-based constraints are tighter than the classic constraints and also discuss preprocessing and variable elimination.

The following paragraphs focus on the different solution approaches proposed in the literature, specifically the hybrid methods used for both uncapacitated and capacitated discrete location problems. The review includes mostly heuristic-heuristic hybrids, but also some hybrids employing Lagrangean relaxation, subgradient methods and pricing/column generation. Uncapacitated problems have been addressed by Abdinnour-Helm (1998), Lapierre et al. (2004), Diaz and Fernandez (2005), and Resende and Werneck (2006) among others. Abdinnour-Helm (1998) used a hybrid Genetic Algorithm (GA) and Tabu Search (TS) to tackle the uncapacitated hub location problem. In this hybrid, the best solution provided by the GA after a fixed number of generations is passed to the TS, which performs a predefined number of iterations in a multi-start fashion. Lapierre et al. (2004) used a Tabu Search-Variable Neighborhood Search (VNS) hybrid to solve a location-allocation problem. The complex objective function of the problem in the study required choosing among several transportation alternatives (i.e., LTL, FTL, Parcel or own fleet), which made the problem unsolvable with exact methods. The neighborhood developed for this application uses a pseudo-random sampling technique to reduce the number of solutions to be evaluated but retains the possibility of concentrating the shipment in a specific hub. Their hybrid algorithm is essentially a Tabu Search, in which the VNS chooses the neighborhood to be explored based on the search status. Resende and Werneck (2006) proposed a two-phase hybrid procedure for the uncapacitated facility location problem. Their algorithm starts by randomly generating a set of feasible solutions S and running a local search on each of them. Then, the solutions in S are combined with the solutions from a 'best solutions' pool using path relinking. The resulting solutions may integrate the pool depending on their "quality", measured in terms of the objective function, but also other factors. During the second phase of the procedure, solutions in the pool are recombined using the same path-relinking approach. The authors called this algorithm "hybrid" because it combines elements from several metaheuristics, such as scatter and tabu search, path relinking and genetic algorithms.

Capacitated problems have also received attention in the literature. Diaz and Fernandez (2005) compared several combinations of the metaheuristics GRASP (Feo et al., 1994), path relinking (PR) and scatter search (SS) (Glover et al., 2000) for the capacitated p-median problem. In their approach, the GRASP heuristic is used to generate an initial solution for the hybrids GRASP-SS, GRASP-PR and GRASP-PR-SS, which are implemented sequentially without feedback. They concluded that the GRASP-PR-SS combination was the best since it produced solutions whose quality was similar to GRASP-SS but used less computational effort. Other SS-PR hybrids have been proposed both by Perez et al. (2005) for the capacitated p-hub median problem and Keskin and Uster (2007) for the capacitated location problem. Wu et al. (2006) proposed an algorithm based on decomposition and Lagrangean relaxation for a capacitated facility location problem with two types of costs: the setup costs associated to the opening of a facility and a non-linear cost based on the number and the volume of clients allocated to this facility. This algorithm is encapsulated inside a sub-gradient method, which is used to update the Lagrangean multipliers. The Lagrangean relaxation provides a lower bound for the problem and the set of the facilities to be opened. The resulting problem containing only the transportation variables is solved with a linear method, which provides a complete solution and the upper bound for the problem. Doong et al. (2007) has recently applied a hybrid algorithm combining a genetic algorithm (GA) and a subgradient method to solve a single-source capacitated facility location problem. They decomposed the variables into location variables, which are processed by the GA, and allocation variables, which are dealt with by the subgradient method. This algorithm has a hierarchical structure, with the GA occupying the main position and the subgradient method relegated to the bottom layer. After each GA recombination, the location variables are fixed, and the subgradient method is used to find the best values for the allocation variables. The same problem has been tackled by Delmaire et al. (1999) using a hybrid algorithm based on a GRASP in which the local search is done by a Tabu Search. Delmaire proposed a variant of this hybrid using a Reactive GRASP structure. The authors concluded that, although the former performs slightly better, the latter seems to be more robust.

A parallel hybrid heuristic for the multi-commodity capacitated location problem was developed by Gendron et al. (2003). This hybrid, which combines Variable Neighborhood Descent (VND) (Hansen and Mladenović, 2001) and Slope Scaling (Kim and Pardalos, 2000), is based on adaptive memories and is implemented in a master-slave structure, in which the master process manages the memories and the slave processes (VND and Slope Scaling) perform the computations. Velarde and Laguna (2004) proposed a hybrid combining a linear method and a tabu search for solving the capacitated international sourcing problem. The study introduced a scenario-based approach that evaluates potential facility insertions and/or deletions, with swaps being evaluated as an insert/delete combination. Each scenario is solved by the linear method, and the obtained shadow-prices are weighted according to the likelihood of the scenario. Then, the movements (i.e., insertions, deletions, and swaps) are classified and sorted into a candidate list and fully evaluated by the objective function. Finally, the best movement is implemented, and the algorithm restarts with this new solution. Lorena and Senne (2004) developed a column generation approach for solving a capacitated p-median problem. Their method uses a Lagrangean/surrogate relaxation to identify new productive columns, thus accelerating the computational process.

Carrano et al. (2005) examined a real-world location problem: the location of energy substations and the subsequent energy distribution. These authors decomposed the problem into two sub-problems – a substation location using a quasi-Newton algorithm and an energy distribution network topology using a genetic algorithm – and implemented these methods in an intertwined framework.

Cavallet et al. (2000) were the first to address the problem of the Swiss company examined in this paper. They provided the mathematical model and preliminary computational results for several test instances. Their approach consisted of solving the linear relaxation of the MIP model iteratively, with new valid cuts being added at each iteration. Although our work is based on their formulation, we do not consider the valid cuts in this paper because generating them requires a very time consuming secondary knapsack model, and the method becomes inefficient for solving large real-world instances. The next section presents the mathematical formulation of the problem examined in this paper.

4.3 Mathematical formulation

This section describes the decision variables, sets, parameters and constraints of the mathematical formulation. As the contribution of this paper is not the modeling, but the design of an efficient solution approach to the problem studied, a compact version of the formulation detailing exclusively the objective function is presented. However, the complete formulation as proposed in Cavallet et al. (2000) is available in Appendix A.

The network considered here includes internal nodes $i \in I$ (national clients), hubs $h \in H$ (national freight terminals) and external customers $e \in E$ (international freight terminals). An export commodity travels from an internal customer to an external client and is defined by the ordered pair $(i, e) \in J^1$. Similarly, an import commodity travels from an international client to a national client and is denoted by the ordered pair $(e, i) \in J^2$. Thus, the set $J = J^1 \cup J^2$ includes all the possible commodities. The demands, denoted $d_{i,e}$ and $d_{e,i}$ respectively, are deterministic and known in advance. All the arcs in the network are uncapacitated. Transportation costs between hubs and the international customers present economies of scale with respect to the transported volume. Also, the capacity of each hub must be selected among several options, with the hub implementation cost depending non-linearly on the capacity selected. These non-linear costs are modeled in the following by piecewise linear functions.

Variables

Four distinct groups of variables are proposed. Transportation variables, hub location variables and piecewise variables (which are used to define the non-linear cost functions) are boolean. Continuous flow variables are associated to the arcs in the network but also to the hubs. Transportation variables represent the path traveled by each commodity. Export commodities are denoted by (x^E), while (x^I) refers to import commodities. Thus, x^E_{ihe} takes the value 1 if the export commodity travels from internal customer i to international customer e through hub h and the value 0 otherwise. Hub location variables $y_h \in \mathbb{B}$, take the value 1 if the hub h is active (open) and zero otherwise. Following the same convention as the one proposed for the transportation variables, The total flow through each hub and arc are represented by $w_h \in \mathbb{R}^+$, $w^E_{eh} \in \mathbb{R}^+$, and $w^I_{eh} \in \mathbb{R}^+$, respectively. Note that the model can be also formulated without flow variables. Indeed, these variables are nothing but the sum of the products of transportation variables multiplied by the demands. However, they are used here to facilitate the understanding of the reader. Note also that flow variables and piecewise variables are strongly related to represent the mentioned economies of scale in costs.

Mathematical model

$$
\begin{aligned}
\text{Minimize Z} \quad = \quad & \sum_{ihe} \theta_{ih}(d_{ei} * x^I_{ehi} + d_{ie} * x^E_{ihe}) + \\
& + \sum_{h} f_h(w_h) + \\
& + \sum_{h,e} [f^I_{eh}(w^I_{eh}) + f^E_{eh}(w^E_{eh})] && (4.1) \\
& Ax <= b && (4.2) \\
& Bw <= r && (4.3) \\
& x^I_{ihe}, x^E_{ehi}, y_h \in \mathbb{B}; w_h, w^I_h, w^E_h \in \mathbb{R}^+ && (4.4)
\end{aligned}
$$

The objective function (4.1) includes four terms. The first term computes the linear transportation costs, θ_{ih}, associated to the collection and distribution between internal clients i and the hubs h. The second term corresponds to the cost of opening and operating hubs. Note that $f_h(w_h)$, the cost associated to hub h, depends on the total flow through h in a non-linear manner. The third term refers to the transportation costs, $f^I_{eh}(w^I_{eh})$, $f^E_{eh}(w^E_{eh}$, between the hubs and the international clients in both directions. Again, the cost associated to each arc depends of the flow transported in a non-linear manner. The model is completed with Equations 4.2 which refer to classical constraints of path unicity and transportation through open hubs only, and Equations 4.3, a set of equations used to link the flow to the piecewise cost structures. Equations 4.2 and 4.3 are explicitly formulated in Appendix A. Finally, Equations 4.4 define the variables domain.

4.4 Iterative Restricted Space Search - IRSS

The Iterative Restricted Space Search (IRSS) is a heuristic algorithm that extends the research of Pécora et al. (2007). Like the original algorithm, IRSS is divided into two phases: the restriction phase, in which an interesting region, called RS, is defined; and the search phase, in which this RS is thoroughly explored. However, IRSS includes two new and important features that resolve the major weakness of the original algorithm. In their implementation, Pécora et al. (2007) used two heuristic methods to probe the search space with the goal of defining the RS, which is explored by a branch-and-cut algorithm at the end of the procedure. Their approach, despite finding near-optimal solutions for most of the test instances proposed, has two main drawbacks. First, the "size" of the RS cannot be precisely controlled, which may lead to RS that are too large to be solved optimally in the allotted time, or to RS that are too small, thus reducing the probability that they will contain optimal solutions. Second, the Pécora's method generates a single RS and does not include any feature that could help improve the RS if the RS chosen is not as good as expected. IRSS remedies these drawbacks by proposing (1) a enhanced restriction phase in which the size of the generated RS can be explored progressively through a procedure called Embedded RS, and (2) a backtrack procedure with a binary tree structure that uses the probing and searching phases in a recursive manner, each time generating a new RS based on the information gathered so far. The next sub-sections describe the two phases and the components of the IRSS approach in detail.

Restriction phase

Since the main goal of this phase is to probe the solution space, a local search does not seem the most suitable approach. Instead, we chose to encourage a wider search covering a larger part of the solution space. To this end, we concentrate our efforts on the location variables (i.e., the opening of hubs and the choice of their size) that characterize the solutions; once these variables have been set, the solution is not very sensitive to changes in the allocation variables. Dealing only with location variables allows our algorithm to explore a larger portion of the solution space. To this end, the restriction phase alternates between two methods called DSSPh and DSSPa. DSSPh aims at proposing a solution structure by choosing the amount of flow through the hubs, while DSSPa completes this structure by adding the allocation variables to the partial solution correcting, if necessary, the location/hub size variables. These two methods alternate for a fixed number of times. Because adding the allocation variables and verifying how the location variables have been adjusted is a source of information that can be used when defining an interesting search region, these additions and corrections are very important. Therefore, from one iteration to the next, the corrections are analyzed and re-introduced into the first method. The following subsections present the methods used in the restriction phase and their application, explain how the information extracted from the location variables adjustments is used when alternating between these methods, and finally, describe how the RS is generated.

Dynamic slope scaling methods

The location-allocation problem considered in this paper has a very particular cost structure that was modeled using two sets of piecewise linear functions (see Appendix A). The restriction phase deals with each of these piecewise cost structures using an adapted version of the Dynamic Slope Scaling and trust intervals (DSSP) search procedure proposed by Kim and Pardalos (2000): DSSPh for dealing with hub costs and DSSPa for dealing with external arc costs. The main idea of DSSP is to approximate the piecewise function using a linear function (i.e. $f(w) \approx \alpha * w$), with each iteration of the method making this approximation more realistic. Figures 2 and 3 depict a piecewise linear function with four continuous pieces. For the given flows ($w > 0$ and $w = 0$, respectively), the algorithm approximates these functions using a pure linear function, represented in the figures by a dashed line. Although the piecewise cost functions are modeled with sets of binary variables, the DSSP approach focuses exclusively on the value of flow w through the hubs and arcs. These flows clearly define a unique piece in the piecewise cost function, which, in the example shown in Figure 4.2, corresponds to active variables z1, z2 and z3 for $w > 0$ and to no variables for $w = 0$ (Figure 4.3).

The restriction phase uses DSSPh and DSSPa to generate an RS. Roughly, DSSPh works as follows. An initial approximation to each piecewise cost function is done. In our case, the approximation suggested in Figure 4.3 was elected arbitrary. Therefore, the cost functions in terms two and three of Equation (4.1) become simply the product of a constant (the slope of the cost approximation) multiplied by the corresponding flow variable. The formulation become a pure linear model and it is solved by *Cplex* efficiently, producing a solution s_i. Based on the flow through each hub and arc in

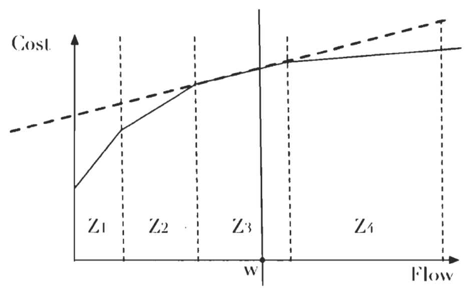

Figure 4.2: Linear approximation, flow > 0

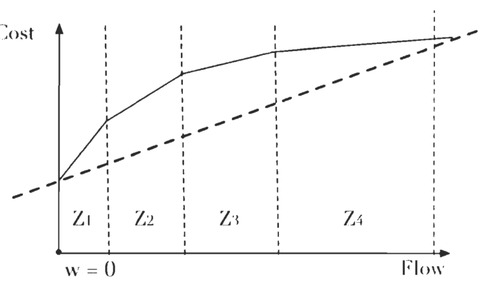

Figure 4.3: Linear approximation, flow $= 0$

s_i, the slope of the piecewise functions for hubs and international arcs are identified. If the hub flows in s_i lead to the same cost approximations as the ones originally proposed, then the algorithm has converged and s_i provides optimal flows. If not, the new cost approximations are proposed based on the flows in s_i. These approximations replace the cost functions in Equation (4.1) and the new formulation is solved. The procedure continues until convergence is reached or a given number of iterations is attained. Note that the procedure compares exclusively the flow through hubs between two consecutive approximations. Also, it is worth to mention that, although proof of convergence is mentioned by Kim and Pardalos (2000), the IRSS algorithm does not require optimality in this phase. The hub sizes and cost approximations in the final solution of the procedure, s_h, are retained and passed to the next search procedure, DSSPa (i.e., the DSSP for the arcs).

DSSPa updates the international arc costs based on the hub structure in s_h. In particular, active hubs and their size are used to bound the maximum and minimum flow through each hub so that the approximations to the piecewise linear functions hold. These constrains are added to the mathematical formulation as well as the approximations to the arc costs, and the resulting linear program is solved. If the flows in s'_h – the solution to this new linear program – lead to the same arc cost approximation, the procedure stops. If not, the search procedure continues until convergence is reached or a given number of iterations is attained. The final solution of the procedure, s_a, is then compared to s_h in order to identify the direction vectors (see Pécora et al., 2007) as explained in the next sub-section. Table 4.1 synthesizes the main characteristics of each DSSP.

Information flow within the restriction phase

Alternating between DSSPh and DSSPa should progressively lead to fixing some intervals, (i.e., a range of location variables values) that can be trusted. In order for this

	Information passed	**Hub Flow**	**Arc Flow**	**Hub Costs**	**Arc Costs**
DSSPh	cost structure + hub flow bounds	free	free	dynamically updated	fixed
DSSPa	cost structure	bounded in the current piece	free	fixed	dynamically updated

Table 4.1: DSSPs

process to be better understood, let us review the interactions of the two DSSP described above, in terms of hub flows. For each hub, DSSPh produces a given value w_h, which is bounded by $\underline{w}_h \leq w_h \leq \bar{w}_h$, the closest flow breakpoints in the cost piecewise function. Then, DSSPa adjusts the values of the arc flow variables. Consequently, the hub flow values may change, with the new value being denoted, w'_h. The difference, $\Delta = w'_h - w_h$, is called the direction vector because it points to a possible profitable change in the hub flow. Intuitively, if Δ is close to zero, the two procedures (DSSPh and DSSPa) agree on the value accorded to the variable, and thus it is reasonable to decide to retain the value of the flow w_h. However, when the two procedures disagree on the value of w_h, this disagreement needs to be taken into consideration during the following DSSPh/DSSPa iteration. A special case may arise when w'_h reaches one of the bounds of w_h (i.e., $w'_h = \underline{w}_h$ or $w'_h = \bar{w}_h$), illustrated in Figure 4.4. In this case, neighbor level of the piecewise function should have been chosen, but given the constraints to the flow, it was not possible to change the capacity level. This may be interpreted as the disagreement in the capacity level to be chosen by the two method belonging to the restriction phase. In this situation a deeper exploration should be performed to solve this issue.

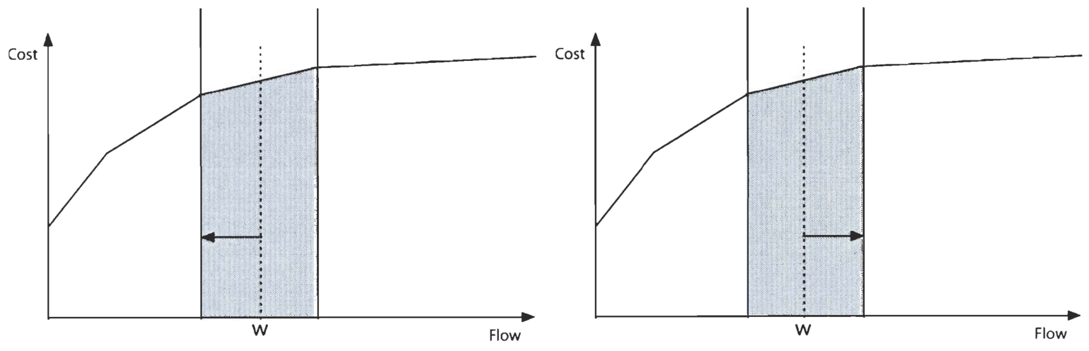

Figure 4.4: Direction vector

At the end of each DSSPh - DSSPa iteration, the information extracted from the direction vectors is considered and translated into flow constraints for the following iteration. However, much care is needed when introducing such flow constraints into the model because blocking several pieces of the cost function at a time may have a major impact on the solution structure, possibly even making the solution infeasible. Therefore, we decided to use one single direction vector per iteration. Direction vectors whose w'_h is equal to one of the flow bounds are preferred, particularly the one with the largest absolute value. If none of the w'_h is equal to one of the bounds, then the largest $|\Delta|$ is selected.

Restricted Space - RS

The main idea of the IRSS is to identify a part of the whole solution space that has a high likelihood of containing good solutions (hopefully, the optimal solution), but that is also small enough to allow a thorough exploration in reasonable time. Intuitively, RS is a sort of convex envelope for the solutions visited during the restriction phase. Let $s^i \in S$ be the set of n solutions found by DSSPa after n iterations of the restriction phase, and $z^i_{h,l}$ be the value of the binary variables associated to the flow through hub h in the particular solution $s^i \in S$. Let J be defined as the set of z (location) variables set to 0 and K be the set of z_s set to 1, which are formally defined as $J = \{(h,l) | \sum_i (1 - z^i_{h,l}) \geq |S|\}$ and $K = \{(h,l) | \sum_i z^i_{h,l} \geq |S|\}$. Therefore, the constraints for generating an RS can be expressed as:

$$\sum_{j \in J} z_j + \sum_{k \in K} (1 - z_k) = 0 \tag{4.5}$$

The size of RS, $|RS|$, is defined as the number of non-fixed variables inside the RS, $|RS| = |H| \text{x} |L| - |J| - |K|$. Thus, the RS can also be seen as the union of several solutions. Unfortunately, this definition of sets J and K may lead to very large RS that cannot be explored thoroughly in a reasonable time. Moreover, an unfortunate choice in the variables of sets J and K could lead to an uninteresting RS. To overcome these drawbacks, the IRSS approach has been enhanced with (1) a decomposition approach called the Embedded RS that allows a "progressive" exploration of the RS, and (2) a binary tree structure that allows several distinct RS to be generated and visited.

Embedded RS

The embedded RS is an incremental search procedure that exploits the information provided by the solutions in S in order to define a set of partitions in the RS. The idea is to divide the original RS into self-contained subspaces of decreasing size. To this end, the information provided by the restriction phase is used as follows. The RS is defined by the set of variables that has the same value (0 or 1) in all the solutions in S. Thus, a subspace of the RS can be defined by all the variables that retain the same value in at least $\lambda * 100\%$, $\lambda \in (0,1]$, of the solutions in S. Let us call this subspace RS^λ, which is formally defined by the sets $J^\lambda = \{(h,l)| \sum_i (1 - z^i_{h,l}) \geq \lambda |S|\}$ and $K^\lambda = \{(h,l)| \sum_i z^i_{h,l} \geq \lambda |S|\}$. Note that as the value of λ decreases, the number of elements in the sets J^λ and K^λ increases and the quantity of the fixed variables increases, leading to smaller RS. Therefore, the values λ and $|RS|$ are proportional, with the original RS being expressed with $\lambda = 1$. The following relationships can be proved in a straightforward manner:

- $RS^{\lambda_1} \subseteq RS^{\lambda_2}, \forall \lambda_1 < \lambda_2$
- $|RS^{\lambda_1}| \leq |RS^{\lambda_2}|, \forall \lambda_1 < \lambda_2$
- If a given RS^{λ_2} is explored to optimality, this optimal solution is better than or equal to any solution with respect to any embedded RS defined by $\lambda_1 < \lambda_2$.
- If a given RS^{λ_2} is proved unfeasible, then any embedded RS defined by $\lambda_1 < \lambda_2$ is also unfeasible.

Consequently, any set $\{\lambda_1 < \lambda_2 < ... < \lambda_t\}$ defines a set of embedded subspaces $RS^{\lambda_1} \subseteq RS^{\lambda_2} \subseteq ... \subseteq RS^{\lambda_t}$. But this partitioning procedure can be improved by using the information provided by the objective value associated to each of the solutions in S. Intuitively, the solutions with better objective values should be more trustworthy, which explains why, in our implementation of the embedded RS procedure, the information provided by each $s^i \in S$ is weighted by its objective value, $g(s^i)$. The sets J^λ and K^λ are then redefined as:

$$J^\lambda = \{(h,l)|C \textstyle\sum_i (1 - z^i_{h,l})/g(s^i) \geq \lambda |S| \sum_i 1/g(s^i)\}$$

$$K^\lambda = \{(h,l)|C \textstyle\sum_i z^i_{h,l}/g(s^i) \geq \lambda |S| \sum_i 1/g(s^i)\}$$

where C is a constant intended to prevent floating point errors. Please note that, since the present application is a minimization, each solution is weighted by the inverse of its objective function, $1/g(s^i)$. Therefore, the embedded RS procedure allows the RS to be partitioned into subspaces of almost any size, thus avoiding the need to solve a large RS.

Search phase and the binary tree structure

The original RS heuristic proposed by Pécora et al. (2007) generates a single RS, which is then thoroughly explored with an exact method (i.e., a standard branch-and-bound algorithm). As mentioned above, such an exact approach can lead to rather poor results if the RS, despite the meticulousness of its construction, does not contain any near-optimal solutions or if its size does not allow a complete exploration within a reasonable time. Therefore, a backtrack mechanism would help to improve the search by allowing several RS to be explored in succession. However, the way in which this exploration is structured is highly important. We propose a binary partition of the search space, inspired by Local Branching (Fischetti and Lodi, 2003). This structure is shown by Figure 4.5.

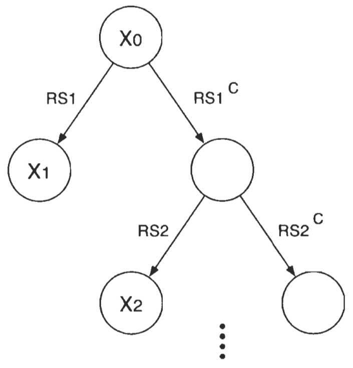

Figure 4.5: Branching structure

Remember that, as mentioned above, the RS is obtained by a restriction as shown in Equation 4.5. Therefore, the complementary space of RS, called RS^C, is defined by the equation:

$$\sum_{j \in J} z_j + \sum_{k \in K} (1 - z_k) \geq 1 \tag{4.6}$$

As can be seen in Figure 4.5, the binary tree starts with an initial solution X0. During the first iteration of the IRSS algorithm, the whole solution space is divided into the first RS, called RS1 (the left branch), whose optimal solution is called X1, and its complementary space, RS1^C (the right branch). The second iteration of the algorithm proposes a new RS, RS2, which is located in the solution space that has not yet been explored (i.e., RS1^C) and whose optimal solution inside RS2 is X2. The right branch at this second level again consists of the solution space that has not been explored so far (e.g., RS1^C/RS2). The next iteration of the algorithm partitions the remaining solution space into two separate subspaces: the (theoretically) most appealing part of the solution space, which will be thoroughly explored by the IRSS algorithm, and the rest. This iterative process continues until a predefined stop criterion (e.g., a number of iterations) is reached.

Exploring the left branch of a particular RS can potentially lead to four types of results: (1) Optimal - proof of local optimality is obtained for a solution Xi; (2) Feasible – a feasible solution is obtained, but there is no proof of optimality; (3) Unknown - no feasible solution is obtained, but the region could not be explored completely within the allotted time; and (4) Unfeasible - the RS does not contain any feasible solution. If the branch is declared Optimal or Unfeasible, the searched RS can be excluded from further iterations. On the other hand, if the branch is declared Feasible or Unknown, this region cannot be excluded without running the risk of missing a local optimal solution. However, when using the embedded RS approach to search the left branches, it is possible to exclude only the part of the RS that has been proven Optimal or Unfeasible. Unlike the Local Branching approach, which requires the left-branch solution to continue the branching sequence, an unfeasible branch is not a major problem for the Embedded RS approach because the IRSS's algorithmic process creates a new left branch based on the restriction phase. Thus, it does not require the solution found at the previous level. Consequently, consecutive RS obtained by IRSS have some interesting characteristics: they are separate because they belong to complementary spaces, and they may be located far away from one another, meaning that no diversification mechanism is required for a robust search.

Obviously, the proposed method can become exact. For this to happen, each right branch, RS, must be completely explored and also a lower bound (upper bound for maximization problems) needs to be computed for each right branch by solving its linear relaxation. Then, if this bound is worse than the best solution found so far, the search stops, and the solution can be declared the global optimum. Despite being able to prove the optimality, the number of branches needed may be huge. Depending mostly of the size of each RS explored at each right branch.

Multi-directional information exchange

One of the main advantages of the IRSS algorithm is that useful information is generated and shared between several search procedures. Figure 4.6 shows the multi-directional information exchange in the IRSS Hybrid Algorithm. The exchange starts with the reduced model passing the initial hub/arc flow information to the iterative DSSP, where the linear costs and the hub flow bounds are exchanged by DSSPh and DSSPa. The iterative DSSP method then sends an RS to the search method, which returns the RS^C, plus a feasible solution (if one exists) and the lower bound for the restriction phase.

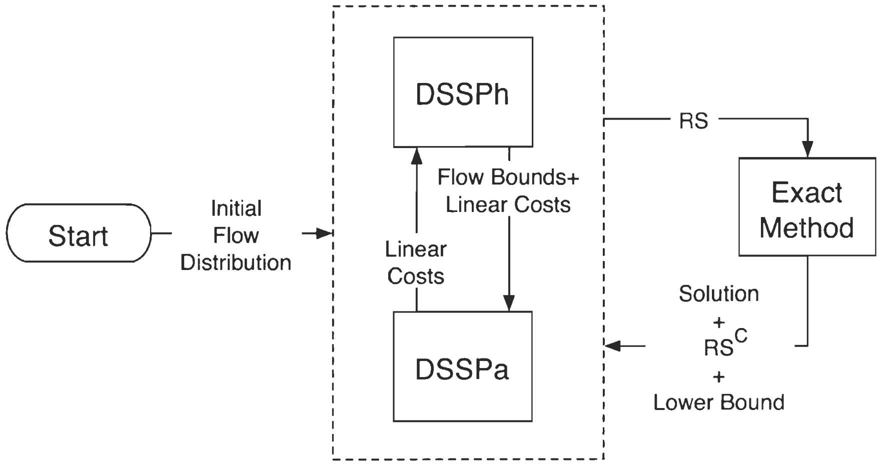

Figure 4.6: Information flow

4.5 Computational experiments

This section presents the computational tests done to validate the hybrid algorithm described above. Preliminary tests were done to set the method parameters – e.g., the number of iterations for each DSSP procedure (h and a) during the restriction phase and the stop criteria of the hybrid algorithm. The parameters used in these tests are presented in Table 4.2.

Restriction Phase	
Number of cycles in DSSPh	2
Number of cycles in DSSPa	3
Iterations DSSPh $\leftrightarrow$ DSSPa before the generation of RS	3

Search Phase	
Maximum allowed time to solve each embedded RS	120 s
Minimum RS size, μ	3
Number of branchings	3
Optimality Gap	0.001

Table 4.2: Parameters setting

Test instances

Five classes of tests were generated to validate the IRSS. These test classes, shown in Table 4.3, basically differ in terms of the number of hubs and the number of internal customers, marked origins in the tables. Each class includes 10 instances; all ten instances have the same cost structure, but they have distinct demands.

	Hubs	Internal Customer	External Hub	Piecewise for Hub	Piecewise for Arc
Class 1	9	10	42	6	4
Class 2	9	50	42	6	4
Class 3	9	100	42	6	4
Class 4	18	50	42	6	8
Class 5	24	50	42	6	8

Table 4.3: Instances description

Computational results

In order to assess the efficiency of IRRS, each instance was run using CPLEX 10.0 in the default configuration (i.e., the heuristic methods were disabled). The solutions provided by the exact solver are used as the basis of comparison throughout this section and are referred to as "the best known solution". The optimality gap was set to 0.001, and the maximum CPU time was set to 15 hours on a dual AMD Opteron 250 2.4 GHz computer with at least 4Gb of RAM and running under a Linux operating system.

Tables 4.4 to 4.8 show the results obtained by the IRSS algorithm. The columns RS1, RS2 and RS3 show the solutions provided by the first, second and third RS, respectively, compared to the CPLEX solution after 15 hours. This comparison was computed using the equation:

$$RS = \frac{Obj_{RS} - Obj_{cplex}}{Obj_{cplex}} \tag{4.7}$$

The column R-Time is the ratio between the time needed by CPLEX to find the best known solution (NB: not the total CPLEX time) and the IRSS algorithm's total time. The last column, IRSS same-t, is the comparison between the solutions obtained in RS3 with the solution provided by CPLEX using exactly the same amount of time as IRSS. The last three rows report the average value of each column (Average) and how many times the RS provided a solution of the same quality (Same) or better (Better) than the best known solution. Since the optimality gap is set to 0.001, solutions that are equal to that figure or that differ by less than 0.1% are considered as equivalent, therefore the solutions in the range $[-0.1\%, 0.1\%]$ are situated in the row, *Same*, and solutions below -0.10% are in the row, *Better*.

Table 4.4 shows the results for smallest test class. In this class, the exact algorithm was able to prove optimality for all the instances in an average of 6 minutes. IRSS found the its best solution, on average 0.36% worst but 2.57 times faster than CPLEX. IRSS also provided solutions that were, on average, 0.21% worse than CPLEX (IRSS same-t), but this was to be expected because the exact algorithm is very efficient for such small test instances. In addition, there was little improvement between the first and last RS, meaning that the first RS is already quite good. There is no value reported for the third test instance because CPLEX was unable to find a solution in the same time as IRSS.

Table 4.5 shows the results for the tests with 9 hubs and 50 origins. There are two instances that can be considered as outliers in this class: instance 1, for which the exact method was able to find a good solution in only a few seconds and the IRSS algorithm was not able to improve it, and instance 5, for which the exact method found a solution in around 100 seconds but was unable to improve it, while IRSS was able to find a solution that was 21.7% better. Excluding these two outliers, the other instances behaved quite similarly, with little or no improvement throughout the three RS, which was to be expected given the size of the problem. The R-Time column shows that IRSS is, on average, 22 times faster than CPLEX, mostly due to its rapidity in solving the three last instances.

Instance	RS1	RS2	RS3	R-Time	IRRS same-t
1	0.56%	0.06%	0.06%	0.26	0.06%
2	0.81%	0.57%	0.57%	0.28	0.57%
3	0.85%	0.85%	0.85%	5.17	-
4	1.81%	0.57%	0.57%	0.72	0.57%
5	0.27%	0.27%	0.27%	0.71	0.27%
6	0.59%	0.55%	0.55%	1.49	0.51%
7	0.26%	0.25%	0.25%	1.06	0.07%
8	0.79%	0.26%	0.22%	1.27	0.19%
9	0.90%	0.36%	0.05%	3.13	-0.57%
10	0.25%	0.25%	0.25%	11.62	0.22%
Average	0.71%	0.40%	0.36%	2.57	0.21%
Same	0 / 10	1 / 10	2 / 10	—	2 / 10
Better	0 / 10	0 / 10	0 / 10	—	2 / 10

Table 4.4: Class 1 computational results - 9 Hubs, 10 Origins

Instance	RS1	RS2	RS3	R-Time	IRRS same-t
1	0.98%	0.65%	0.65%	0.04	0.65%
2	-0.05%	-0.10%	-0.12%	7.64	-0.40%
3	0.19%	0.19%	0.19%	2.88	-0.40%
4	0.87%	0.52%	0.52%	11.49	-0.11%
5	-21.70%	-21.70%	-21.70%	0.01	-21.70%
6	0.99%	0.63%	0.61%	2.42	0.09%
7	0.21%	0.21%	0.21%	14.46	-0.32%
8	0.02%	0.02%	-0.04%	31.37	-0.34%
9	0.28%	0.28%	0.28%	89.10	-0.01%
10	0.24%	0.24%	0.24%	62.85	-0.43%
Average	-1.80%	-1.91%	-1.91%	22.23	-2.297%
Same	1 / 10	2 / 10	1 / 10	—	2 / 10
Better	1 / 10	1 / 10	2 / 10	—	7 / 10

Table 4.5: Class 2 computational results - 9 Hubs, 50 Origins

Table 4.6 shows the results for the test instances with 100 origins. As our mathematical model uses all the paths between internal customers and the external hubs via one or two internal hubs, this value significantly increased the number of binary variables in the model. For this class, the first RS is within 1% of the optimal solution, but the improvement achieved by RS3 - an average of around 0.28% of the optimal solution - shows the usefulness of the tree structure. For this test class, IRSS found better solutions (-0.11% on average) than CPLEX when the exact solver was limited to the same time as IRSS.

Instance	RS1	RS2	RS3	R-Time	IRRS same-t
1	0.91%	0.45%	0.00%	1.44	-0.49%
2	1.40%	0.92%	0.02%	1.27	-0.43%
3	1.83%	1.83%	1.37%	1.30	0.98%
4	0.48%	0.39%	0.00%	0.96	0.00%
5	1.81%	0.01%	0.01%	0.57	0.01%
6	1.37%	1.37%	0.45%	0.13	0.45%
7	0.47%	0.47%	0.04%	1.75	-0.46%
8	0.00%	0.00%	0.00%	0.87	-1.18%
9	0.46%	0.46%	0.46%	1.10	0.04%
10	0.93%	0.93%	0.46%	1.54	0.03%
Average	0.97%	0.69%	0.28%	1.09	-0.11%
Same	1 / 10	2 / 10	6 / 10	—	4 / 10
Better	0 / 10	0 / 10	0 / 10	—	4 / 10

Table 4.6: Class 3 computational results - 9 Hubs, 100 Origins

The results for Class 4, shown in Table 4.7 show that the RS1 values are very close to the CPLEX results – on average 0.58% – and the RS3 results manage to attain the optimality gap. The results in the IRSS same-t column are bellow the optimality GAP. Comparing with instances of Class 2, Table 4.5, the main difference is in the performance: though for Class 2, IRSS is 22.23 times faster than CPLEX, for Class 4, it is only 1.85 times faster, but Class 4 has 3 instances strictly better than CPLEX in RS2 while Class 2 has only one instance. This difference in performance can be explained by the increased number of possible paths linking the internal customer to the external hubs, and the increasing complexity of the problem. This increasing complexity makes the problem harder to tackle in the restriction and search phases, since both phases use the same model.

As the results for largest test instances appeared to be more noteworthy, we tried a class with 24 hubs and 50 internal customers points (i.e., origins). Table 4.8 provides the results for this class. CPLEX was not able to prove the optimality for any of the instances and, for 2 instances (2 and 4), was not even able to find a feasible solution within the 15 hours allotted. In RS1, IRSS was able to find a better solution than the best known, and this solution was improved in RS2 and RS3. The importance of the tree structure is clearly illustrated in instance 7, for which the solution provided in RS1 was 2.39% worse than the best known, but by RS3, it was 0.38% better than the best known solution and used 1.86 times less computational time. The column IRSS

Instance	RS1	RS2	RS3	R-Time	IRRS same-t
1	0.80%	0.80%	0.47%	1.03	0.47%
2	0.20%	-0.20%	-0.31%	3.12	-0.31%
3	0.11%	-0.07%	-0.07%	5.59	-0.07%
4	0.39%	-0.23%	-0.23%	0.74	-0.23%
5	0.73%	0.66%	0.12%	0.81	0.12%
6	0.49%	-0.20%	-0.28%	2.44	-0.47%
7	0.96%	0.68%	0.25%	2.16	-0.06%
8	0.87%	0.63%	0.63%	2.17	0.61%
9	0.78%	0.45%	0.31%	0.37	-1.26%
10	0.47%	0.47%	-0.07%	0.10	-0.07%
Average	0.58%	0.30%	0.08%	1.85	-0.13%
Same	0 / 10	1 / 10	2 / 10	—	3 / 10
Better	0 / 10	3 / 10	3 / 10	—	4 / 10

Table 4.7: Class 4 computational results - 18 Hubs, 50 Origins

same-t shows that IRSS is able to provide solutions that are an average of 1.46% better than CPLEX, using the same computational time for all the instances. For instances 2 and 4, where no solution is available for comparison, the solution provided by IRSS is around 9.7% of the lower bound given by CPLEX. The number of instances for which our hybrid algorithm is strictly better than the exact method in the same CPU time is 9 out 10.

Table 4.9 shows how many times RS1, RS2 and/or RS3 provided the best solution for each test class. As this table shows, the algorithm continues to find better solutions in every branch. Half (25 out 50) of the best solutions were found by the time that RS3 was completed, which means that the tree structure makes the overall algorithm robust, allowing it to continue to identify and search promising regions that were not identified in the first RS. The analysis of the smallest test class shows that 8 out of 10 best solutions were found in RS1 and RS2, which means that, for small test classes, fewer branches are needed. In such cases, the third RS is usually not necessary since the first RS choice is already a very good one.

Instance	RS1	RS2	RS3	R-Time	IRRS same-t
1	-4.08%	-4.08%	-4.10%	0.16	-4.10%
2	–	–	–	–	–
3	0.01%	0.01%	0.01%	0.81	0.01%
4	–	–	–	–	–
5	0.32%	0.15%	0.15%	2.08	-0.72%
6	-0.06%	-0.06%	-0.06%	1.13	-0.43%
7	2.39%	1.63%	-0.38%	1.86	-2.25%
8	-0.45%	-0.45%	-0.53%	2.24	-2.01%
9	0.34%	0.00%	-0.11%	2.24	-0.41%
10	0.10%	0.05%	0.05%	1.96	-1.75%
Average	-0.18%	-0.35%	-0.63%	1.56	-1.46%
Same	3 / 10	4 / 10	3 / 10	—	1 / 10
Better	4 / 10	4 / 10	6 / 10	—	9 / 10

Table 4.8: Class 5 computational results - 24 Hubs, 50 Origins

	Test Class					
	1	2	3	4	5	Total
RS 1	3	5	1	0	2	11
RS 2	5	2	2	3	2	14
RS 3	2	3	7	7	6	25

Table 4.9: Improvement through the RS

4.6 Conclusion

The main goal of this study was to solve a location-allocation problem for an import/export network with economies of scale by identifying and solving a suitable RS, small enough to be thoroughly explored by a given method but still containing enough near optimal solutions to make it interesting. Our computational tests showed that IRSS performs better for larger problems, achieving an average improvement of 1.46%. In addition, exploring a restricted space allows a feasible solution to be found, even for problems that an exact method would be unable to solve in the given time span. Using a binary tree to structure the search allows our algorithm to explore multiple RS. Thus, even when the solution from the first RS is not good enough, the algorithm can adapt itself to find a better RS in the next branch. The binary tree also helps to avoid duplicate searches. IRSS is a very promising search method, which benefits both from the kind of thorough search that only an enumerative method can provide and the efficiency of the heuristic methods for solution space exploration.

Acknowledgements

Financial support for this research was provided by the Canadian Natural Sciences and Engineering Research Council (NSERC) and Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES-MEC) - Brazil. This support is gratefully acknowledged.

References

Abdinnour-Helm, S. (1998). A hybrid heuristic for the uncapacitated hub location problem. *European Journal of Operational Research*, 106(2-3):489–499.

Carrano, E. G., Takahashi, R. H. C., Cardoso, E. P., Saldanha, R. R., and Neto, O. M. (2005). Optimal substation location and energy distribution network design using a hybrid ga-bfgs algorithm. *Iee Proceedings-Generation Transmission and Distribution*, 152(6):919–926.

Cavallet, A., Malucelli, F., and Wolfler Calvo, R. (2000). A non linear multicommodity network design approach to solve a location-allocation problem in freight transportation. In *ODYSSEUS 2000 Workshop in freight transportation and logistics.* Chania (Greece).

Delmaire, H., Diaz, J. A., Fernandez, E., and Ortega, M. (1999). Reactive grasp and tabu search based heuristics for the single source capacitated plant location problem. *Infor*, 37(3):194–225.

Diaz, J. A. and Fernandez, E. (2005). Hybrid scatter search and path relinking for the capacitated p-median problem. *European Journal of Operational Research*, 169(2):570–585.

Doong, S. H., Lai, C. C., and Wu, C. H. (2007). Genetic subgradient method for solving location-allocation problems. *Applied Soft Computing*, 7(1):373–386.

Feo, T., Resende, M., and Smith, S. (1994). A greedy randomized adaptive search procedure for maximum independent set. *Operations Research*, 42:860–878.

Fischetti, M. and Lodi, A. (2003). Local branching. *Mathematical Programming*, Ser. B - 98:23 – 47.

Gendron, B., Potvin, J.-Y., and Soriano, P. (2003). A parallel hybrid heuristic for the multicommodity capacitated location problem with balancing requirements. *Parallel Computing*, 19:591–606.

Glover, F., Laguna, M., and Martí, R. (2000). Fundamentals os scatter search and path relinking. *Control and Cybernetics*, 29(3):653–684.

Hansen, P. and Mladenović, N. (2001). Variable neighbourhood search: Principles and applications. *Euroupean Journal of Operational Research*, 130:449–467.

Keskin, B. B. and Uster, H. (2007). A scatter search-based heuristic to locate capacitated transshipment points. *Computers & Operations Research*, 34(10):3112–3125.

Kim, D. and Pardalos, P. M. (2000). Dynamic slope scaling and trust interval techniques for solving concave piecewise linear network flow problems. *Networks*, 35(3):216–222.

Klose, A. and Drexl, A. (2005). Facility location models for distribution system design. *European Journal of Operational Research*, 162(1):4–29.

Lapierre, S. D., Ruiz, A. B., and Soriano, P. (2004). Designing distribution networks: Formulations and solution heuristic. *Transportation Science*, 38(2):174–187.

Lorena, L. A. N. and Senne, E. L. F. (2004). A column generation approach to capacitated p-median problems. *Computers & Operations Research*, 31(6):863–876.

Marin, A., Canovas, L., and Landete, M. (2006). New formulations for the uncapacitated multiple allocation hub location problem. *European Journal of Operational Research*, 172(1):274–292.

Pécora, J. E., Ruiz, A., and Soriano, P. (2007). High level hybrid method for the pulp production scheduling problem. *Working Paper.*

Perez, M., Almeida, F., and Moreno-Vega, J. M. (2005). A hybrid grasp-path relinking algorithm for the capacitated p-hub median problem. In *Hybrid Metaheuristics, Proceedings*, volume 3636 of *Lecture Notes in Computer Science*, pages 142–153. Springer-Verlag Berlin, Germany.

Resende, M. G. C. and Werneck, R. F. (2006). A hybrid multistart heuristic for the uncapacitated facility location problem. *European Journal of Operational Research*, 174(1):54–68.

ReVelle, C. S. and Eiselt, H. A. (2005). Location analysis: A synthesis and survey - invited review. *European Journal of Operational Research*, 165(1):1–19.

ReVelle, C. S., Eiselt, H. A., and Daskin, M. S. (2008). A bibliography for some fundamental problem categories in discrete location science. *European Journal of Operational Research*, 184(3):817–848.

Velarde, J. L. G. and Laguna, M. (2004). A benders-based heuristic for the robust capacitated international sourcing problem. *IIE Transactions*, 36(11):1125–1133.

Wu, L. Y., Zhang, X. S., and Zhang, J. L. (2006). Capacitated facility location problem with general setup cost. *Computers & Operations Research*, 33(5):1226–1241.

Part IV

Framework

The whole is more than the sum of the parts.

Aristotle (384-322 BC)
Metaphysica

Introduction

Chapters 3 and 4 developed the basic concepts of the IRSS. Chapter 3 laid its foundations by defining the basic concepts of the RSS and Chapter 4 developed these ideas repairing the drawbacks identified into the previous chapter and extends the basic solution method to a more robust iterative method which is now named IRSS. The main objective of this part is to study the trade off mechanism between the intensification and diversification, represented in this framework by the time dedicated to solve each RS and its size. The mastering of this mechanism will allow us to better understand the subtleness of the relationship between the macro and micro phases. We are also interested in observe how the the overall framework performs for the two problems proposed in Chapters 2 and 4.

In order to master this issues we implement the Framework and all its extensions, including the binary tree structure and the embedded RSS, for the two previous problems, the PPSP and the Location-Allocation. As was observed in the previous chapters the size of the RS that may be solved in the search phase changes in running time. To tackle this difficulty, a new adaptive procedure that changes the size of the RS to be explored in the search phase is proposed. Computational results shows that this adaptive procedure plays an important role in the overall performance.

It is worth to observe that this is a new flexible and generic solution method, which can implemented in several different ways. Therefore the computational tests presented in this part does not pretend to be a complete set of tests to validation of all possible combinations of methods used in the macro and micro search. This avenue of research will be explored in the upcoming years. Here we attained ourselves to a hybrid method composed of a heuristic and an exact method. Evidently different combinations may be proposed. Also, we chose to keep the same problems used during the development of the IRSS in order to have a comparison criteria. Again, there are a number of different problems in OR that are fit to continue to validate the IRSS framework and this research avenue will be explored in the upcoming years.

Chapter 5

Iterative Restricted Space Search: A Framework for designing efficient search algorithms

Despite recent advances in both heuristic and implicit enumeration methods, exploring the whole solution space for large scaled problems remains intractable. Indeed, enumeration and heuristic methods share their ability to thoroughly explore limited regions of the solution space. However, for different reasons they both fail to do it efficiently when the size of the solution space increases as in the case of most of real-life problems. Therefore, most of the research efforts in the last ten years have been concentrated in increasing the robustness of solution methods. In particular, the idea of hybridizing two or several methods has been brought in to address what we call the intensification/diversification dilemma.

The Iterative Restricted Space Search (IRSS) approach is a framework for the design of efficient search algorithms where the main idea is to iteratively define and explore restricted regions of the global solution space that have a high potential of containing good (hopefully, optimal) solutions. IRSS is a generic search approach based on the interaction of two algorithmic phases with complementary goals. The first phase identifies a Restricted Space (RS) and the second phase explores this RS. The algorithm alternates between the two phases for a fixed number of times, or until the allotted time is expired. The idea of hybridizing is fruitfully applied within the first phase of the algorithm, but also in the management of the interactions between the two phases. This chapter elaborates on the main concepts underlying the IRSS framework and illustrates its implementation with two distinct real life applications. Extensive computational experiments are used to analyze the behavior of research algorithms developed under the

IRSS framework and to provide general guidelines and insights with respect to other potential implementations.

5.1 Introduction

Research during the last ten years has resulted in numerous contributions concerning solution methods to large scaled problems. For example, commercial Mixed Integer Programming (MIP) solvers have made much progress, which is evidenced especially when the cutting plane approaches have made their way into generic solvers. The latest advances in mixed integer programming are a result of combination of pre-processing techniques, intelligent enumeration schemes (such as the so-called strong branching), and generic MIP based heuristics, such as feasibility pump (Fischetti et al., 2005). Also, heuristic methods have made important improvements. Tabu search (TS), Genetic Algorithms (GA) and Variable neighborhood search (VNS), among others, have been successfully applied when the size of the problems make them intractable by means of exact methods. However, their ability to explore large sized problems is not without a cost: heuristic methods are not able to provide proof of optimality nor are they able to estimate an optimality gap. Moreover, as the size of the instances to solve increases, more and more sophisticated methods are required to maintain a certain degree of efficiency. This has led to the development of specific tools, in contrast to the current trend that aims at transforming successful application specific tools into generic approaches that can be broadly applied.

The design of efficient yet simple solution methods remains an unresolved issue. During the last decade, researchers have concentrated their efforts on improving the efficiency of the proposed solution approaches. They have done this as a means of addressing the trade-off between the time that they allot to the search methods exploring a precise region of the solution space, and the number of such regions that can be explored or, in other words, the intensification/diversification dilemma.

Hybrid algorithms have succeeded, at least partially, in striking a balance between intense and diversified research. A hybrid method, in Operational Research, encompasses two or more solution methods aimed at making them cooperate towards the same goal. In general, when designing a hybrid method, one aims at exploiting the synergy or complementarity of the methods involved by choosing them carefully. For example, although TS has proven its efficiency for finding local optima, it has been merged with other methods such as VNS in order to diversify the search and increase overall performance, see for example (Lapierre et al., 2004). In other cases hybridization arises after a problem decomposition leading to subproblems for which efficient solution methods are available. This is the case in Doong et al. (2007) where a single-source capacitated facility location problem is decomposed into location decisions – tackled by a GA – and allocation decisions – tackled by a subgradient method. Therefore, most of the work done in hybrid algorithms is generally concerned by how to adapt two or more specific solution methods. However, little has been done towards the developing of a generic hybrid algorithmic structure independent of the search methods used.

The Iterative Restricted Space Search (IRSS) approach proposes a framework for designing efficient search algorithms that embeds the idea of intensification/diversification balance from an algorithmic point of view independent of the particular methods to be used. To this end, the IRSS Framework is divided into two phases. The first phase, named the restriction phase, performs a wide, "macro" search in the solution space, with the goal of identifying its the most appealing subregions. These subregions, named Restricted Spaces (RS), are families of solutions containing the same characteristics and, as it will be explained further on, their construction is based on the hybridization of two solution methods. After the identification of these RS, a thorough "micro" search is performed in each of the RS in the second phase, called the search phase in the sequel. The interaction between the macro and micro searches, which can be viewed as the hybridization of these two algorithmic phases, reproduces the intensification/diversification approach and enhances the robustness of the algorithm. This allows the search to be as deep as possible, but only in targeted regions of the space. To the best of our knowledge, IRSS is the first generic framework for the design of hybrid methods that explicitly aims at exploiting the synergy between a macro and a micro search using an algorithmic rather than a methodical approach.

This chapter generalizes the concepts presented in Chapters 3 and 4. In Chapter 3 we introduced a solution approach based on the identification of a unique RS, which was explored by an exact algorithm. This work laid the basis for the Iterative Restricted Search Space Framework by introducing the idea of using the main variables of the problem to identify a suitable region of the space. Chapter 4 presented and added two new concepts to the recently developed hybrid method, namely: the embedded RS and the binary tree. The embedded RS was developed to tackle a difficulty found in Chapter 3: the uncontrolled size of the RS. The binary tree offers a structure for the search procedure allowing the exploration of multiples RS without exploring the same region twice. The addition of these two new concepts makes the overall method more efficient by fixing the flaws identified in Chapter 3. Also, this chapter addresses implementation issues for algorithms designed following the IRSS framework. In particular, it proposes an adaptive method to find the best size for the RS to be solved given the allotted time.

The reminder of this chapter is structured as follows: in Section 5.2 the key concepts of the approach, such as structural variables and RS, are revised and formalized. Section 5.3 is dedicated to the description of the IRSS Framework and its phases and components. Section 5.4 presents an enhancement to the embedded RSS which allows adjusting the size of the partitions done in the RS during the embedded procedure. Section 5.5 provides the pseudo-code of the IRSS Framework. Section 5.6 presents the computational experiments for the extended, full framework. Since the implementations leading to the algorithms presented in Chapters 3 and 4 did not include all the features of the IRSS Framework, these implementations need to be extended accordingly. Finally, Section 5.7 concludes the chapter.

5.2 Key concepts

Structural variables

Broadly speaking, *Structural Variables* are the most important variables of the problem. That is, once they are fixed at a given value, the values of the other variables become almost pre-determined or highly bounded by the structural ones. Being dependent of the nature of the problem, the concept of structural variables cannot be generalized for all problems in Operational Research. In fact, it is based on the hierarchy of decision levels that exist in the problem, (Schneeweiss, 2003) but also on the solution approach used. In order to decide which variables are structural for each problem, it is necessary to consider the problem and the solution method together.

A hierarchy can usually be established between the different sets of variables in a given problem. This hierarchy is intrinsic in problems having several sorts of decisions to make or, in other words, several decision levels. For example, consider a typical location-allocation problem arising in distribution or transportation settings where one cannot decide which client will be assigned to each depot without deciding a priori which facilities are open. In this case, the structural variables correspond to the location variables. Doong et al. (2007) solve a location-allocation problem with a hybrid algorithm that tackles the location variables using a GA while the allocation variables are tackled by a subgradient method. After each GA recombination, the location variables are fixed and the subgradient method is used to find the best values for the allocation variables. This algorithm reproduces the hierarchical structure of the problem, with the GA occupying the top level and the subgradient method relegated to the bottom one. Other examples adopting this hierarchical decomposition, where each method provides a partial solution to the problem, are presented in (Potter and De Jong, 1994 ; Wiegand, 2003).

However, a hierarchy among the variables is not always clear. For example, consider the nurse/physician scheduling problems (see Carter and Lapierre, 1999 ; Pécora, 2002). In this problem, the assignment of unpopular night shifts has a greater impact on the quality of the solutions than do the assignment of day shifts. Therefore, a fair assignment of night shifts usually makes the whole work schedule looks better and increases overall nurse/physician satisfaction. Thus, although such assignment problems have only one set of variables, the ones representing night shifts are more important and therefore they can be seen as the structural variables for these problems. Another example where one can find only one set of variables, but the solution approach encompasses the concept of structural variables is the work of Lorena and Furtado (2001) that uses the concept of structural variables to build a constructive genetic algorithm applied to clustering problems. In this work, the structural variables correspond to the points acting as attractors, or the center of each cluster, while the non-structural variables represent the allocation of each remaining point to these clusters.

Restricted Space - RS

An RS is defined as a subset of the solutions universe which constitutes a promising, reduced space to search. Highly desired characteristics of this Restricted Space (RS) are: 1) it should be small enough so that it may be thoroughly explored in reasonable time, and 2) it should have a high potential of containing the global optimum or at least a good local optimum. Clearly, there is a tradeoff between the size of the RS, defined as the dimension or the number of free variables inside it, and the quality of the solutions it may contain. Evidently, the larger the RS, the higher the likelihood of containing good solutions and the higher the time required to explore it.

Structural variables play a key role in the definition of RS. Indeed, an RS is obtained by applying a restriction on all or some of the structural variables. An RS can be viewed as a region of the space in which the values for a given set of structural variables are fixed. According to the definition of structural variables, it follows that the solutions belonging to an RS have common characteristics. But how are these variables chosen and how are these values set? The objective value of the visited solutions, the history of the search, and the lower and upper bounds of the problem are used in an iterative filtering process. More precisely, promising regions are identified by analyzing how the quality of solutions is affected by particular values of structural variables. An intuitive manner of building an RS is to generate a sample of "good" solutions and then identify variables that always kept the same value in the sample.

For example, let us assume that a set of binary structural variables $Z_s, s \in S$ has been defined and that a set of "good" solutions I to the problem have been produced by any method. At this point, good solutions may be applied to locally optimal solutions or to any solution produced by an approximated method. We define $J = \{s | Z_s^i = 0, \forall i \in I\}$ as the set of the structural variables which took the value zero across the solutions in I. Intuitively, it can be expected that if setting variables in J to zero always led to good solutions, the likelihood of these variables also being zero in other (hopefully the best) solutions is high. It follows that an RS can be defined as the region for which all variables in J are set to zero. Mathematically, this restriction is expressed by the following constraint:

$$RS: \ \textstyle\sum_{s \in J} Z_s \leq 0 \tag{5.1}$$

It is worth mentioning that the complement of this region, RS^C where at least one variable in J is not equal to zero, is given by the constraint:

$$RS^C: \ \textstyle\sum_{s \in J} Z_s \geq 1 \tag{5.2}$$

Symmetrically, if we define the set $K = \{s | Z_s^i = 1, \forall i \in I\}$ as the set of variables where s took the value one in all the solutions in I, the RS and RS^C can be defined by the following inequations:

$$\begin{aligned} RS: &\quad \textstyle\sum_{s \in K} Z_s \geq |K| \\ RS^C: &\quad \textstyle\sum_{s \in K} Z_s \leq |K| - 1 \end{aligned}$$

Evidently, more sophisticated RS can be generated using adaptive memory, frequency tables or solutions weighted by objective values. In particular, Pécora et al. (2007) use the history of the search and a weighted frequency table to construct the RS.

Note that, although the RS concept concerns a set of solutions sharing some particular characteristics, it should not be confused with a neighborhood. Unlike a neighborhood, which is the region within a given distance from a reference solution, the RS does not use either a reference solution or a distance in its definition. Therefore, they have different topologies by definition. The following example illustrates the differences between the neighborhood and RS concepts. Let $s_0 = \{1, 0, 0, 0\}$ be a reference solution for a given problem. The neighborhood of s_0, $\mathcal{N}_1$, is defined as all the solutions having the Hamming distance equal to 1 to s_0. Thus, the solutions belonging to $\mathcal{N}_1$ are: $s_1 = \{0, 0, 0, 0\}$, $s_2 = \{1, 1, 0, 0\}$, $s_3 = \{1, 0, 1, 0\}$ and $s_4 = \{1, 0, 0, 1\}$. For the same problem, let $r_0 = \{1, 0, 0, 0\}$ and $r_1 = \{1, 1, 1, 0\}$ be the set of good solutions used to generate the RS. Therefore, keeping the common values of r_0 and r_1, $r_* = \{1, *, *, 0\}$, and using the RS definition given previously, the solutions $r_0 = \{1, 0, 0, 0\}$, $r_1 = \{1, 1, 1, 0\}$, $r_2 = \{1, 1, 0, 0\}$ and $r_3 = \{1, 0, 1, 0\}$ belongs to the RS.

As illustrated by this example, the neighborhood does not necessarily include the reference solution but the RS contains all the solutions used to form it. Note also that a part of the solution – the first and the fourth variables in the example – always remain fixed but they all change in the neighborhood. In fact, we consider a neighborhood as all the solutions that belong to a given distance, using a specific metric which is defined by the neighborhood, from a starting point, what mathematically is a closed ball. On the other hand, the RS is a polytope, so they are different mathematical entities.

Finally, it should be noted that despite the care taken in the design of an RS, there is a risk that it reveals a "poor" RS – i.e. an RS which does not contain either the optimal or near-optimal solutions. This is the why the IRSS framework proposes an iterative structure that allows for the search of multiple and distinct RS. Even if the first RS does not contain good solutions, the following RS will lead the search to other promising regions. The mechanisms driving the interactions between consecutive iterations, as well as the global picture of the IRSS framework, are described in the next section.

Direction vectors

Driving the search to potential regions of the solution space is not an easy task. Almost every solution method aims at driving the search to regions containing good solutions, but very few do it explicitly, usually good regions are found only based on the last (or the best) solutions found. Here we propose to use information in the structural variables as search directions. As said before, the structural variables are the main variables of the problem, so if these variables are set to good values it will intuitively drive the solution approach to good regions of the space.

As will be detailed in the next section, the restriction phase is a hybrid method itself having two component methods. The first look for good values for the structural variables neglecting the non-structural ones and the second evaluates the final structured proposed by including the non-structural variables in the solution. The drawback in exploring the solution space using just the structural variables is that we are not aware of the impact of the non-structural variables in the solution. The direction vectors take this issue in consideration by comparing the values assigned to the structural variables before and after the inclusion of the non-structural ones.

To illustrate the concept of direction vectors we chose the classical capacitated location allocation problem. The main decision for this problem, the structural variables, is the location decision, the hubs to be opened and its respective capacity. The allocation variables are the non-structural ones. Figure 5.1 shows the tree possible capacity implementation costs for the hubs. The light grey area represents the available capacity given by the structural variables. When evaluating these configuration, the total passing flow in this hub may be in the first part (represented by the dark grey area), meaning that the capacity is under used. This is a clear indication that the structure proposed may be not the best and should be reevaluated. In this case a natural search direction will be to force the second level to close and reevaluate this new structure. This information is generated in a form of a direction vector having its origin in the solution given by the structural variables and pointing to a possible profitable new region of the solution space. This direction vector is then passed to the structural method where the search restarts. The next section describes the restriction phase and details the use of the direction vectors.

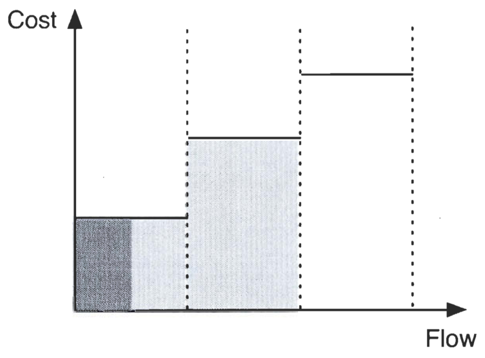

Figure 5.1: Direction Vectors

5.3 IRSS Framework

This section presents a complete description of the IRSS Framework detailing each phase, the structure in which it is implemented, and the flow of information between its components. This framework applies the concept of hybridization in two different ways; first, during the restriction phase and second, in the management of the interactions between the restriction and the search phases. It should be noted that, depending on the particular methods implemented in the restriction phase, an initial solution may be required to start the algorithm. In this case, the initial solution can be produced by a constructive procedure, a reduction or a relaxation of the main problem or any other heuristic method, as the robustness of the framework makes the overall method non-sensitive to the quality of the initial solution. Finally, this section concludes by discussing whether the framework can be implemented as a heuristic or an exact method depending on the solution method used in the search phase.

The IRSS Framework is divided into two phases: The first phase, named Restriction Phase, performs the macro search in the solution space. This phase is not interested in finding good solutions, or in other words, it is not a local search. Instead, it focuses on finding common characteristics of good solutions. These common characteristics define a family of similar solutions belonging to the same region of the space. In order to identify these characteristics, the restriction phase should explore the solution space as widely as possible to gather as much information as possible. This wide exploration, or probe, is more efficient if the search method used in the restriction phase concentrates its efforts in simply the most important variables of the problem, the structural variables. Once these families of solutions, or Restricted Spaces (RS), are identified the method goes to the second phase, the Search Phase, where a thorough exploration of each subregion is performed. The IRSS Framework iterates between Restriction and Search Phases until a stopping criterion is met. In addition, an Initial Phase with the goal of finding an initial solution may be included if needed.

Restriction phase

The Restriction Phase aims at probing the solution space in order to identify interesting solution subspaces or appealing research regions. Rather than execute a local search, such as neighborhood based metaheuristics, this phase performs a macro search, covering a larger portion of the solution space. We therefore decided to elaborate the restriction phase around the concept of structural variables, proposing a hybrid algorithmic structure that alternates between two methods having complementary goals. The first, named structural method, focuses on the evaluation of the impact of structural variables on the solution, and a second method, named the evaluation method, adds the contribution of the non-structural variables. Evidently the manner in which these two methods interact is of highest importance and it will be described in the forthcoming paragraphs.

The structural method may be designed in such a way that only structural variables are concerned by means of a problem relaxation, partition, or simplification. For example, in a location-allocation context (e.g., a distribution network), it is possible to focus exclusively on location variables by grouping individual customers into super-customers, leading to a simplified network. Clearly, even if an optimal location is found to the simplified problem, there is not guaranty of its efficiency when applied to the original problem. Moreover, a solution for the facilities location is nothing but a partial solution to the location-allocation example. This is why the evaluation method is used. Its role is to find good, often locally optimal, values for the non-structural variables given a particular partial solution produced by the structural method. It can occur that finding a feasible solution requires some adjustments in the values of the structural variables. Moreover, it could be expected that starting from the partial solution produced by the structural method, the evaluation method would be able to question and then to modify the values assigned to the structural variables. For example, when solving the allocation part of the location-allocation problem, we could realize that one of the open depots is underused and consequently, it should be closed or have its capacity decreased (in the case of the generalized location-allocation problems). The importance of such adjustments or modifications and how they are taken into consideration with the goal of building an RS is now explained.

At the end of the evaluation method two solutions are available. One is a partial solution having interesting values for the structural variables, Z_s^1, and another is complete and feasible solution, where the structural variables take values Z_s^2. The comparison between them gives a search direction, starting from a potential region of the space and pointing to a feasible solution. Defining the vector $\Delta = Z_s^2 - Z_s^1$, one can measure the difference between the information produced by the structural method and the impact of assigning the non-structural variables. The interpretation of the values in Δ is rather intuitive. When different from zero, it indicates that the two methods **do not agree** on the value of the particular variable and further search needs to be done to solve this issue in a new structural-evaluation methods iteration. At each new iteration the direction vector, as well as the last solution produced by the evaluation method, are used by the structural method. Moreover, all the locally optimal solutions found during the restriction phase will be taken into consideration to generate an RS, as described in section 5.2.

Search phase

The objective of the search phase is to thoroughly explore a given RS. An exact or heuristic method can be used to implement this phase. However, the manner in which the exploration is accomplished determines if the overall method performs as a heuristic or an exact method. If the RS is completely explored, it can be excluded from further search, but this is not the case if the search method performs a heuristic exploration. A deeper discussion about this issue is done further in this chapter.

Iteration between the restriction and search phases

The IRSS Framework proposes an iterative algorithmic structure between the restriction and search phases that can be seen as the hybridization of the macro and micro searches. This schema offers two advantages: first, it reproduces the intensification/diversification approach, and second, since there is no way to insure that an RS produced by the restriction phase contains the optimal or near-optimal solutions, exploring several RS enhances the robustness of the algorithm. But the interactions between the restriction and search phases need to be designed and carefully implemented to keep track of the visited RS and to maximize the efficiency by avoiding search of previously explored regions. To this end, the IRSS framework proposes an algorithmic implementation using a binary tree structure as the one depicted in Figure 5.2. In particular, Figure 5.2 illustrates two iterations of the Framework. The algorithm starts when the initial solution X0 is passed to the restriction phase, which divides the whole solution space S into a region $RS1$ and its complementary space $RS1^C$. $RS1$ is thoroughly explored by the search phase in the first left branch, producing a best solution $X1$. Then, the restriction phase is applied in the unexplored space $RS1^C$ to identify a new region $RS2$. Note that by construction $RS1$ and $RS2$ are disjoints because $RS2 \subset RS1^C$. In the second left branch, the search phase explores $RS2$ that returns a better solution $X2$. Finally, a new restriction phase is applied in the remaining unexplored space $RS2^C = RS1^C \backslash RSS2 = S \backslash RS1 \backslash RS2$. The search continues until a stop criterion (for example, a fixed tree depth, a fixed number of branches) is met. The particular case in which the algorithm performs as an exact method will be discussed in the next section.

It is important to note that the particular tree structure adopted here encompasses elements of both the classical Branch and Bound tree (Land and Doig, 1960) and the relatively new Local Branching concept proposed by Fischetti and Lodi (2003). Indeed, as in the case of the Branch and Bound approach, defining an RS consists in fixing some variables at given values in order to obtain a restriction or a projection of the solution space. However, the tree structure in the framework (similarly to Local Branching) branches in a set of variables instead of in one single variable like the standard Branch and Bound. Yet, the framework's tree structure also differs from Local Branching. The main difference is that in Local Branching, the left branches are defined by neighborhoods around the solution found in the upper left branch, while the

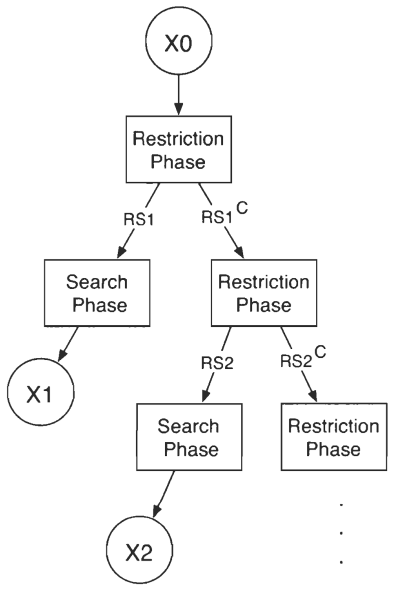

Figure 5.2: Binary tree structure

left branch in the framework's tree is an RS generated by the restriction phase. In other words, the framework does not only use the last solution found to determine the next region to be searched and consequently two consecutive RS may identify very different regions of the space. This results in a branching structure that is not necessarily local. Table 5.1 summarizes the main aspects of each branching structure, illustrates the similarities and differences of the IRSS branching structure with respect to both Local Branching and Branch and Bound.

	RSS Framework	Branch and Bound	Local Branching
How is the branching is done?	By fixing variables at a given value	By fixing variables at a given value	By defining a region around a given solution (neighborhood)
Several or one variable at a time?	Several	One	Several

Table 5.1: Branching structures

Heuristic or exact method?

The search strategy embedded in the IRSS framework, which consists in defining restricted regions of the solution space to be searched, may perform as an exact search tool provided that (1) an adequate number of RS is explored and (2) the exploration of each RS is performed to optimality. Given the tree structure proposed by the IRSS framework, the manner in which each RS (each left branch of the tree) is explored determines if the framework implementation is an exact, ε-optimal, or heuristic method. The first two cases may be obtained only if an exact method is used.

When the left branches are explored by an exact method, four possible results are possible: *Infeasible, Optimal, Feasible or Unknown.* An *Infeasible* or *Optimal* result means that the RS was completely explored and therefore it can be excluded in further searches. A *Feasible* result suggests that the RS may be cut (using the branch and bound terminology) if the optimality gap is less than ε in an ε-optimal approach or ignored in an optimal approach. An *Unknown* result is obtained when no feasible integer solution is found in the time allotted to search the RS. In other words, this result implies that the RS was not completely explored. Nevertheless, the best lower bound obtained (for minimization problems) may be compared to the best solution found so far and consequently the same reasoning applied to *Feasible* branches holds. Clearly, *Unknown* is the least useful result. In order to minimize these kinds of unproductive search Pécora et al. (2007) (see Chapter 4) propose a procedure called embedded RSS, which is an incremental exploration of the RS, allowing a part of the RS be excluded even in cases where the complete RS is declared *Feasible or Unknown* to be cut.

Also when considering a minimization [maximization] problem a lower [upper] bound is required in order to prove optimality, which in this case may be provided by a linear relaxation before each restriction phase, as seen in Figure 5.2.

When exploring the left branch using an approximate method, one cannot insure that the RS has been fully explored and therefore good solutions may have been missed. On the other hand, metaheuristics have been empirically proven to be very efficient solving large problems. This is why in cases where very large RS are generated, metaheuristics are the best methods to use in the search phase. In this case the framework is a heuristic method since the optimality cannot be proven.

Moreover, the way to represent the regions RS and RS^C plays an important role in the overall framework. In the example showed in the Section 5.2 we used the constraints 5.1 and 5.2 because they are suitable to the method used in the search phase, an exact method. When using other methods in the search phase, the representation of the regions RS and RS^C should be adapted accordingly. For example, if the RS will be explored by a meta-heuristic, it is more efficient to fix the variables directly at their values instead of using the constraints 5.1 and 5.2, when using constraint programming another type of constraint may be used.

Despite the way chose to represent RS and RS^C the basic idea is the same, they can be used to temporarily prohibit the search inside a region of the space. This temporary prohibition is the same as putting some regions of the space in a tabu state. Thus, using the idea of tabu list applied to regions of the space, rather than variables or neighborhoods, the tree structure can be heuristically explored ensuring an adequate diversification of the search.

Information flow

The exchange of information, and how it is transformed into intelligence, is the basis of the IRSS Framework. Unlike most of the hybrid algorithms proposed in the literature, the manner in which the solution methods share and use information is the greatest strength of the IRSS Framework. In particular, the IRSS framework includes two different information loops that are shown in Figure 5.3. The first loop takes place inside the restriction phase; consequently there is a flow of information between the structural and the evaluation methods. As can be observed, the structural method feeds the evaluation method with a partial solution (concerning only the structural variables) to the problem. On the other hand, the evaluation method suggests search directions (provided by the direction vectors) to the structural method. The second loop concerns the interactions between the macro and micro searches, i.e. the interactions between the restriction and search phases. As illustrated by Figure 5.3, the restriction phase produces a set of restrictions (the RS) that limit the region to be explored by the search method. However, historical information on the search done so far, like best solutions and bounds, are also used in the search phase. The information exchange between the search and the following restriction phase depends on the type of result produced by the exploration of the RS. Two cases may arise. If the RS was completely explored, the RS can be excluded in further iterations. However, if the RS exploration could not be completed in the allotted time, more sophisticated information is required to decide whether to eliminate a part of the RS or the whole RS, in which case the IRSS become a heuristic method. The causes leading to these particular situations have already been outlined in the two previous subsections.

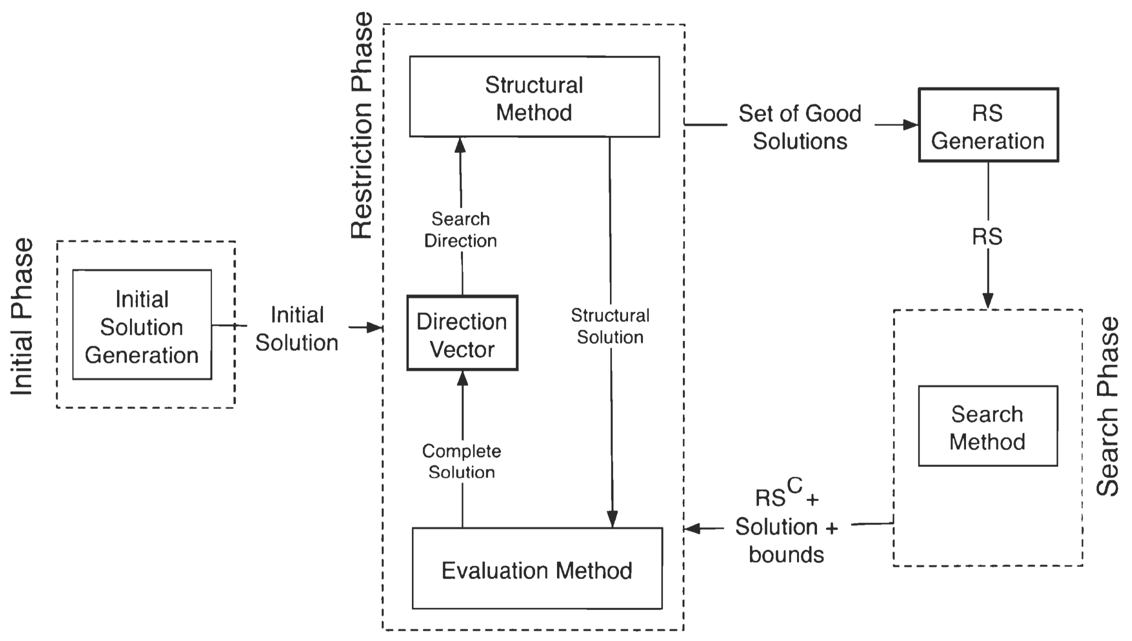

Figure 5.3: Generic solution approach

5.4 An improvement to the IRSS Framework: dynamic embedded RS

The computational experiments in Chapter 4 showed that, although the Embedded RS helps to perform a more efficient exploration of the RS, it has several parameters that are very difficult to set. For example, the size of the first partition in the RS needs to be known a priori. Moreover, the scaling step (the increase in the size of two consecutive partitions in the embedded RS execution) also takes a fixed value. Consequently, the performance of the procedure may fail if these parameters are not set carefully. The natural approach to avoid this difficulty is to conceive a self-adaptive embedded RS. In our case we propose a set of rules that adapts the target size of each new partition within the embedded RS approach. These rules of thumb are based on empiric observations concerning how the exploration time increases with the RS size (see Section 3.3 in Chapter 3).

However, before explaining these rules, we need to explain the difference between the targeted size of the RS and the current size of RS, denoted tRS and cRS respectively. While the latter is the size of the last partition treated, the former provides the size that we expect to be able to solve in the allotted computational time. Assuming that a maximum constant computational time, t_{max}, is allotted to solve each partition within the embedded RS scheme, we define t, $t \leq 1$, as the ratio between the time required by the last explored partition and t_{max}. Therefore, $t = 1$ means that the last partition was not explored to optimality, and smaller values of t suggest that larger partitions may be solved in the allotted time. Two possible implementations of the adaptive Embedded RS procedure are proposed. The "conservative" method simply starts with a very small cRS and this value increases by the adapting rules. In contrast, the "optimist" method starts with the largest RS and, if the exploration does not succeed, this value will be reduced by the adapting rules.

Three rules are proposed to manage the size of the targeted RS. The first rule states that if the last partition was not completely explored in the allotted time, the size of the next partition needs to be reduced by half. The second rule concerns the case where the last exploration was "fast" with respect to the allotted time, fast meaning, in our case, less than half the allotted time. It is therefore reasonable to expect that larger partitions may be explored within t_{max} so tRS is enlarged. Finally, tRS is not changed if the allotted time seems to be adequate to solve the targeted RS size. The following equations formalize the mentioned rules.

- Rule 1 - If $t = 1$

  tRSS = cRS/2

- Rule 2 - If $t \leq 0.5$

  tRS = Max$\{tRS, cRS + \lfloor -log_2(t) \rfloor\}$

- Rule 3 - If $0.5 < t < 1$

  No change in the tRS

It is worth recalling that the comparison threshold ($t = 0.5$) and scaling factors (0.5 and $log_2(t)$) were chosen arbitrarily after remarking that the exponential relationship between the solving time with respect to the size of an RS (Figure 3.3 in Chapter 3).

5.5 IRSS Framework pseudo-code

This section presents the pseudo-code of the IRSS Framework. The algorithm below has all the components and concepts defined in this thesis. The two phases (restriction and search) are shown in steps (4 to 9) and (11 to 12) respectively. Nevertheless, in some cases an initial phase producing a first starting-solution (step 1) may be required according to the restriction phase characteristics. As can be seen, the algorithm has two nested loops: one large loop that defines the search strategy by alternating between the macro and micro searches, and one inner loop within the restriction phase. Steps 1, 5, 7 and 11 call the different solution methods in the hybrid structure. These methods may be standard or tailor made to better match the requirements of the problems. The algorithm ends by returning the best solution found so far.

Algorithm 5 IRSS Framework

```
1: Build an initial solution
2: Search direction: $\Delta \leftarrow \emptyset$
3: while stop criteria is not met do
4:    repeat
5:       Search in the structural variables using ($\Delta$) as direction
6:       Bound the structural variables
7:       Search in the non-structural variables
8:       Generate the new search directions ($\Delta$)
9:    until enough solutions to generate the RS
10:   Generate the RS
11:   Solve the RS problem inside
12:   Generate the $RS^C$ and add it to the main model
13:   Update the best solution
14: end while
15: Return the best solution
```

5.6 Two practical implementations of the IRSS framework

The purpose of this section is threefold. Firstly, it aims at assessing the potential of the IRSS Framework as a guide to design search algorithms. Secondly, it intends to evaluate the impact of new adaptive embedded RS with respect to the version proposed in Chapter 4. Finally, it addresses important implementation issues concerning algorithms designed following the IRSS framework. In particular, the trade off between the number of iterations (i.e. the number of RS) and the time allotted to each of them is discussed and some insights or rules are provided.

To aid in this, the instances generated for the two application problems proposed in Chapters 3 and 4 will be solved again. However, it is important to remember that the solution methods proposed in these Chapters only partially implemented the concepts and components of the IRSS Framework. Therefore, before proceeding to execute the computational tests, these methods needed to be extended to match the IRSS Framework. Although the search phase and the search tree can be added in a rather straightforward manner to the hybrid heuristic for the PPSP described in Chapters 3, implementing the Embedded RSS is not trivial. Consequently, the interested reader may find some details concerning this implementation in the Appendix B. Table 5.2 summarizes the components used in the extended versions of the algorithms presented in Chapters 3 and 4, respectively.

	PPSP	Location-Allocation
Structural Variables	Basic Density's level of work	Hub's flow
Initial Phase	Rolling Horizon	Trivial Solution
Restriction Phase		
Structural Method	Rolling Horizon	DSSP hub
Evaluation Method	Greedy allocation procedure	DSSP arcs
Search Phase	**Embedded RSS** with Cplex 8.1	Embedded RSS with Cplex 10
Tree Structure	**Not Present**	Present
Adaptive RS Size	**Not Present**	**Present**

Table 5.2: RSS Framework components used in the final tests

All the numerical tests were executed on *AMD Opteron 250 2.4 GHz* computers with at least 4Gb of RAM running the *Linux* operating system. We did not solve all the tests instances proposed in Chapters 3 and 4. Instead, we preferred to focus exclusively on the most difficult, largest sized test classes (i.e., the 32 period class for the PPSP and the 24 hubs class for the location allocation problem). All the values reported in the following section are averages over all the instances in the selected classes. In order to be consistent with the results reported in the other chapters, the performance figures in this section are not absolute values but the normalized gap between the best solutions found by the IRSS methods (Obj_{IRSS}) with respect to the best solutions found by the previous implementations (Obj). The gap is computed by $Gap\% = 100 * (Obj_{IRSS} - Obj)/Obj$. The two next subsections present the numerical results and their analysis for each of the two considered problems.

Numerical results for the Pulp Production Scheduling Problem

This section reports the computational results produced for the Pulp Production Scheduling Problem. Since one of the objectives of this analysis is to determine the performance of the method with respect to the time that is allotted to solve each RS, each of the Figures 5.4 to 5.7 rapports five cases where the time span alloted to solve each RS is respectively $\{2, 10, 20, 30, 60\}$ minutes.

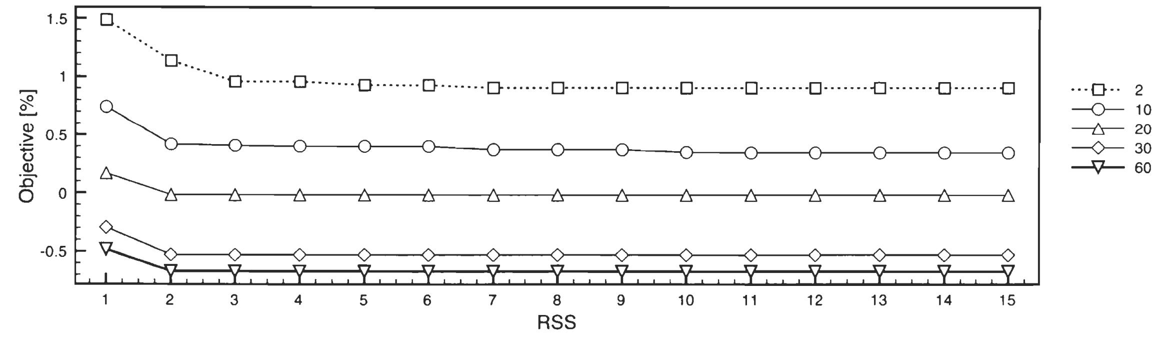

Figure 5.4: PPSP - Objective function

In Figure 5.4 the abscissa axis is the depth of the branch and the ordinate axis shows the normalized gap between the best solution found by the IRRS and the best known solution. As it could be foreseen, the quality of the solution found is proportional to the time spent at each RS. Being around 1.5% worse than the reference solution when using 2 minutes for each RS (the reference solution used 20 minutes) and was approximately 0.7% better if 60 minutes is used. However, comparing the curves at 30 and 60 minutes we can see that the improvement is negligible. Also, independently of the allotted time, there is a stagnation of the best solution found after the 3^{rd} RS explored.

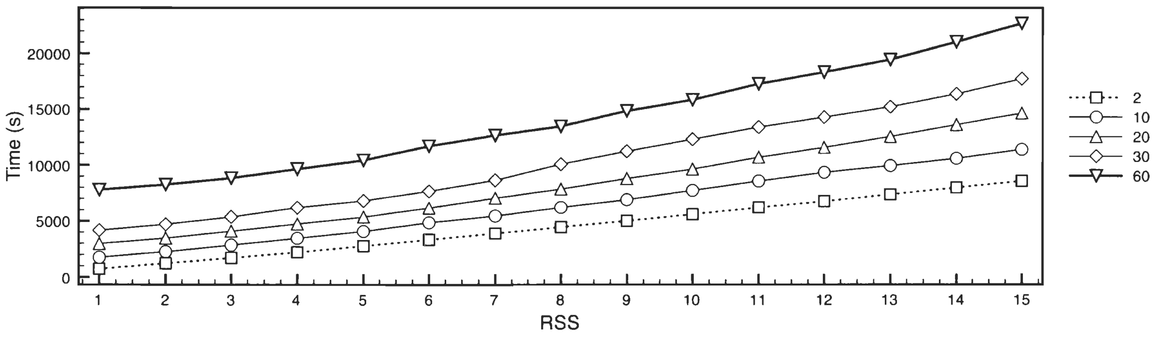

Figure 5.5: PPSP - Total time

Figure 5.5 shows the cumulative execution time (including the restriction and search phases) with respect to the number of RS explored. It can be observed that, in all the cases, the cumulative time increases linearly with the number of RS – a linear regression computed an $R^2 \sim 0.99$ for all the curves. One can conclude that the time spent for defining and solving each RS is almost constant.

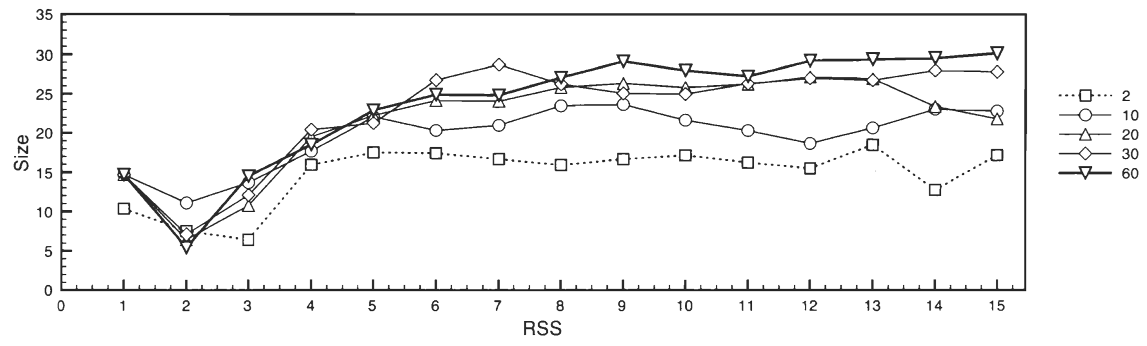

Figure 5.6: PPSP - Solvable RS size

The behavior of the *Adaptive embedded RS* can be observed in Figure 5.6. The ordinate of this figure shows the size of the RS solved until the optimality or infeasibility was proven; in other words, the size of the RS that was excluded from the solution space. The behavior of these curves starting solving small RS and increasing the size until a stability is reached is an empirical prove that the rules used in the *Adaptive embedded RS* are useful and help to converge to RS of solvable sizes. Moreover, we noted that after the 6th branch the sizes were ranged inside a small interval, for example the *2 min* curve is kept around a size of 17 while the 60*min* is between 25 and 30.

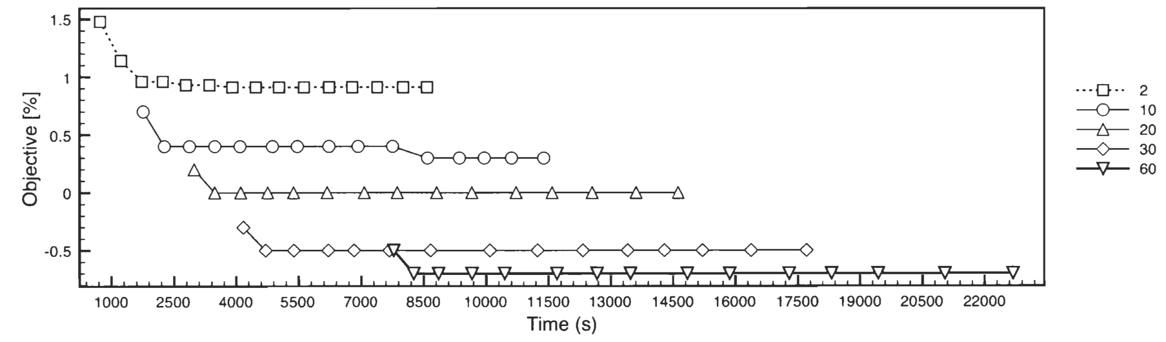

Figure 5.7: PPSP - Trade off time and solution quality

The last figure will help us analyzing the trade off between the exploration time and the quality of the solutions produced. Figure 5.7 reports how the Gap% evolves during the execution of the method. This graphic is similar to the one in Figure 5.4, but the abscissa here is the CPU time instead of the number of branch. It can be observed that the curves rarely crosses each other. Also, in all the cases the search stagnates quickly (i.e. there is no improvement in the objective value), which requires extended discussion.

First of all, let us recall that PPSP is a difficult combinatorial problem. We tried to solve the formulation proposed in Chapter 2 with a commercial branch-and-bound algorithm and we observed that, in general, the method was able to find a good integer solution in a rather short time. However, the optimality gap of the first integer solution was, in most of the cases, very big. The branch and bound explored different branches and, since very few of them were "pruned" or eliminated from the search, the procedure usually took a lot of time. This behavior is classic of combinatorial problems for which exists a very large number of local optimal solutions having similar objective values. The fact is that, although the integer solutions produced by the method were of a very high quality, neither a proof of optimality nor an acceptable optimality gap were available in a reasonable time. When solving instances of PPSP with the RS method, we observed that the quality of the solutions produced was as good as the ones produced by the exact method but again, it was not possible to be sure that better solutions were missed out of the RS. The IRSS method brought new information. In particular, Figure 5.4 confirms that it exists several local optimal solutions having similar objective values. Indeed, we observed that several RS – which are distinct by definition – lead to similar best solutions. Moreover, the quality of the solutions produced depends strongly of the time span. Therefore, we can conclude that the RS proposed by the method are of a high quality but, on the other hand, the time alloted to the RS exploration need to be selected carefully. In order to confirm our conclusions concerning the efficiency of the IRSS approach, the results produced for the location-allocation problem are presented in the next section.

Numerical results for the Location-Allocation Problem

This section reports the computational results for the location-allocation problem presented in Chapter 4. All the values shown are averages of the test instances for the largest test class (24 Hubs). Figures 5.8 to 5.11 follow the same pattern as for the PPSP, showing 5 lines, each one corresponds to the time allotted, $\{5, 10, 20, 30, 60\}$ minutes, to solve each RS.

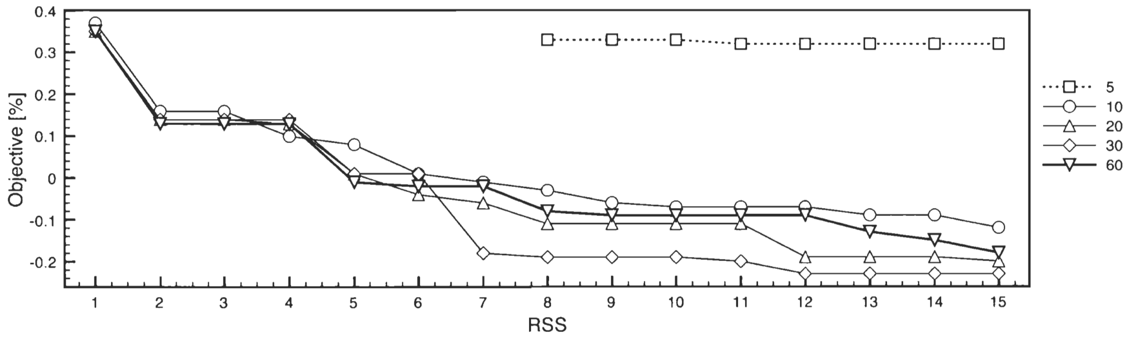

Figure 5.8: Location Allocation - Objective function

Figure 5.8 shows, for each time span, the best solution (in terms of Gap%) found as the number of explored RS (iterations) increases. By looking at Figure 5.8 we conclude that 5 minutes is not enough to insure a good exploration of the RS: the method was not able to produce solutions for several RS and, when it provided it, the quality of the solutions produced was rather poor. On the other hand, Figure 5.8 presents a very different behavior if compared to Figure 5.4. In fact, we observe that in all the cases but the 5 minutes time span, the quality of the solutions improves continuously and that the 20, 30 and 60 minutes perform in a very similar manner. One could also conclude that is not worth to spend 60 minutes or more time exploring the RS.

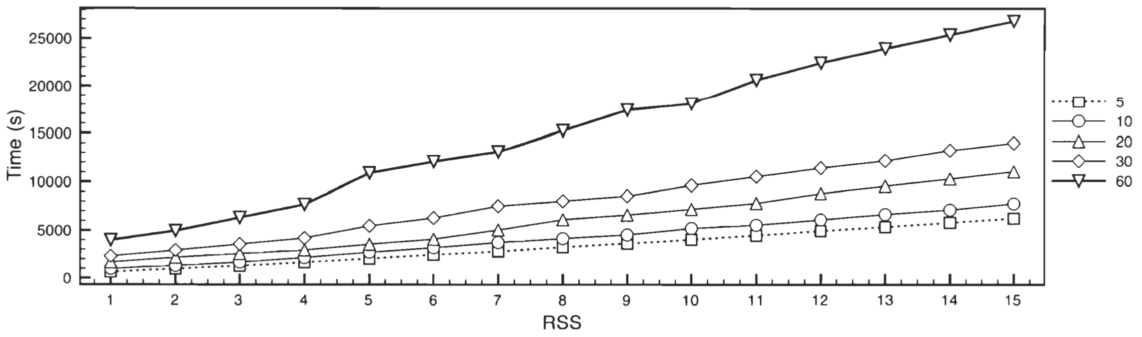

Figure 5.9: Location Allocation - Total time

The total time shown in Figure 5.9 also follows a linear progression trough the RS branching having an $R^2 \sim 0.99$, as in the PPSP computational tests. Thus the total time spent in the identification of exploration of each RS is also constant.

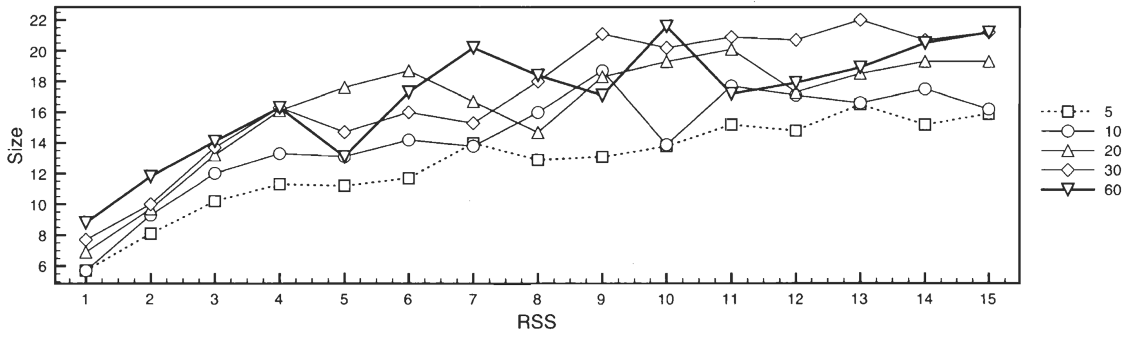

Figure 5.10: Location Allocation - Solvable RS size

Figure 5.10 illustrates the impact of the *Adaptive embedded RS* procedure on the size of the RS. All these curves present a similar behavior, smoothly increasing their values in the first 6 RS after which they are kept inside a small range varying no more than 5 free variables. This behavior is coherent with what we expected from the rules used in the *Adaptive embedded RS* procedure.

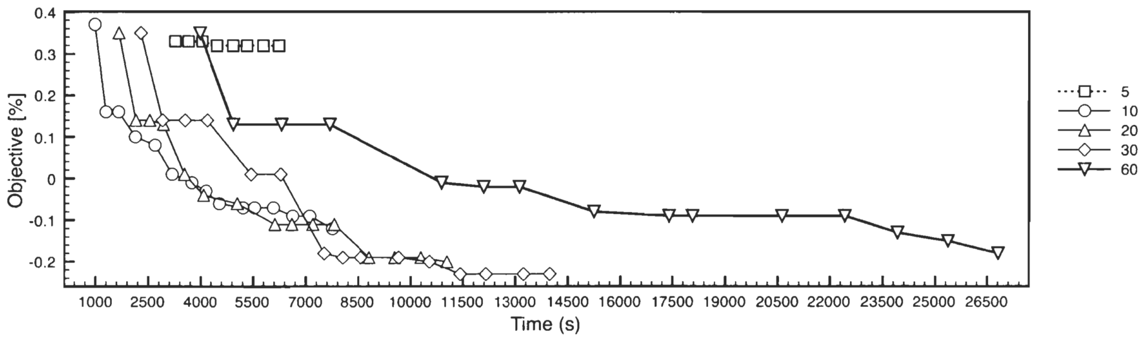

Figure 5.11: Location Allocation - Trade off time and solution quality

Figure 5.11 reports how the quality of the solution increases (the Gap% decreases) with respect to the execution time. This graphic is similar to the one in Figure 5.8, but the abscissa here is the cumulative CPU time instead of the number of the branch. As mentioned before, the behavior of the line corresponding to a time span of 5 minutes will not be taken into account for our analyzes. Unlike in Figure 5.7, where it can be observed that the search stagnates quickly, Figure 5.11 shows that the best solution improves continuously. Again, choosing a time span of 20 or 30 minutes for each RS seems to be the right choice for the solved instances, as confirmed by the fact that the pattern of the 60 minutes span converges to the same results but in a much longer time frame.

In general terms, one can conclude that the two solution methods present good performances. Applied to two distinct problems, they are able to produce solutions which improve the best ones produced by a commercial branch and bound software in a fraction of the time. Moreover, both methods success doing so for two problems having very different characteristics. Indeed, the topology of the solution space associated to the location-allocation problem differs of the "degenerated" one shown by the PPSP. Since little effort has been dedicated to the design of specific solving tools for each of the problems, we conclude that the IRSS algorithmic framework exploits in an adequate manner the synergies between its component methods leading to efficient search algorithms.

5.7 Conclusion

This chapter proposes a generic Framework for the design of hybrid solution methods. The Iterative Restricted Space Search Framework (IRSS) is based on the collaboration of two search phases having complementary goals and exploits the concept of algorithmic hybridization. A "macro" search phase which probes the solution space to find appealing regions is hybridized with a "micro" search phase which performs a thorough exploration of these targeted regions, thus reproducing a diversification/intensification scheme. The concept of hybridization is also used in the macro search phase to identify promising regions and to define restricted spaces (RS) within these regions. The IRSS Framework is independent of the solution methods used in each phase and can be applied to a large range of problems. In particular, it has been used to implement solution methods for two real-life problems, a scheduling problem inspired by a paper and pulp production company, and a location-allocation problem based on the context of an international transportation company. The computational results produced by the IRSS implementations prove the efficiency of the approach when compared with previous solution methods and with a commercial MIP solver.

References

Carter, M. W. and Lapierre, S. D. (1999). Scheduling emergency room physicians. Technical Report 99-23, Centre for Research on Transportation, Montreal, Canada.

Doong, S. H., Lai, C. C., and Wu, C. H. (2007). Genetic subgradient method for solving location-allocation problems. *Applied Soft Computing*, 7(1):373–386.

Fischetti, M., Glover, F., and Lodi, A. (2005). The feasibility pump. *Mathematical Programming*, 104(1):91–104.

Fischetti, M. and Lodi, A. (2003). Local branching. *Mathematical Programming*, Ser. B - 98:23 – 47.

Land, A. H. and Doig, A. G. (1960). An automatic method of solving discrete programming-problems. *Econometrica*, 28(3):497–520.

Lapierre, S. D., Ruiz, A. B., and Soriano, P. (2004). Designing distribution networks: Formulations and solution heuristic. *Transportation Science*, 38(2):174–187.

Lorena, L. A. N. and Furtado, J. C. (2001). Constructive genetic algorithm for clustering problems. *Evolutionary Computation*, 9(3):309–327.

Pécora, J. E. (2002). Alocação de médicos em salas de emergências - uma abordagem híbrida. Master's thesis, Campinas University, Campinas, Brasil.

Pécora, J. E., Ruiz, A. B., and Soriano, P. (2007). High level hybrid method for the location-allocation problem. *Working Paper*.

Potter, M. A. and De Jong, K. A. (1994). A cooperative coevolutionary approach to function optimization. In *The 3rd Parallel Problem Solving from Nature (PPSN), Proceedings of the Conference.*, pages 249–257. Springer-Verlag.

Schneeweiss, C. (2003). *Distributed Decision Making.* Springer, New York, 2 edition.

Wiegand, P. (2003). *An Analysis of Cooperative Coevolutionary Algorithms.* PhD thesis, George Mason University.

Chapter 6

Conclusions

In this thesis we have proposed a generic and flexible framework for hybrid solving methods. As optimization problems arise in all fields of science and engineering and as the size or the complexity of such problems evolves exponentially, the development of efficient solving methods is of utmost importance from both practical and theoretical points of view. We have applied the proposed method to two real-world problems: the Pulp Production Scheduling Problem and a Location-Allocation Problem in an import/export network.

6.1 Summary of contributions

The rationale behind this framework is quite intuitive: to perform a fine search only when it is worth. However, the main challenges found during the development of this thesis were to answer the two questions resulting from this intuition. (1) How should this idea be implemented? (2) How do we identify a space to search thoroughly?

Chapter 5 provides what we hope to be a complete answer for these two questions by: (1) proposing a generic algorithmic structure for solution methods which is implemented using two search phases - "macro" and "micro" exploration. The "macro" search is responsible for identifying the suitable space, named Restricted Space, to be explored in the "micro" search. The communication between the phases comes from the proposed algorithmic structure, therefore it is independent of the component methods used within each phase. (2) To identify the RS – performed in the "macro" search – we proposed a hybrid algorithm also composed of two phases. The first phase will probe the solution space focusing on finding good values for the structural variables – defined as the main variables of the problem – once the structural variables are fixed, which defines a region of the space, the second method evaluates this region by finding good values for the non-structural variables, thus determining a complete solution that is representative of the region. Once a number of such promising regions are found they are used to define the RS, by identifying the common characteristics of the representative solutions for each region. Even though Chapter 5 provides a fully developed answer for these questions, the importance and the contribution of the Chapters 1, 3 and 4 cannot be neglected.

Chapter 1 provided the fundamental basis for the development of the hybrid framework. The study about the types of exchange of information between the component methods – which we consider as one of the main characteristics of hybrid methods – allowed us to identify that, to the best of our knowledge, there were no published papers presenting a generalization of hybrid approaches based on information exchange. Also, the study of the possible restriction spaces approaches and decomposition methods gave us good insight as to the direction which could be followed during this research. Chapter 3 laid the basis for the development of the IRSS Framework. The main concepts of "macro" and "micro" search and structural variables were already present in this work; however they had not yet been generalized. Chapter 4 proposed solutions for the two drawbacks found in Chapter 3, the size of the RS which could not be precisely controlled and the need of the exploration of several RS, by proposing the embedded RS – an incremental search procedure – and the binary tree structure to explore several RS. These improvements increased the efficacy and the efficiency of the IRSS.

The application of the IRSS Framework in two very different real-world problems, that have been used to illustrate the framework, is a significant part of the contributions of this thesis. To this end we carried out extensive computational experimentations for both problems. These tests demonstrated the importance of each solution phase and the increase of the overall efficiency by structuring the search using a binary tree. Also, the study of the trade-off between exploring several small or few large RS – presented in Chapter 5 – has a noteworthy importance in the contributions of this thesis. In order to study this trade-off an extension for the IRSS – the dynamic Embedded RS – was developed and presented in Chapter 5.

As the IRSS Framework is based on the interaction between two algorithmic phases, performed by the exchange of information. It is classified as flexible, due to the fact that almost any available solution method can be used as a component in its phases. It is also classified as a generic solving hybrid approach, because it can be applied to practically any problem in OR.

Consequently, the IRSS Framework has proven to be a very promising search method that fulfills its main objective of identifying promising subregions of the solution space to be explored in reasonable time by a dedicated method. It is based on the exchange of information between two phases making it an efficient, flexible and generic solving method.

A secondary contribution is the presentation and modeling of the Pulp Production Scheduling Problem, Chapter 2, a new real-world problem that aims at minimizing the variance of the basic densities of the wood chips used in the cooking process of pulp production. Minimizing the variance naturally leads to a non-linear problem, which were linearized by proposing the discretization of the range of the basic densities and the use of the slack variables. A solution approach based on a constructive heuristic was also proposed in this paper/chapter.

6.2 Future research directions

In this thesis we proposed a generic framework for hybrid solving approaches; our research suggested some interesting new research avenues.

The parallelization of the IRSS Framework is a special and challenging research avenue. Evidently, in a parallel environment the tree structure in which the IRSS Framework is implemented will not be the most efficient implementation. The simplest parallel implementation would be one processor running the Restriction Phase and several others executing the Search Phase in parallel. However, even in this simple implementation there are a number of unanswered questions. Such as, how would the different Search Phases communicate?

The comparison of the performance of the Framework against well established methods for classical problems is another interesting research avenue, which could lead to new implementation guidelines and assist in proving the performance and efficiency of the IRSS Framework.

As the Framework is independent of the search methods used in both Restriction and Search Phases an interesting research avenue would be to make a comparative study on the performance of the several heuristics and exact methods which could be used in both Restriction and Search phases. This could lead to a ranking of the most suitable methods that may be used for a class of problems.

Finally, there are number of challenging problems in both real-life and classical spheres which could be interesting to address with this approach in order to continue the validation the IRSS Framework.

Thesis Bibliography

Abdinnour-Helm, S. (1998). A hybrid heuristic for the uncapacitated hub location problem. *European Journal of Operational Research*, 106(2-3):489–499.

Ahuja, R. K., Ergun, O., Orlin, J. B., and Punnen, A. P. (2002). A survey of very large-scale neighborhood search techniques. *Discrete Applied Mathematics*, 123(1-3):75–102.

Ahuja, R. K., Liu, J., Orlin, J. B., Sharma, D., and Shughart, L. A. (2005). Solving real-life locomotive-scheduling problems. *Transportation Science*, 39(4):503–517.

Alvim, A. C. F., Ribeiro, C. C., and Glover, F. (2004). A hybrid improvement heuristic for the one-dimensional bin packing problem. *Journal of Heuristics*, 10(2):205–229.

Artigues, C. and Feillet, D. (2008). A branch and bound method for the job-shop problem with sequence-dependent setup times. *Annals of Operations Research*, 159(1):135–159.

Atkin, J. A. D., Burke, E. K., Greenwood, J. S., and Reeson, D. (2007). Hybrid metaheuristics to aid runway scheduling at london heathrow airport. *Transportation Science*, 41(1):90–106.

Barnhart, C., Hane, C. A., and Vance, P. H. (2000). Using branch-and-price-and-cut to solve origin-destination integer multicommodity flow problems. *Operations Research*, 48(2):318–326.

Benders, J. (1962). Partitioning procedures for solving mixed-variables programming problems. *Numerische Mathematik*, 4:238–252.

Bertels, S. and Fahle, T. (2006). A hybrid setup for a hybrid scenario: combining heuristics for the home health care problem. *Computers & Operations Research*, 33(10):2866–2890.

Biermann, C. J. (1996). *Handbook of Pulping and Papermaking*. Academic Press, second edition.

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

Boctor, F. F. (1993). Discrete optimization and multi-neighbourhood, local improvement heuristics. Technical report, Groupe de Recherche en gestion de la logistic : Université Laval.

Borraz-Sanchez, C. and Rios-Mercado, R. Z. (2005). A hybrid meta-heuristic approach for natural gas pipeline network optimization. In *Hybrid Metaheuristics, Proceedings*, volume 3636 of *Lecture Notes in Computer Science*, pages 54–65. Springer-Verlag Berlin, Germany.

Budenbender, K., Grunert, T., and Sebastian, H. J. (2000). A hybrid tabu search/branch-and-bound algorithm for the direct flight network design problem. *Transportation Science*, 34(4):364–380.

Carrano, E. G., Takahashi, R. H. C., Cardoso, E. P., Saldanha, R. R., and Neto, O. M. (2005). Optimal substation location and energy distribution network design using a hybrid ga-bfgs algorithm. *Iee Proceedings-Generation Transmission and Distribution*, 152(6):919–926.

Carter, M. W. and Lapierre, S. D. (1999). Scheduling emergency room physicians. Technical Report 99-23, Centre for Research on Transportation, Montreal, Canada.

Cavallet, A., Malucelli, F., and Wolfler Calvo, R. (2000). A non linear multicommodity network design approach to solve a location-allocation problem in freight transportation. In *ODYSSEUS 2000 Workshop in freight transportation and logistics*. Chania (Greece).

Correa, A. I., Langevin, A., and Rousseau, L. M. (2004). Dispatching and conflict-free routing of automated guided vehicles: A hybrid approach combining constraint programming and mixed integer programming. In *Integration of Ai and or Techniques in Constraint Programming for Combinatorial Optimization Problems*, volume 3011 of *Lecture Notes in Computer Science*, pages 370–379. Springer-Verlag Berlin, Berlin.

Costa, A. M. (2005). A survey on benders decomposition applied to fixed-charge network design problems. *Computers & Operations Research*, 32:1429–1450.

Crainic, T. G. and Gendreau, M. (2002). Cooperative parallel tabu search for capacitated network design. *Journal of Heuristics*, 8(6):601–627.

da Silva, C. G., Figueira, J., and Climaco, J. (2007). Integrating partial optimization with scatter search for solving bi-criteria 0,1-knapsack problems. *European Journal of Operational Research*, 177(3):1656–1677.

Danna, E., Rothberg, E., and Le Pape, C. (2005). Exploring relaxation induced neighborhoods to improve mip solutions. *Mathematical Programming*, 102(1):71–90.

Dantzig, G. B. and Wolfe, P. (1960). Decomposition principle for linear programs. *Operations Research*, 8(1):101–111.

Delmaire, H., Diaz, J. A., Fernandez, E., and Ortega, M. (1999). Reactive grasp and tabu search based heuristics for the single source capacitated plant location problem. *Infor*, 37(3):194–225.

Diaz, J. A. and Fernandez, E. (2005). Hybrid scatter search and path relinking for the capacitated p-median problem. *European Journal of Operational Research*, 169(2):570–585.

Doong, S. H., Lai, C. C., and Wu, C. H. (2007). Genetic subgradient method for solving location-allocation problems. *Applied Soft Computing*, 7(1):373–386.

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

Dorndorf, U., Pesch, E., and Phan-Huy, T. (2000a). A time-oriented branch-and-bound algorithm for resource-constrained project scheduling with generalised precedence constraints. *Management Science*, 46(10):1365–1384.

Dorndorf, U., Pesch, E., and Toan, P. H. (2000b). Constraint propagation techniques for the disjunctive scheduling problem. *Artificial Intelligence*, 122(1-2):189–240.

Duffy, G. G. and Kibblewhite, R. P. (1989). A new method of relating wood density, pulp quality and paper properties. *Appita Journal*, 42(3):209–214.

Feo, T., Resende, M., and Smith, S. (1994). A greedy randomized adaptive search procedure for maximum independent set. *Operations Research*, 42:860–878.

Fischetti, M., Glover, F., and Lodi, A. (2005). The feasibility pump. *Mathematical Programming*, 104(1):91–104.

Fischetti, M. and Lodi, A. (2003). Local branching. *Mathematical Programming*, Ser. B - 98:23 - 47.

Fisher, W. D. (1958). On grouping for maximum homogeneity. *Journal of American Statistical Association*, 53(284):789–798.

Foelkel, C. E. B., Mora, E., and Menochelli, S. (1992). Densidade básica: sua verdadeira utilidade como índice de qualidade da madeira de eucalipto para produção de celulose. *O Papel*, 53:35–40.

French, A. P., Robinson, A. C., and Wilson, J. M. (2001). Using a hybrid genetic-algorithm/branch and bound approach to solve feasibility and optimization integer programming problems. *Journal of Heuristics*, 7(6):551–564.

Gallardo, J., Cotta, C., and Fernandez, A. (2007). On the hybridization of memetic algorithms with branch-and-bound techniques. *IEEE Transactions On Systems Man and Cybernetics Part B-Cybernetics*, 37-1:77–83.

Gendron, B., Potvin, J.-Y., and Soriano, P. (2003). A parallel hybrid heuristic for the multicommodity capacitated location problem with balancing requirements. *Parallel Computing*, 19:591–606.

Geoffrion, A. M. (1970a). Elements of large-scale mathematical programming part i. *Management Science*, 18(11):652–675.

Geoffrion, A. M. (1970b). Elements of large-scale mathematical programming part ii. *Management Science*, 16(11):676–691.

Geoffrion, A. M. (1972). Generalized benders decomposition. *Journal of Optimization Theory and Applications*, 10(4):237–260.

Geoffrion, A. M. and Powers, R. F. (1995). 20 years of strategic distribution-system design - an evolutionary perspective. *Interfaces*, 25(5):105–127.

Gilmore, P. and Gomory, R. (1961). A linear programming approach to the cutting-stock problem. *Operations Research*, 9(6):849–859.

Glover, F. (1977). Heuristics for integer programming using surrogate constraints. *Decision Sciences*, 8:156–166.

Glover, F. and Laguna, M. (1997). *Tabu search.* Kluwer academic publishers.

Glover, F., Laguna, M., and Martí, R. (2000). Fundamentals os scatter search and path relinking. *Control and Cybernetics*, 29(3):653–684.

Glover, F. W. and Kochenberger, G. A., editors (2003). *Handbook of Metaheuristics.* Springer.

Gronkvist, M. (2006). Accelerating column generation for aircraft scheduling using constraint propagation. *Computers & Operations Research*, 33(10):2918–2934.

Hansen, P. and Mladenović, N. (2001). Variable neighbourhood search: Principles and applications. *Euroupean Journal of Operational Research*, 130:449–467.

Holland, J. (1975). *Adaptation in Natural and Artificial Systems.* MIT Press.

Hooker, J. N. (2006). An integrated method for planning and scheduling to minimize tardiness. *Constraints*, 11(2-3):139–157.

Jain, A. S. and Meeran, S. (2002). A multi-level hybrid framework applied to the general flow-shop scheduling problem. *Computers & Operations Research*, 29(13):1873–1901.

Johnson, M. R. and Wang, M. H. (1998). Economical evaluation of disassembly operations for recycling, remanufacturing and reuse. *International Journal of Production Research*, 36(12):3227–3252.

Karlsson, J., Ronnqvist, M., and Bergstrom, J. (2003). Short-term harvest planning including scheduling of harvest crews. *International Transactions in Operational Research*, 10:413–431.

Keedwell, E. and Khu, S. T. (2006). A novel evolutionary meta-heuristic for the multi-objective optimization of real-world water distribution networks. *Engineering Optimization*, 38(3):319–336.

Kennedy, J. F., Philips, G., and Williams, P., editors (1989). *Wood processing and utilization.* Ellis Horwood Limited.

Keskin, B. B. and Uster, H. (2007). A scatter search-based heuristic to locate capacitated transshipment points. *Computers & Operations Research*, 34(10):3112–3125.

Kibblewhite, R. P. and Uprichard, J. M. (1996). Kraft pulp qualities of eleven radiata pine clones. *Appita*, 49(4):243–250.

Kim, D. and Pardalos, P. M. (2000). Dynamic slope scaling and trust interval techniques for solving concave piecewise linear network flow problems. *Networks*, 35(3):216–222.

Kirkpatrick, S., Gelatt, C. D., and Vecchi, M. P. (1983). Optimization by simulated annealing. *Science*, 220(4598):671–680.

Klose, A. and Drexl, A. (2005). Facility location models for distribution system design. *European Journal of Operational Research*, 162(1):4–29.

Land, A. H. and Doig, A. G. (1960). An automatic method of solving discrete programming-problems. *Econometrica*, 28(3):497–520.

Langston, M. A. (1982). Improved 0/1-interchange scheduling. *Bit*, 22(3):282–290.

Lapierre, S. D., Ruiz, A. B., and Soriano, P. (2004). Designing distribution networks: Formulations and solution heuristic. *Transportation Science*, 38(2):174–187.

Lawler, E. (1976). *Combinatorial Optimization: Networks and Matroids.* Holt, Rinehart and Winston, USA.

Lorena, L. A. N. and Furtado, J. C. (2001). Constructive genetic algorithm for clustering problems. *Evolutionary Computation*, 9(3):309–327.

Lorena, L. A. N. and Senne, E. L. F. (2004). A column generation approach to capacitated p-median problems. *Computers & Operations Research*, 31(6):863–876.

Mackworth, A. K. (1977). Consistency in networks of relations. *Artificial Intelligence*, 8(1):99–118.

Manber, U. (1989). *Introduction to Algorithms - A Creative Approach.* Addison-Wesley Publishing Company.

Marin, A., Canovas, L., and Landete, M. (2006). New formulations for the uncapacitated multiple allocation hub location problem. *European Journal of Operational Research*, 172(1):274–292.

Molteberg, D. (2004). Methods for the determination of wood properties, kraft pulp yield and wood fibre dimensions on small wood samples. *Wood Science and Technology*, 37(5):395–410.

Murray, A. T. and Church, R. L. (1995). Measuring the efficacy of adjacency constraint structure in forest planning-models. *Canadian Journal of Forest Research-Revue Canadienne De Recherche Forestiere*, 25(9):1416–1424.

Murray, A. T. and Church, R. L. (1996a). Analyzing cliques for imposing adjacency restrictions in forest models. *Forest Science*, 42(2):166–175.

Murray, A. T. and Church, R. L. (1996b). Constructing and selecting adjacency constraints. *Infor*, 34(3):232–248.

Nemhauser, G. L. and Wolsey, L. A. (1988). *Integer and Combinatorial Optimization.* Wiley-Interscience.

Nguyen, H. D., Yoshihara, I., Yamamori, K., and Yasunaga, M. (2007). Implementation of an effective hybrid ga for large-scale traveling salesman problems. *IEEE Transactions on Systems, Man and Cybernetics, Part B*, 37(1):92–99.

Padberg, M. (2000). Approximating separable nonlinear functions via mixed zero-one programs. *Operations Research Letters*, 27(1):1–5.

Papadimitriou, C. and Steiglitz, K. (1982). *Combinatorial Optimization: Algorithms and Complexity.* Prentice-Hall.

Pécora, J. E. (2002). Alocação de médicos em salas de emergências - uma abordagem híbrida. Master's thesis, Campinas University, Campinas, Brasil.

Pécora, J. E., Ruiz, A., and Soriano, P. (2007a). High level hybrid method for the pulp production scheduling problem. *Working Paper.*

Pécora, J. E., Ruiz, A., and Soriano, P. (2007b). Minimization of the wood density variation in pulp and paper production. *INFOR*, 45(4):187–196.

Pécora, J. E., Ruiz, A. B., and Soriano, P. (2007c). High level hybrid method for the location-allocation problem. *Working Paper.*

Pendharkar, P. C. (2005). Hybrid approaches for classification under information acquisition cost constraint. *Decision Support Systems*, 41(1):228–241.

Peng, J., Shang, G., and Liu, H. (2006). A hybrid intelligent algorithm for vehicle routing models with fuzzy travel times. *Computational Intelligence, Pt 2, Proceedings*, 4114:965–976.

Perez, M., Almeida, F., and Moreno-Vega, J. M. (2005). A hybrid grasp-path relinking algorithm for the capacitated p-hub median problem. In *Hybrid Metaheuristics, Proceedings*, volume 3636 of *Lecture Notes in Computer Science*, pages 142–153. Springer-Verlag Berlin, Germany.

Potter, M. A. and De Jong, K. A. (1994). A cooperative coevolutionary approach to function optimization. In *The 3rd Parallel Problem Solving from Nature (PPSN), Proceedings of the Conference.*, pages 249–257. Springer-Verlag.

Prins, C., Prodhon, C., Ruiz, A., Soriano, P., and Wolfler Calvo, R. (2007). Solving the capacitated location-routing problem by a cooperative lagrangean relaxation-granular tabu search heuristic. *Transportation Science*, 41(4):470–483.

Puchinger, J. and Raidl, G. R. (2005). Combining metaheuristics and exact algorithms in combinatorial optimization: A survey and classification. In *Artificial Intelligence and Knowledge Engineering Applications: a Bioinspired Approach, Pt 2, Proceedings*, volume 3562 of *Lecture Notes in Computer Science*, pages 41–53. Springer-Verlag, Berlin.

Rei, W., Cordeau, J., Gendreau, M., , and Soriano, P. (2006). Accelerating benders decomposition by local branchings. Technical report, Center for Research on Transportation - C.P. 6128, succursale Centre-ville Montr´eal QC H3C 3J7 Canada.

Repoussis, P. P., Paraskevopoulos, D. C., Tarantilis, C. D., and Ioannou, G. (2006). A reactive greedy randomized variable neighborhood tabu search for the. vehicle routing problem with time windows. In *Hybrid Metaheuristics, Proceedings*, volume 4030 of *Lecture Notes in Computer Science*, pages 124–138. Springer-Verlag Berlin, Germany.

Resende, M. G. C. and Werneck, R. F. (2004). A hybrid heuristic for the p-median problem. *Journal of Heuristics*, 10(1):59–88.

Resende, M. G. C. and Werneck, R. F. (2006). A hybrid multistart heuristic for the uncapacitated facility location problem. *European Journal of Operational Research*, 174(1):54–68.

ReVelle, C. S. and Eiselt, H. A. (2005). Location analysis: A synthesis and survey - invited review. *European Journal of Operational Research*, 165(1):1–19.

ReVelle, C. S., Eiselt, H. A., and Daskin, M. S. (2008). A bibliography for some fundamental problem categories in discrete location science. *European Journal of Operational Research*, 184(3):817–848.

Ronnqvist, M. (2003). Optimization in forestry. *Mathematical Programming*, 97(1-2):267–284.

Rossi, F., van Beek, P., and Walsh, T., editors (2006). *Handbook of Constraint Programming*. Elsevier.

Sadykov, R. (2008). A branch-and-check algorithm for minimizing the weighted number of late jobs on a single machine with release dates. *European Journal of Operational Research*, 189(3):1284–1304.

Sahoo, B. and Maity, D. (2007). amage assessment of structures using hybrid neuro-genetic algorithm. *Applied Soft Computing*, 7-1:89–104.

Schimleck, L., Payne, P., and Wearne, R. (2005). Determination of important pulp properties of hybrid poplar by near infrared spectroscopy. *Wood Fiber and Science*, 37(3):462–471.

Schlottmann, F. and Seese, D. (2004). A hybrid heuristic approach to discrete multi-objective optimization of credit portfolios. *Computational Statistics & Data Analysis*, 47(2):373–399.

Schneeweiss, C. (2003). *Distributed Decision Making*. Springer, New York, 2 edition.

Slats, P. A., Bhola, B., Evers, J. J. M., and Dijkhuizen, G. (1995). Logistic chain modeling. *European Journal of Operational Research*, 87(1):1–20.

Talbi, E. G. (2002). A taxonomy of hybrid metaheuristics. *Journal of Heuristics*, 8(5):541–564.

Tarantilis, C. D., Zachariadis, E. E., and Kiranoudis, C. T. (2008). A hybrid guided local search for the vehicle-routing problem with intermediate replenishment facilities. *Informs Journal on Computing*, 20(1):154–168.

Toth, P. and Vigo, D. (2003). The granular tabu search and its application to the vehicle-routing problem. *Informs Journal on Computing*, 15(4):333–346.

Vaidyanathan, B., Ahuja, R. K., Liu, J., and Shughart, L. A. (2008). Real-life locomotive planning: New formulations and computational results. *Transportation Research Part B-Methodological*, 42(2):147–168.

Velarde, J. L. G. and Laguna, M. (2004). A benders-based heuristic for the robust capacitated international sourcing problem. *IIE Transactions*, 36(11):1125–1133.

Voβ, S., Martello, S., Osman, I., and Roucariol, C., editors (1999). *Metaheuristics - Advances and Trends in Local Search Paradigms for Optimization*. Kluwer Academic Publishers.

von Neumann, J. and Burks, A. W., editors (1966). *Theory of Self-Reproducing Automata*. University of Illinois Press.

Weintraub, A., Church, R. L., Murray, A. T., and Guignard, M. (2000). Forest management models and combinatorial algorithms: analysis of state of the art. *Annals of Operations Research*, 96:271–285.

Wiegand, P. (2003). *An Analysis of Cooperative Coevolutionary Algorithms*. PhD thesis, George Mason University.

Williams, M. F. (1994). Matching wood fiber characteristics to pulp and paper processes and products. *Tappi Journal*, 77(3):227–233.

Wolsey, L. A. (1998). *Integer Programming*. Wiley-Interscience.

Wu, L. Y., Zhang, X. S., and Zhang, J. L. (2006). Capacitated facility location problem with general setup cost. *Computers & Operations Research*, 33(5):1226–1241.

Yan, S. and Zhou, K. (2006). Three-tier multi-agent approach for solving traveling salesman problem. *PRICAI 2006: Trends In Artificial Intelligence, Proceedings*, 4099:813–817.

Appendix A

Formulation for the Location-Allocation Problem - Chapter 4

In this appendix the mathematical model for the location allocation problem described in Chapter4 is presented. This model, which based on the work of Cavallet et al. (2000) incorporates the inter-hub transportation and the piecewise linear functions for the hubs implementation costs and transportation to/from an international hub.

A.1 Variables

The variables are divided into three distinct groups: transportation variables which decides the path used by each commodity, hub allocation variables associated with the opening of each hub and piecewise variables related with the size of each hub/arc.

The transportation variables are binary and they represent the path that each commodity is transported. These variables are always symbolized by the letter x followed by a superscript second letter indicating the origin of the shipment, (I)nternal $\rightarrow x^I$ or (E)xternal $\rightarrow x^E$ and they are formally defined as:

- $x^I_{i,h,e} \in \mathbb{B}$ assuming the value 1 if commodity (i, e) passes through the hub h and 0 otherwise.

- $x^E_{e,h,i} \in \mathbb{B}$ assuming the value 1 if commodity (e, i) passes through the hub h and 0 otherwise.

For the transportation variables the special case, $h_1 = h_2$, means that the commodity uses only one hub.

The second set of variables, the hub allocation variables $y_h \in \mathbb{B}$, decides if a hub will be opened or not. They are binary and assume the values $y_h = 1$ if the hub h is active (open) and zero otherwise.

We have one set, $W \in \mathbb{R}^+$ and $Z \in \mathbb{B}$, of piecewise variables for hubs and another for the arcs $Hub \leftrightarrow External$ in each direction. They are named as $w_{h,l}$ and $z_{h,l}$ for the hubs and, following the same convention as the transportation variables, $w^I_{h,e,l}$, $z^I_{h,e,l}$ and $w^E_{h,e,l}$, $z^E_{h,e,l}$ for the arcs. The W variables are the passing flow through each hub/arc, the Z variables are the binary variable associated with the size, (i.e. the piece of the piecewise function).

A.2 Mathematical model

$$\begin{aligned}\text{Minimize Obj} = \sum_{i,h,e} \theta_{i,h}(d_{i,e} * x^I_{i,h,e} + d_{e,i} * x^E_{e,h,i}) + \\ \sum_h \sigma * y_h + \\ \sum_h f(w_{l,h}) + \sum_{h,e}[f(w^I_{l,e,h}) + f(w^E_{l,e,h})] \end{aligned} \tag{A.1}$$

$$\begin{aligned}
x^I_{i,h,e}, &\leq y_h \ \forall i, h, e \in J\text{x}H && \text{(A.2)}\\
x^E_{e,h,i}, &\leq y_h \ \forall e, h, i \in J\text{x}H && \text{(A.3)}\\
\sum_h x^I_{i,h,e} &= 1 \ \forall (i, e) \in J^1 && \text{(A.4)}\\
\sum_h x^E_{e,h,i} &= 1 \ \forall (e, i) \in J^2 && \text{(A.5)}\\
w_{h,l} &\geq (a_{h,l} - a_{h,l-1}) * z_{h,l}, l = 1, ..., n_1 - 1;\ h \in H && \text{(A.6)}\\
w_{h,l+1} &\leq (a^1_{h,l+1} - a^1_{h,l}) * z_{h,l},\ l = 1, ..., n_1 - 1;\ h \in H && \text{(A.7)}\\
w^I_{h,e,l} &\geq (a^1_{h,e,l} - a^1_{h,e,l-1}) * z^I_{h,e,l},\ l = 1, ..., n_2 - 1; (h, e) \in H\text{x}E && \text{(A.8)}\\
w^I_{h,e,l+1} &\leq (a^2_{h,e,l+1} - a^2_{h,e,l}) * z^I_{h,e,l},\ l = 1, ..., n_2 - 1; (h, e) \in H\text{x}E && \text{(A.9)}\\
w^E_{h,e,l} &\geq (a^2_{h,e,l} - a^2_{h,e,l-1}) * z^E_{h,e,l},\ l = 1, ..., n_2 - 1; (h, e) \in H\text{x}E && \text{(A.10)}\\
w^E_{h,e,l+1} &\leq (a^2_{h,e,l+1} - a^2_{h,e,l}) * z^E_{h,e,l}, l = 1, ..., n_2 - 1;\ (h, e) \in H\text{x}E && \text{(A.11)}
\end{aligned}$$

$$\begin{aligned}
\sum_{l=1}^{n_1} w_{h,l} &= \sum_{i,e \in J} d_{i,e} * x^I_{i,h,e} + d_{e,i} * x^E_{e,h,i} \ \forall h \in H && \text{(A.12)}\\
\sum_{l=1}^{n_2} w^I_{h,e,l} &= \sum_i d_{i,e} * x^I_{i,h,e} \ \forall h, e \in H\text{x}E && \text{(A.13)}\\
\sum_{l=1}^{n_2} w^E_{h,e,l} &= \sum_i d_{e,i} * x^E_{e,h,i} \ \forall h, e \in H\text{x}E && \text{(A.14)}
\end{aligned}$$

$$\begin{aligned}
x^I_{i,h_1,h_2,e}; x^E_{e,h_1,h_2,i}; y_h; z_{h,l}; z^I_{h,e,l}; z^E_{h,e,l} &\in \mathbb{B} && \text{(A.15)}\\
w_{h,l}; w^I_{h,e,l}; w^E_{h,e,l} &\in \mathbb{R}^+
\end{aligned}$$

In the objective function, Equation A.1, the costs θ, γ, are linear costs associated with the internal node - hub arcs and interhub transportation respectively, σ is hub allocation fixed cost, while the costs $f(w_{l,h})$, $f(w^I_{l,e,h})$ and $f(w^E_{l,e,h})$ refers to the piecewise costs of hub implementation and transportation through an international arc. The function $f(\vec{W})$ is defined as follows:

$$f(\vec{W}) = b_0 + \frac{b_1 - b_0}{a_1 - a_0} w_1 + \frac{b_2 - b_1}{a_2 - a_1} w_2 + ... + \frac{b_{|L|} - b_{|L|-1}}{a_{|L|} - a_{|L|-1}} w_{|L|} \qquad \text{(A.16)}$$

Where the points (a_l, b_l) are the breakpoints in which the piecewise function defined and $w_l \in [0, a_l - a_{l-1}]$ is the flow passing through each piece l. Constraints A.2 to A.3 impose transportation through open hubs only, Equations A.4 and A.5 assures the unicity of the path used to transport each commodity.

The Inequations A.6 to A.11 force the sequential use of each piece into the piecewise functions. Note that inequations A.6 and A.7 imply that $z_{h,1} \geq z_{h,2} \geq ... \geq z_{h,|L|}$, the same reasoning can be made for the other piecewise variables. A proof that this modelization for the piecewise function is locally ideal can be found In the work of Padberg (2000).

The Constraints A.12 to A.14 provide a link between the transportation and the piecewise variables, computing the total flow trough each piecewise entity. There n_1 and n_2 are the number of pieces of the piecewise function for hub implementation and Hub-External transportation costs respectively. Equation A.12 computes the total passing flow through each hub, while Equations A.13 and A.14 are dedicated to the flow through the arcs (I, E) and (E, I).

Finally, constraints A.15 are the non negativity constraints.

Appendix B

Embedded RS for the PPSP - Chapter 5

In this appendix a possible implementation of the Embedded RS is presented. This implementation follows the same structure used to implement the Embedded RS for the location-allocation problem in Chapter 4, using the objective function value of each solution to build a weighted frequency matrix.

B.1 Embedded RS

In order to implement the Embedded RSS procedure, a frequencies matrix is required (see Section 4.4). Since the structural variables chosen in the case of the PPSP were the basic density working level, represented by the variable $Z_{f,p,l} \in \mathbb{B}$, the definition of the frequency matrix is defined as:

$$M_{f,p,l} = C \sum_{s \in S} \frac{(1 - Z^s_{f,p,l})}{f(s)} \tag{B.1}$$

where C is a constant used to avoid floating point errors, $f(s)$ is the objective function of the solution $s \in S$ – the set of solutions used to generate the RS – and $Z^s_{f,p,l}$ is each structural variable belonging to solution s. Note that the PPSP is a minimization problem, the frequency matrix is weighted by the inverse of the objective value $1/f(s)$.

The variables belonging to each Embedded RS, RS^λ, are chosen in decreasing order of their values in the matrix M.

B.2 Tree structure

The tree structure used in this implementation is not different from IRSS one. However, to avoid cycling, a part of the current solution space must be cut after every search phase. This may be a problem when finding a non solvable (feasible or unknown) RS. In order to avoid this drawback the the rules defined in the Dynamic Embedded RS are used to identify a RS which is solvable in the alloted time.

"Strange! I don't understand how it is that we can write mathematical expressions and calculate what the thing is going to do without being able to picture it."

Richard Feynman (1918-1988)

" ...for better or worse, our future will be determined in large part by our dreams and by the struggle to make them real."

Mihaly Csikszentmihalyi (1934-)