Alma Mater Studiorum - Università di Bologna

DOTTORATO DI RICERCA IN

INFORMATICA

Ciclo XXVIII

Settore Concorsuale di afferenza: 01/B1

Settore Scientifico disciplinare: INF/01

MODEL-BASED HEURISTICS FOR COMBINATORIAL OPTIMIZATION

Presentata da: Elena Rocchi

Coordinatore Dottorato
Prof. Paolo Ciaccia

Relatore
Prof. Vittorio Maniezzo

Esame finale anno 2016

Abstract

Many problems arising in several and different areas of human knowledge share the characteristic of being intractable in real cases. The relevance of the solution of these problems, linked to their domain of action, has given birth to many frameworks of algorithms for solving them. Traditional solution paradigms are represented by exact and heuristic algorithms. In order to overcome limitations of both approaches and obtain better performances, tailored combinations of exact and heuristic methods have been studied, giving birth to a new paradigm for solving hard combinatorial optimization problems, constituted by model-based metaheuristics. In the present thesis, we deepen the issue of model-based metaheuristics, and present some methods, belonging to this class, applied to the solution of combinatorial optimization problems.

Acknowledgements

I want to express my thanks to my thesis supervisor, Professor Vittorio Maniezzo, for his guidance during the years of my PhD program.

I also want to thank Dr. Marco Antonio Boschetti for his support and his advices.

I wish to thank Professor Thomas Stützle for his guidance during my six-months period abroad spent in the laboratory IRIDIA, at the Universitè Libre de Bruxelles (ULB).

Lastly, I want to thank Dr. Francesco Strappaveccia.

Contents

List of Figures

List of Tables

LIST OF TABLES

List of abbreviations

ACO	Ant Colony Optimization
ALNS	Adaptive Large Neighborhood Search
B&B	Branch-and-Bound
B&C	Branch-and-Cut
B&C&P	Branch-and-Cut-and-Price
B&P	Branch-and-Price
CG	Column Generation
CM	Corridor Method
COP	Combinatorial Optimization Problem
CP	Cutting Plane
CVRP	Capacitated Vehicle Routing Problem
DP	Dynamic Programming
EA	Evolutionary Algorithms
FLP	Facility Location Problem
FSP	Flow Shop Problem
GA	Genetic Algorithms
GRASP	Greedy Randomized Adaptive Search Procedure

IG	Iterated Greedy
ILP	Integer Linear Programming
ILS	Iterated Local Search
IP	Integer Programming
JSSP	Job Shop Scheduling Problem
LB	Local Branching
LNS	Large Neighborhood Search
LP	Linear Programming
MILP	Mixed Integer Linear Programming
MIP	Mixed Integer Programming
MP	Mathematical Programming
MSTP	Minimum Spanning Tree Problem
OP	Orienteering Problem
QAP	Quadratic Assignment Problem
RINS	Relaxation Induced Neighborhood Search
SA	Simulated Annealing
SERR	Selection, Extraction, Recombination and Reallocation
SC	Set Covering
SP	Set Partitioning
TS	Tabu Search
TSP	Traveling Salesman Problem
VILS	Variable Intensity Local Search

VLNS	Very Large Neighborhood Search
VND	Variable Neighborhood Descent
VNS	Variable Neighborhood Search
VRP	Vehicle Routing Problem
VRPTW	VRP with Time Windows

1

Introduction

Let us start our exposition by considering the following three problems:

- a multinational company has to open new warehouses in a region, in order to supply factories present in that region. Several sites are eligible to become a warehouse. The company has to decide how many and in which sites warehouses will be opened, in order to serve all factories with the smallest operating costs associated;

- at the end of each month, in a ward of a hospital, shifts of nurses are established for the following month. Since the timetable of nurses is not always sufficient for covering the minimum of personnel presence in every shift, freelance nursing is employed. The identification of the best assignment of shifts to nurses, in order to reduce costs linked to the outsourced work, must be done;

- given an unknown DNA fragment, such a fragment must be sequenced, i.e. it is necessary to find the order in which sequences of nucleotides appear in the unknown DNA fragment. The sequencing by hybridization method, (Pevzner and Lipshutz (169)), is one of the existing methods for DNA sequencing. If no experimental errors occur during hybridization, reconstructing the DNA sequence can be done in polynomial time, (Pevzner (168)). If experimental errors occur during hybridization, the reconstruction becomes a difficult problem, (see Caserta and Voß (61), Caserta and Voß (63) and Blum et al. (47) for some solution methods for this problem).

In spite of their different areas of knowledge and domain, the three presented problems have a common issue: these problems are all very difficult to solve in practice on nontrivial instances. They represent examples of hard COPs. Many other examples of hard COPs can be brought to the attention, each of them emerging from various areas of expertise. All of them share the peculiarity of being difficult to treat, since the nature of all these problems requires to treat NP-hard subproblems; this condition is independent of the particular scope of the problem itself.

Several approaches have been developed and studied for the solution of COPs; they can be classified in two general categories: exact and heuristic methods.

Exact methods take their origins from the field of Operations Research. They guarantee to find an optimal solution for COPs and to prove its optimality for every instance of the considered problem; however, because of the high increase of run time with respect to the size of instances to be solved, the applicability of these methods is often limited to small-size instances or instances having particular properties known a-priori. On the other side, heuristic algorithms adopt an opposed approach with respect to exact methods. Knowing the intrinsic hardness in solving real world instances of COPs, heuristics renounce to the aim of identifying an optimal solution, focusing on determining feasible solutions, having a "good" quality and computable in a reasonable time.

Both the exact and heuristic approaches have their strengths and weaknesses. The exact methods guarantee to identify the optimal solution of a COP, but real-world instances have, often, a prohibitive size, that makes them intractable for these methods, because of the high increase of run time. The heuristics do not guarantee to find the optimal solution, but they are able to identify good quality feasible solutions for the treated COP, in a reasonable run time. Current research within the field of the solution of hard COPs is more and more concentrating its efforts on the integration of exact and heuristic methods, with the aim of exploiting the strengths of both the approaches, in such a way to cancel the weaknesses of both. The term *model-based metaheuristics* comes from this idea, i.e. the idea of designing heuristic methods aware of mathematical programming features, able to exploit these elements to improve the state of the art for the treated problems.

The developed thesis is situated in the context of the model-based metaheuristics, treating the application and the design of model-based metaheuristics for solving hard

COPs, both from a theoretical and practical point of view. The present thesis is structured in five main chapters. In chapter 2 we made a survey of the state of the art of model-based heuristics, presenting a classification of these approaches and reporting several works from the literature in the field. In chapter 3 we propose a Lagrangean column generation heuristic for solving the CVRP; the heuristic is able to produce both a valid lower bound and a feasible solution for the treated CVRP instances, hence providing a procedure to preliminarily evaluate the quality of the identified feasible solutions. In chapter 4 we study the parameter tuning problem, applied both to a Lagrangean heuristic and to an ILS, for solving, respectively, instances of CVRP, VRPTW and QAP. In the chapter we define the problem of identifying the best tuning of optimization methods, introducing some algorithms designed to treat this kind of problem. We, then, introduce the Lagrangean heuristic we developed to solve both the CVRP and the VRPTW, detailing the problem of its tuning to obtain better quality valid lower bounds for the treated instances. We, then, introduce the ILS paradigm, detailing the problem of its tuning to find the best quality possible solutions for instances of QAP. In chapter 5 we treat a real-world problem, encountered in the context of the management of a warehouse of tiles, located in Thailand. The treated problem asks to build a model, able to predict the duration of the queues of work for the resources operating within the warehouse. We designed a heuristic method, able to simulate the daily working of the resources of the warehouse, with the aim of producing a scenario in which the duration of the queues of work is minimized, through the maximization of the use of all available resources of the warehouse. In chapter 6 the conclusions of the exposed thesis can be found. After the references to the related works from the literature, appendices A.1 and B.1 can be found, in which the detailed computational results related to the designed approaches are reported.

2

Model-based Heuristics: state of the art

2.1 The context

The combination of heuristics and exact methods in optimization has a very long history; this is not a recent phenomenon. There been no separation of heuristics researchers from exact methods researchers either. One of the classic examples is mixed-integer linear programming, in which most algorithms are essentially some combinations of heuristics (e.g. for branching variable selection, for node selection etc.) within an overall exact branching structure that guarantees to find the optimum solution. However, during the last years, the combination of exact and heuristic methods has gone towards a deep exploitation of features derived from the mathematical model of the problem to be solved; this has led to the use of algorithms originally designed for exactly solving problems within heuristic contexts. Boschetti et al. (54) introduce a new word, named *matheuristics* to define a relatively new class of optimization algorithms combining metaheuristics with MP techniques. Sometimes also the term *hybrid metaheuristics* is used to identify matheuristics: in fact, they represent to all effects hybrids of exact and metaheuristic algorithms. But it is necessary to point out that hybrid metaheuristics encompass a wider set of methods, counting also other non-pure metaheuristic approaches, like combinations of metaheuristics with other metaheuristics.

During the last years, several approaches that can be classified as matheuristics have been proposed to solve optimization problems; a huge variety of applications character-

izes the action of these methods. Many scientific papers and international workshops have been devoted to this topic. Contributions given to this area are manifold and can be classified in several manners, depending on the point of view adopted in analyzing interactions between MP and heuristic components. We classify matheuristics in three main classes, as shown in figure 2.1.

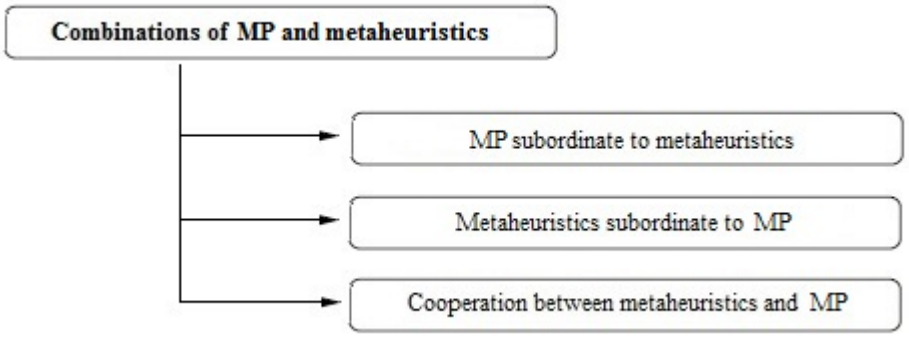

Figure 2.1: Classification of matheuristics

The structure of this chapter reflects the analysis of this classification; section 2.2 reviews approaches in which MP techniques are subordinate to metaheuristics, section 2.3 reviews methods in which metaheuristics are dependent on MP, while section 2.4 reviews collaboration approaches between MP and metaheuristics. Interesting collections about the topic of matheuristics have been also presented by Blum et al. (48), Boschetti and Maniezzo (53), Puchinger and Raidl (180), Raidl and Puchinger (184).

2.2 MP subordinate to metaheuristics

A huge part of the contributions given to the field of matheuristics consists in using MP, especially MIP components, for strengthening metaheuristic frameworks. In this context the contribution of MP is fundamental for the functioning of metaheuristics, because several elements of metaheuristics, (e.g. the definition of neighborhoods, the solution of emerging subproblems or the definition of whole heuristic frameworks) rely on MP components. We identified several research directions in this field. Following subsections review each one of these directions, discussing some examples from the literature. Interesting reviews about the integration of MP into heuristics can be found in Dumitrescu and Stützle (102, 103).

2.2.1 MP for neighborhood definition

This subsection presents some examples of matheuristics that use MP components to deal with the definition of neighborhoods. Local search is a very important part of heuristic methods, because it can improve the quality of feasible solutions; it relies on the exploration of a neighborhood; the larger the neighborhood the more the chances of finding improving good quality feasible solutions. The basic idea of using MP techniques for neighborhoods is that of defining large neighborhoods and use MP methods to explore them. In many cases, e.g. for MIP, this means defining neighborhoods that can be represented as MIP models and exploring them using a generic MIP solver.

We can identify two mainstreams in this field: some approaches mainly rely on well-known metaheuristics and use MP components to explore a particular customized large neighborhood. Other methods, instead, can be defined as *new* local searches, in that the MP contribution is the base for their internal functioning, giving birth to new local search approaches. In the following five subsections we report new local search methods, i.e. Very Large Neighborhood Search, Dynasearch, Local Branching, Corridor Method and variable fixing-based algorithms. In the last subsection we show some works from the literature that mainly rely on metaheuristics, (e.g. Tabu Search), and use exact components for exploring a customized neighborhood.

2.2.1.1 Very Large Neighborhood Search

Ahuja et al. (12) introduced VLNS as a paradigm for performing local search procedures, in which MIP techniques are used to define and explore neighborhoods. Local search algorithms produce better solutions when they explore large neighborhoods; but, because of the necessary explicit enumeration of neighboring solutions, the exploration of the whole large neighborhood can be very time consuming and, hence, the time to get a local optimum can be very long. The larger the neighborhood, the higher the time to explore it. To overcome this situation, VLNS avoids the explicit exploration of neighboring solutions, (that represents a waste of time), defining nonetheless neighborhoods whose size grows exponentially with the problem dimension, (because this permits to obtain better local optima); the explicit exploration of the neighborhood is possible if the neighborhood exploration can be defined as a combinatorial optimization problem itself.

An approach developing a VLNS has been presented by De Franceschi et al. (85) for solving a Distance-Constrained CVRP. The proposed method is an elaboration of a refinement procedure proposed by Sarvanov and Doroshko (194). In their work Sarvanov and Doroshko (194) defined the neighborhood of a given solution by all possible ways a set of removed customers can be reinserted in the solution itself. De Franceschi et al. (85) developed a so called SERR algorithm. This is a local search algorithm in which the neighborhood is defined by extracting a subset of customers, recombining the order of visit through the creation of new sequences and reallocating them in the partial solution to form a feasible, and hopefully, better solution. The reallocation problem has been modeled by the authors as a set partitioning problem, in which it is necessary to assign each sequence of customers to one feasible insertion point. CPLEX optimizer has been used to find an almost-optimal integer solution for the reallocation model. Computational results showed that SERR is able to improve results for instances of Distance-Constrained CVRP in 22 out of 32 cases w.r.t. results reported by Gendreau et al. (116). A similar approach in the context of the Open VRP has been presented by Salari et al. (192). Hewitt et al. (131) proposed an IP-based local search scheme for solving a Capacitated Fixed Charge Network Flow Problem, where IP is used to search large neighborhoods; the neighborhood is defined according to the arc-based formulation of the problem by choosing a subset of variables in the formulation, fixing the value of all remaining variables and solving the resulting restricted model with an IP solver. Computational results compared the proposed method against other heuristics for the same problem, like the cycle-based TS by Ghamlouche et al. (117) and path-relinking by Ghamlouche et al. (118); results showed the better performances obtained by the proposed IP-based local search method. Other promising results for large neighborhood search have been presented by Copado-Méndez et al. (77), where faced problems dealt with supply chain management. The large neighborhood here defined consists in fixing a subset of variables in the mathematical model of the problem and solving the resulting model. Computational results of the large neighborhood have been compared against the B&C algorithm of CPLEX; results showed that large neighborhood is able to produce near optimal solutions in a fraction of time w.r.t. the B&C method of CPLEX, and also its capability of identifying feasible solutions even in those cases in which CPLEX fails to converge. Chiarandini et al. (67) presented a DP algorithm for exploring a large neighborhood defined for treating the Graph Colouring Problem (see

Allen et al. (14), Barnier and Brisset (34), Lewandowski and Condon (146)); the neighborhood is called cyclic exchange and is composed by all solutions that can be obtained by changing colors to elements in a cyclic manner. Computational results executed on some instances from (4) demonstrated the better quality of final solutions obtained by the large neighborhood w.r.t. other local search procedures. Pirkwieser and Raidl (172) proposed a VNS (Hansen and Mladenović (128), Mladenović and Hansen (160)) integrating several large neighborhoods for treating a location routing problem. Numerical results executed on instances from (10) showed the effectiveness of the enhanced VNS w.r.t. the state of the art. Another interesting application of VLNS has been presented by Manerba and Mansini (154) for solving a supplier selection problem. The authors present a MILP local search method for solving this problem. Promising results of the proposed method against a greedy procedure by Manerba and Mansini (153) were obtained in treating a set of real-world instances: the average gap of solutions obtained by the MILP local search method w.r.t. the optimal solution of such real-world instances corresponds to 0.09%, while the average gap of the greedy heuristic is larger than 36%. Other interesting large neighborhood approaches have been presented by Ahuja et al. (13), Roli et al. (190), Santos et al. (193), Schmid et al. (197).

2.2.1.2 Dynasearch

Another paradigm for performing local search is represented by Dynasearch (Congram et al. (75)). Dynasearch wants to overcome defects of traditional local search methods, (i.e. entrapment in local optima), by allowing several local search moves of a certain type to be made in a single iteration; this permits to define a larger neighborhood, that will be explored in polynomial time through dynamic programming algorithms with the aim of identifying the best sequence of moves to be performed. Although the idea of exploring exponential size neighborhoods was not new, (see Carlier and Villon (59), Sarvanov and Doroshko (195)), the authors underlined how Dynasearch is innovative, since the other methods that use dynamic programming for searching large neighborhoods, (e.g. Balas (27), Balas and Simonetti (28), Carlier and Villon (59), Sarvanov and Doroshko (195)), do not explore them by considering multiple moves in one iteration. The authors proposed a computational study of Dynasearch for solving the Single-Machine Total Weighted Tardiness Scheduling Problem, where Dynasearch uses a swap neighborhood; the recursion of dynamic programming defines a set of states (k,

σ), each one representing the minimum total weighted tardiness for partial job sequence $\sigma(1),...,\sigma(k)$. Computational results executed on instances from Crauwels et al. (80) revealed how Dynasearch performs better in terms of solution quality w.r.t. other local search heuristics, one of which is a TS proposed by Crauwels et al. (80). Grosso et al. (125) presented an enhanced Dynasearch neighborhood for the same problem tackled by Congram et al. (75); the new neighborhood is based on the work of Congram et al. (75) and includes the usage of other operators for the exploration. Computational results showed the effectiveness of the enhanced neighborhood w.r.t. results presented by Congram et al. (75).

2.2.1.3 Local Branching

Like VLNS and Dynasearch, LB represents a general framework for performing local search; it was introduced by Fischetti and Lodi (110). LB is a local search method in which the neighborhood of an incumbent solution is defined through the introduction of linear inequalities in the mathematical model of the problem to be solved; the inequalities are called local branching cuts. These cuts represent soft fixing constraints for the variables of the model, imposing that only a predefined number of variables can change their value; the neighborhood defined in this way corresponds to a k-opt neighborhood, that has to be solved to optimality through the use of a MIP solver, e.g. CPLEX. The solver is used to optimize the original mathematical model of the problem enriched by local branching cuts, exploring, in this way, the defined neighborhood. The nature of LB is exact, even if it is designed to improve the heuristic behavior of MIP solvers. The introduction of branching cuts defines a tree of mathematical models; at each node of this tree, a MIP solver is used to explore the neighborhood defined at the node and find a possible improving incumbent solution. The aim of LB is that of favoring early updatings of incumbent solutions, producing in this way improved solutions at early stages of the computation. Given its exact nature, the qualities of LB have been assessed against the usage of CPLEX optimizer. Computational results executed on 7 instances from MIPLIB 3.0 and 22 hard instances from several authors showed that LB obtained better results in 23 out of 29 instances, demonstrating its effectiveness as general-purpose heuristic for MIPs. A successful application of LB has been presented by Rodríguez-Martín and José Salazar-González (189), where the authors used LB technique to solve a Capacitated Fixed-Charge Network Design Problem.

Computational results compared LB heuristic against a cycle-based TS by Ghamlouche et al. (117), a path relinking procedure by Ghamlouche et al. (118) and a multilevel cooperative TS by Crainic et al. (79); the comparison showed better performances for the proposed LB approach. A variant of VNS relying on LB has been presented by Fischetti et al. (112) for a FLP; the algorithm, called Diversification, Refining and Tight-refining, proved to be particularly appropriate for MIPs in which the set of binary variables can be separated in two subsets (levels), where, if first level variables are fixed, an easier subproblem is produced with the second-level variables. The refining phase consists in almost fixing the first-level variables to their value in the current solution; this is realized through the addition of a branching cut to the current MIP; a MIP solver is then used to optimize the model. If the model is not solved to proven optimality, the tight-refining phase comes, in which also the second-level variables are limited in their variations by the addition of other branching cuts to the model. The diversification phase is responsible for diversifying the search through the addition of another branching cut related to first-level variables. Computational results showed very good performances for the presented approach w.r.t. other heuristics proposed for solving the same problem. Liberti et al. (147) presented a method combining LB and VNS with B&B and sequential quadratic programming to obtain an algorithm called RECIPE for general mixed integer nonlinear programming. Other applications of LB can be found in the following papers by Acuna-Agost et al. (11), Hansen et al. (129). For a review of LB we refer to the work by Fischetti et al. (113).

2.2.1.4 Corridor Method

Another framework for dealing with the problem of defining effective neighborhoods is represented by the CM, proposed by Sniedovich and Voß (199). This is a local search method in which the structure of the neighborhood is defined according to the method M that will be used to explore it, be it a MIP solver, DP, etc. The method M can effectively explore the neighborhood; this is achieved through the addition of exogenous constraints on the original problem, that define a sort of corridor around an incumbent solution. The corridor identifies the neighborhood and the employed solver is forced to move along it. Caserta et al. (65) proposed an application of the CM to address a blocks relocation problem. An initial collection of stacks of blocks and a pickup sequence for blocks are given; blocks have to be picked up following the given sequence.

If there are other blocks above the block that has to be picked up, a pickup operation involves the relocation of these blocks in other stacks. The blocks relocation problem requires to find the relocation pattern for each pickup operation such that the number of future relocation moves for blocks is minimized. The authors define a DP recursion that identifies all possible relocation configurations that can be generated following the known pickup sequence. Because of the exponential growth of the number of possible states, the CM is applied to the DP recursion; the corridor imposes limitations in the number of stacks to be considered when relocating a block and in the maximum number of blocks per stack. The proposed method has been compared with the code developed by Kim and Hong (138) for the same problem; computational results were given for two sets of random generated instances, small-medium and large-scale instances. Results showed that the proposed CM approach is effective in finding optimal solutions in short computational time, (for small-medium size instances), and in improving the quality of solutions, (for large-scale instances). Applications of the CM for treating the problem of DNA sequencing have been proposed by Caserta and Voß (61) and Caserta and Voß (63); in both these papers, the problem is modeled as an OP (210). The peculiarity of CM implemented by Caserta and Voß (61) is the capability of adapting the width of the corridor basing on the presence of improving solutions in the examined neighborhood: if an improving solution is found in the neighborhood, the incumbent solution is updated and a new corridor is defined around this new solution. Otherwise, the width of the corridor is widened, in the hope of finding improving solutions. The corridor is formally defined by the addition of an inequality to the mathematical model of the OP, with a parameter that defines the width of the corridor. This method has been tested on a benchmark of 320 instances from Blazewicz et al. (46), demonstrating of being able to find optimal or near-optimal solutions for all instances w.r.t. the state of the art. Other applications of the CM have been presented by Caserta et al. (64) and Caserta and Voß (60).

2.2.1.5 Variable fixing frameworks

In previous subsections, we examined local search frameworks that define large neighborhoods as optimization problems themselves, and solve neighborhood exploration using exact methods. Often the definition of the neighborhood relies on *soft fixing* of variables, i.e. the local search framework forces some variables to change their value

without specifying what variables must do so, (e.g. LB). There are also some approaches that make a *hard fixing* of variables, i.e. the method identifies specific variables that are forced to change their value. In this subsection, some examples of local search frameworks using hard variable fixing will be summarized.

Danna et al. (82) introduced a new local search method, called RINS. As its name suggests, the structure of the neighborhood defined by RINS is induced by information contained in the continuous relaxation of the MIP model of the problem. If we consider a generic node of a B&C tree, it is possible to have two important information: the solution of the continuous relaxation at that node and the corresponding incumbent feasible solution. The basic idea of RINS is that of fixing values for variables that have the same ones both in the incumbent and in the relaxed solution while optimizing on the remaining variables; this variable fixing procedure defines the neighborhood, that will be explored through a MIP solver. Computational results to assess qualities of RINS approach have been executed by considering LB, modified w.r.t. Fischetti and Lodi (110) to work in the same way as RINS, i.e. as a heuristic within a MIP tree. Results obtained on difficult MIP models showed that RINS outperforms LB both in generating good feasible solutions and in faster solving the neighborhood exploration. A pre-processing technique for RINS has been proposed by Gomes et al. (124); it consists in searching for the ideal number of variables to be fixed for producing subproblems of controlled size.

Another framework in which neighborhood definition is based on hard variable fixing procedures is VILS; the method has been presented by Mitrović-Minić and Punnen (159) and it represents a local search framework for solving MIPs. The neighborhood of a given solution x is defined according to a so called *binding set*; this is a subset of the whole set of variables, in which the value of each variable is fixed to the corresponding value in the current solution. The definition of the binding set corresponds to the definition of the neighborhood. The exploration of the neighborhood is made by a generic MIP solver that optimizes on the remaining non-fixed variables. Hence, the general framework of VILS consists in varying in a tailored way the neighborhood, i.e. varying the dimension of the binding set, in such a way that it is possible to change neighborhood dimension. Initially, the size of the binding set is large with small search times for the MIP component; successively, VILS permits to intensify the local search

by decreasing the size of the binding set and increasing the time for MIP to explore the neighborhood.

A general heuristic framework for solving 0-1 MIPs has been proposed by Lazić et al. (142). The method is a two-level heuristic; the first level is based upon a VNS-like algorithm, in which hard fixing of variables is made to define the neighborhood. After having optimized on the remaining variables, if an improving solution is found, the second-level of optimization takes place through a VND procedure, that adds new constraints to the formulation of the problem to explore only some parts of the solution space. The proposed heuristic has been tested on the same instances used by Fischetti and Lodi (110). The method has been compared against the following algorithms: Variable Neighborhood Search Branching, (Hansen et al. (129)), LB, (Fischetti and Lodi (110)), RINS and the usage of CPLEX MIP solver. Computational results showed that the proposed framework is very competitive with other algorithms; it is able to improve solutions in 8 cases out of 29. Moreover, the proposed algorithm is able to reach the best solution results among all the other methods in 16 out of 29 cases, whereas the RINS heuristic obtains the best result in 12 cases, Variable Neighborhood Search Branching in 10 cases, CPLEX alone in 6 and LB in 2 cases. An extension of the approach proposed by Lazić et al. (142) can be found in the paper of Maraš et al. (156), where the applicability of the 0-1 heuristic is extended to general integer variables. Another example of variable fixing heuristic can be found in the work proposed by Perboli et al. (167) applied for solving a Two-Echelon CVRP.

2.2.1.6 MP for improving local search in metaheuristics

In this subsection, metaheuristics using MP for ameliorating local search procedures will be reviewed. Table 2.1 summarizes some of these approaches; for each paper, the name of its authors, the tackled problem and the adopted hybrid solution method are identified.

Many heuristics like TS, (Glover and Laguna (121)), VNS and GAs (Reeves (185)) gain benefits from MP techniques in performing local search, both in terms of quality of obtained solutions and in terms of computational efficiency.

For example, Yaghini et al. (215) presented a neighborhood based on a CP procedure combined with TS for solving a capacitated p-median problem. The neighborhood of a given solution is defined by the closure of an open median. Its exploration is made by

solving the Linear Programming (LP) model generated from the original one by relaxing integer constraints; CP inequalities are added to the relaxed model to strengthen it. The solution of this strengthened LP is considered as the best neighboring solution. The proposed method has been compared against the B&P by Ceselli and Righini (66); results showed that the proposed TS enhanced by the CP neighborhood obtains better performances w.r.t. Ceselli and Righini (66) in terms of solution quality. Ngueveu et al. (162) proposed to use the solution of a b-matching problem, (Edmonds (104)), for defining the candidate list of a TS method addressed to solve an m-peripatetic VRP; the list is composed by the set of unused edges that are in the solution of b-matching. Computational results executed on classic instances of VRP and TSP showed that the hybridized TS performs better than simple TS.

Hu et al. (134) presented a hybridization of VNS and ILP to solve the Generalized MSTP. VNS makes use of a VND procedure that combines three different neighborhoods; one of these is a Global Edge Exchange neighborhood, that makes use of a DP procedure to explore the corresponding neighborhood. Another one is called Global Subtree Optimization neighborhood and it is explored via MIP. Several computational tests have been performed to assess the quality of the proposed VNS method. Comparisons were made considering approaches proposed by Ghosh (119), Golden et al. (122), Pop (178); results showed that the presented VNS has performances equal or significantly better w.r.t. other heuristics. Comparisons among the different neighborhoods underlined how the DP component outperforms the other ones. Other works adopting VNS approaches enriched by MIP to strengthen local search procedures have been presented by Prandtstetter and Raidl (179), Strodl et al. (203), Walla et al. (214).

In the context of GAs and EAs, one of the most interesting investigated issues has been the exact recombination of parents for finding the best possible offspring. This problem represents to GAs and EAs what the problem of searching a large neighborhood means to the previous seen heuristics, because, given a population of solutions, the heuristic has to move to a new, possibly better, population using a recombination operator. The topic of generating the best possible offspring given two parent solutions represented by binary encoding has been theoretically treated by Eremeev (106); here the author presents some polynomial and NP-hard cases of recombination problems. Cotta and Troya (78) introduced the concept of dynastically optimal recombination, that deals with the usage of the problem knowledge to identify the best combination

of the features of the ancestors; a B&B method is used within an EA as operator to explore the potential of recombined solutions. Several MIP-based recombination operators have been presented in the literature. Borisovsky et al. (49) proposed a MIP-based recombination operator integrated within a GA for solving a supply management problem; the operator has the aim of finding the best possible combination of two given parent genotypes, simulating the mutation process with the addition of a randomness element; the recombination problem is modeled as a MIP problem in which all variables with zero value in both parent genotypes are fixed, except for a random subset of such variables. Computational results reported for the tackled problem indicated the validity of the proposed recombination operator w.r.t. the greedy-based GA reported by the authors in the same paper. Dolgui et al. (95) implemented a similar MIP-recombination operator for solving a problem of balancing transfer lines with multi-spindle machines. We mention another example of integration of an exact method within a GA proposed by Flushing and Di Caro (114); here the problem of a relay placement for wireless sensor networks has been tackled by a decomposition process, in which, at the top level, a GA finds possible relay placements and a MILP solver, at the bottom level, computes the optimal flow routing for the considered relay placement.

Other examples of hybrids between exact methods and heuristics can be found in the context of ILS (Lourenço et al. (152)). Lopes et al. (149) treated a machine reassignment problem. This problem was proposed in the Google ROADEF/EURO Challenge (2012), requiring to find an alternative reassignment of processes to machines that optimizes the usage of machine resources w.r.t. the initial given assignment. The authors implemented different versions of ILS; in particular, two of these versions involved the usage of an IP component to perform the perturbation phase of ILS. Computational experiments conducted on the proposed Google ROADEF/EURO Challenge (2012) instances showed the superiority of the IP based perturbations w.r.t. other ILS approaches and the high degree of competitiveness against other heuristics in the literature, e.g. the one proposed by Masson et al. (157). Other examples of hybrid ILS methods can be found in papers by Duarte et al. (98), Umetani et al. (212) for solving, respectively, a referee assignment problem, (Duarte et al. (99)) and a cutting stock problem.

ACO (Dorigo et al. (96)) framework has been hybridized with MP. An example from the literature can be found in the paper by Reimann (187), in which the author

proposed an ACO method for solving a Symmetric TSP, in which the visibility between all pairs of customers is defined using information derived by the calculation of the MSTP. MSTP and visibility information are computed before the beginning of ACO; the structural information of the presence of arcs in MSTP is used for calculating the attractiveness of each arc. Computational tests have been executed on some instances from TSPLIB; comparisons of the proposed method against two algorithms presented by Le Louarn et al. (143) showed that using MSTP information permits to obtain solutions of better quality.

Other works from the literature hybridizing exact methods and heuristics have been presented by Fernandes and Lourenço (108) and Cabrera G et al. (57). The authors of the first paper tackled a JSSP integrating a B&B method within a GRASP, (Feo and Resende (107)), to solve the scheduling related to one machine. The second paper deals with a Capacitated FLP, where a hybrid Artificial Bee Algorithm (Pham et al. (170)) is used for solving it; the usage of a MIP solver is integrated within the Bee Algorithm for providing the cost of each bee.

Table 2.1: Summary of hybrids subordinating MP to metaheuristics

Paper	Application	Solution approach
Yaghini et al. (215)	capacitated p-median problem	TS + CP
Ngueveu et al. (162)	m-peripatetic VRP	TS + perfect b-matching
Hu et al. (134)	generalized MSTP	VNS + MIP
Strodl et al. (203)	2-dimensional loading VRP	VNS + MIP
Walla et al. (214)	video-on-demand balancing problem	VNS + MIP
Prandtstetter and Raidl (179)	car sequencing problem	VNS + ILP
Borisovsky et al. (49)	supply management problem	GA + MIP
Dolgui et al. (95)	transfer line balancing problem	GA + MIP
Flushing and Di Caro (114)	relay node placement problem	GA + MILP
Lopes et al. (149)	machine reassignment problem	ILS + IP
Duarte et al. (98)	referee assignment problem	ILS + MIP
Reimann (187)	TSP	ACO + MSTP
Fernandes and Lourenço (108)	job shop scheduling problem	GRASP + MIP
Cabrera G et al. (57)	capacitated FLP	Bee Algorithm + MIP

2.2.2 MP for generating new heuristics

In the previous subsections examples of matheuristics that use MP components to deal with the definition of neighborhoods were discussed. In this subsection hybrids of MP and heuristic techniques will be summarized, in which the contribution of MP permits to define *new heuristic methods*. When we speak about new heuristic methods we mean that heuristics cannot be defined without the contribution of MP; this contribution is given by the definition of the internal functioning of heuristics, that is derived from the functioning of MP techniques.

Angelelli et al. (16) proposed a heuristic framework relying on MIP, called Kernel Search (KS). KS works on the MIP formulation of the problem, in particular on the solution of a so called kernel problem, that corresponds to the original problem restricted to a subset of variables. The kernel problem is iteratively solved and its size is continuously increased by the addition of new variables. The initial kernel is established using information provided by the solution of the continuous relaxation of the original problem. After the initialization phase, the extension phase of the kernel takes place. During this stage, the current kernel is enlarged by the addition of variables identified by solving a sequence of small MIPs, i.e. MIPs restricted to the previous kernel plus a set of variables determined by the previous treated MIP problem. By working in this way, KS can be defined as a heuristic framework because some general steps of the algorithm must be parameterized. Applications of KS have been presented by Angelelli et al. (16) and Angelelli et al. (15) for, respectively, a portfolio selection problem and a multi-dimensional knapsack problem.

Other examples of heuristics rooted in MP can be found in the literature. Methods such as Lagrangean and Dantzig-Wolfe decompositions have been considered and reinterpreted so as to define new heuristic frameworks, (see Beasley (36)). Boschetti and Maniezzo (50) and Boschetti et al. (52) revisited Benders decomposition, (Benders (40)), Lagrangean relaxation and Dantzig-Wolfe decomposition, (Dantzig and Wolfe (84)), focusing on the possibility to define general frameworks for metaheuristics from the structure of these decomposition techniques. The interest of the authors in this issue is moved from the consideration that metaheuristics are inspired from natural phenomena and rarely from MP. This justifies the interest of the authors in investigating this methodology and in showing how, even for a basic implementation of an

MP-based metaheuristic, it is possible to obtain state of the art performances. To validate these approaches, different classes of problems have been tackled by the authors, in particular the Single Source Capacitated Facility Location, the Membership Overlay and the Multi-Mode Project Scheduling. An application of a Lagrangean heuristic for solving a traffic counter location problem has been presented by Boschetti et al. (55).

2.3 Metaheuristics subordinate to MP

In this section the usage of metaheuristics to help and improve exact algorithms is considered. This area of matheuristic contributions has been less investigated than the previous one; it represents a relatively little explored line of research in the field. The usage of heuristics integrated in the context of MP techniques does not limit its power in calculating tight bounds in order to strongly prune the search tree; in fact, it is possible to consider integration policies in which metaheuristics help MP, for example, in performing separation procedures or pricing operations. Table 2.2 reports some of these approaches. An interesting review of the topic has been presented by Puchinger et al. (182).

2.3.1 Metaheuristics for separation problems

One of the investigated issues regarding the internal functioning of B&C methods corresponds to the treatment of the separation problem, i.e. the problem of finding valid inequalities to be added to the current model that are able to cut off the current infeasible linear solution. Augerat et al. (25) presented a first proposal for dealing with the separation problem using metaheuristics. The authors presented a B&C algorithm for solving the CVRP, where different heuristics are proposed to treat the problem of separating capacity constraints, ranging from simple construction to TS. A similar approach has been proposed by Gruber and Raidl (126) to solve a MSTP. Here the separation procedure involves so called jump inequalities; because of the difficulty of the separation problem, to speed up the computation, two construction heuristics are applied to find initial partitions; they are improved by a local search and, in case no violated jump inequalities have been found, by TS. Tricoire et al. (208) proposed a hybrid approach using VNS to provide information for deriving subsets of constraints for the mathematical model of a Multi-Pile VRP; computational results executed on

small-sized instances derived from the dataset proposed by Doerner et al. (94) showed how the hybrid B&C algorithm is able to find optimal solutions for instances with up to 44 customers in less than 2 hours.

2.3.2 Metaheuristics for pricing problems and Benders decomposition

Another application of metaheuristics within the context of MP procedures can be found in the field of B&P methods, (Barnhart et al. (33)); here an investigated issue regarding the internal functioning of these algorithms involves the solution of the pricing problem, i.e. the problem of identifying new columns to be added to the mathematical model. A hybrid B&P method has been implemented by Puchinger and Raidl (181) for solving a Bin Packing Problem; here, the pricing problem is tackled by a four level hierarchy of pricing methods, composed by a greedy heuristic, an EA, the usage of CPLEX to solve, first, a restricted model of the pricing problem and then the usage of CPLEX for solving a complete IP model. Caserta and Voß (62) used a CM-inspired scheme for the column generation phase of a Dantzig-Wolfe algorithm used to solve a capacitated lot sizing problem. The authors reported preliminary computational tests by comparing results obtained by Belvaux and Wolsey (39) and Degraeve and Jans (87) on 6 instances taken from the test set used by Trigeiro et al. (209); results showed the good performances of the proposed method w.r.t. the other compared approaches, especially in terms of computational time. We refer also to the works by Ribeiro Filho and Lorena (188) and dos Santos and Mateus (97) for other applications of heuristics for the column generation phase.

Benders decomposition also benefits from metaheuristics. Rei et al. (186) focused on the usage of LB for accelerating Benders decomposition. The advantages of this integration come from the capability of LB to find upper bounds for the problem at hand and to derive different additional cuts before solving the Benders master problem. Computational results for the Multicommodity Capacitated Fixed-Charge Network Design Problem demonstrated benefits of this approach. Another integration of a metaheuristic within Benders decomposition can be found in the work proposed by Poojari and Beasley (177), where Benders decomposition is hybridized with a GA for solving MIP problems.

Table 2.2: Summary of hybrids subordinating metaheuristics to MP

Paper	Application	Solution approach
Augerat et al. (25)	CVRP	B&C + TS
Gruber and Raidl (126)	bounded diameter MSTP	B&C + TS
Puchinger and Raidl (181)	bin packing problem	Column Generation + EA
Ribeiro Filho and Lorena (188)	graph coloring problem	Column Generation + GA
dos Santos and Mateus (97)	crew-scheduling problem	Column Generation + GRASP

2.4 Cooperation between metaheuristics and MP

The definition of the word *cooperation* within matheuristics could not be well identified, because all hybrids of MP and heuristic approaches include the coexistence of these two paradigms. In the previous sections we examined some hybrid approaches that have a common feature: one of the two methodologies, (MP or heuristics), is subordinate to the other one. This means that the subordinated approach deals with performing some "tasks" that could be done by the non-subordinated one, (just think about the neighborhood exploration performed by MP procedures). Here we consider as cooperative method a framework in which MP and heuristic procedures work "at the same level", i.e. it does not exist a methodology that invokes the other one to solve specific emerging subproblems. In a cooperative approach it does not exist a method that works "as a function of" the other one. On the contrary, MP and heuristics work independently of one another, just computing on their own and interchanging information about the solution ongoing process and exploiting these information to internally produce new elements hopefully useful for the solution process.

Many contributions propose algorithms in which MP and heuristics are combined in a cooperative manner. This section presents some contributions in this area; table 2.3 reports some of the approaches that will be presented in this section.

2.4.1 Iterative cooperative approaches

In this section hybrids of MP and heuristics that iteratively exchange information will be reviewed. Here, the whole hybrid cooperative algorithm is an iterative method, in which at each iteration the heuristic and MP techniques are executed, one after the other, exchanging information about the solution ongoing process.

Archetti et al. (22) and Chouman and Crainic (68) proposed hybrids between TS and MIP in which the two methodologies iteratively exchange information. Archetti et al. (22) dealt with an inventory-routing problem; their approach iteratively applies TS to explore the neighborhood of the current solution; if a better solution is found, a MIP improvement procedure is executed, in which, first, a new assignment of routes to time periods and, second, a new assignment of customers to routes are searched for. The authors tested their approach on instances by Archetti et al. (20) and computational results compared the hybrid TS with optimal and heuristic solutions provided, respectively, by Archetti et al. (20) and Bertazzi et al. (42). The performance of the hybrid TS is very close to optimal results, i.e. the average gap is 0.06%, and the proposed algorithm gains better results w.r.t. the heuristic proposed by Bertazzi et al. (42). Other computational analysis were executed to assess the contribution of using both MIP components instead of the use of only one of the two and none. Chouman and Crainic (68) proposed an itarative method for solving a network design problem. Here the cooperation between TS and MIP is based on the creation of a restricted model that a MIP solver has to solve; the restricted model is created through variable fixing procedures, that are guided by statistical information collected by TS during its execution. Benchmark instances by Ghamlouche et al. (117) and Hewitt et al. (131) were used as test bed; the proposed hybrid method reveals its effectiveness w.r.t. path relinking procedure by Ghamlouche et al. (117) and the matheuristic proposed by Hewitt et al. (131). Pedroso and Kubo (165) tackled a lot sizing problem proposing a combination of a variant of the relax-and-fix heuristic, (by Pochet and Van Vyve (176)), and TS. The relax-and-fix heuristic consists in solving partial relaxations of the original problem through a MIP solver; this is realized by fixing some subsets of variables and relaxing the remaining ones. An ad-hoc variant of the classical relax-and-fix heuristic is implemented and used to build a starting solution; then, TS explores neighborhoods of this solution. After performing TS, the current solution is partially destructed and its reconstruction is made by a procedure relying on the relax-and-fix heuristic. Another iterative approach has been proposed by da Silva and Ochi (81); the authors implemented an algorithm hybridizing an EA with CPLEX solver for treating a scheduling problem, where the interaction between the two components consists in exchanging information every time one of the two methods finds a new best solution. CPLEX uses this information to improve its primal bound and hence remove nodes from the

tree. If CPLEX finds a better primal bound, it gives the evolutionary algorithm this information, converting the CPLEX solution into a priority list that is inserted into the current population of EA.

Schmid et al. (196) proposed to hybridize a VNS component with MIP for solving a routing problem emerging in the concrete industry. The problem requires to identify an efficient plan for the delivery of concrete from production plants to customer construction sites. The authors proposed an integer multi-commodity network flow model for the problem, where the basis of the model is the concept of fulfillment pattern, that identifies a set of operations that can completely fulfill the associated order. The cooperation between VNS and MIP components works as follows: VNS generates new patterns that are given as new components to the MIP model, that, subsequently, is optimized. After the optimization, the resulting solution is the input for a new run of VNS, that will try to improve this solution. Computational results were executed using real-life data from a concrete company located in northern Italy. Interesting comparisons between the proposed hybrid approach and a commercial solution developed for the same problem have been reported; comparisons assess the better quality of solutions that the hybrid method can achieve w.r.t. the commercial solution relying on a simulated annealing method. A similar approach in which a multiple VNS and an ILP component cooperate has been presented by Pirkwieser and Raidl (171) and applied for solving a Periodic VRPTW. Coelho et al. (74) presented a hybrid approach in which an ALNS, (Ropke and Pisinger (191)), algorithm and the exact solution of subproblems cooperate to solve an inventory routing problem; every time ALNS computes a new solution, an optimization problem called Delivery Quantities is solved with the aim of optimizing the delivery quantities associated with a given set of vehicle routes. The hybrid approach has been tested on instances derived from small single vehicle inventory routing instances presented by Archetti et al. (20) and numerical results compared the hybrid approach against optimal solutions obtained by a B&C algorithm presented by Archetti et al. (20). Results showed that the percentage gaps of the hybrid method w.r.t. optimal solutions are very small, obtaining an average optimality gap of 0.37% over a set of 160 instances.

Ljubić et al. (148) presented a hybrid method combining a CP algorithm with a multi-start heuristic approach to solve a network design problem. Starting from fractional solutions obtained by CP, feasible solutions for the original problem are

generated by applying a constructive heuristic and, subsequently, a local improvement method. Computational results have been reported for large instances generated by the authors starting from real-world inputs. To assess the contribution of the MIP component, performances of the hybrid method have been compared against results obtained by the corresponding heuristic method, i.e. the method derived from the hybrid one but without the MIP contribution. Results obtained by the hybrid algorithm were better w.r.t. its heuristic variant, permitting to calculate the best solution in 14 out of 21 instances.

Other examples of iterative cooperative methods from the literature have been proposed by Fernandes and Lourenço (109) and Archetti et al. (23).

2.4.2 Non-iterative cooperative approaches

Unlike methods shown in the previous subsection, cooperative frameworks that will be reviewed in the present subsection are composed by two well-separated phases; in each one of these phases one methodology is executed, and, results produced at the end of the first phase are given as input for the second phase.

Archetti et al. (21) and Vasquez et al. (213) proposed two-phase cooperative approaches between TS and an IP; during each phase only one methodology is executed and the cooperation is based on the usage of information provided by the first phase execution to the second phase. Archetti et al. (21) presented a two phase method to tackle a Split Delivery VRP; the idea at the basis of this cooperation is to use information provided by the solution space identified by TS for generating high quality solutions using an IP component. An analysis of solutions generated by TS is made, and the relevant features of these solutions, (e.g. the number of times a certain edge appears in all TS solutions), are used to heuristically generate a set of promising routes. Once generated this set, routes are used to build a restricted formulation of the problem, that will be solved through an IP solver. Computational results given by comparing the hybrid TS against TS presented by Archetti et al. (19) showed the effectiveness of the proposed method. Another example of two-phase hybrid method has been presented by Vasquez et al. (213) for solving a knapsack problem; here, the basic idea is to search around fractional optimum of some relaxations of the original problem, as it is supposed to find good high quality solutions in this subspace. The first phase of the algorithm consists in

solving the linear relaxation of the problem enforced by appropriate hyperplanes; during the second phase, TS is executed to explore the neighborhood around the solution calculated at the end of the first phase. Taillard (207) proposed to use TS as heuristic column generation procedure for solving a Heterogeneous Fleet VRP. TS generates a large set of routes which is used within a set partitioning formulation of the problem; the formulation is successively solved by CPLEX. Linear relaxation information have been used to improve a GA in the approach proposed by Raidl (183); here, the author implements a GA for solving a Multiconstrained 0-1 Knapsack Problem; the solution of the linear relaxation of the model of the problem is used in different phases of GA, in particular for making a solution feasible and for locally improving a feasible solution. Computational tests have been executed on large sized test data proposed by Chu (73) and Chu and Beasley (72), available from OR-library (Beasley (37, 38)); results showed that the proposed method is able to produce lower gaps w.r.t. previous approaches presented by Hinterding (132), Chu (73) and Chu and Beasley (72). Another example of cooperative method has been presented by Haouari and Chaouachi Siala (130) to tackle a Steiner tree problem. Here, the authors implemented a cooperation between a Lagrangean decomposition technique and a GA; the volume algorithm by Barahona and Anbil (32) is used to solve the Lagrangean dual problem associated to the corresponding relaxation of the model; then, GA is executed exploiting information provided by the previous execution of the volume algorithm, in particular produced reduced costs are used to generate feasible solutions that will constitute a part of the initial population for GA. Following a similar idea, a hybrid approach between Lagrangean decomposition and EA has been proposed by Pirkwieser et al. (173) and Pirkwieser et al. (174) for solving the Knapsack Constrained Maximum Spanning Tree Problem.

Bent and Van Hentenryck (41) presented a two-phase hybrid algorithm applied to solve the VRPTW; the first phase deals with the minimization of the number of vehicles to be used and the second phase aims at decreasing the total traveled distance. In the first phase a SA, (Kirkpatrick et al. (139)), is used to minimize the number of vehicles, while the second phase is carried out by a LNS, (Shaw (198)), using a B&B method to explore the neighborhood.

A cooperation between Recovering Beam Search, (Della Croce et al. (89)), and MIP has been proposed by Della Croce et al. (90) for solving a Flow Shop Problem. The problem asks to find a sequence of jobs, to be executed on two machines, that

minimizes the sum of all completion times of jobs. The proposed method is based on a two-stage scheme. In a first phase, a heuristic solution is generated using RBS. During the second phase, an iterative process of neighborhood search is executed, in which the neighborhood of a solution is defined by choosing a particular subsequence of jobs and optimizing the positioning of identified jobs in the identified subsequence; jobs not belonging to the subsequence maintain the same position as in the original sequence. The defined neighborhood is explored by means of a MILP solver. Similar neighborhood structures have been used by Della Croce and Salassa (88) for solving a Nurse Rostering Problem.

Other examples of cooperative methods from the literature have been proposed by Archetti et al. (18), Pitakaso et al. (175), Ioannou et al. (135), Anghinolfi et al. (17) and Leitner and Raidl (144).

Table 2.3: Summary of cooperating hybrids

Paper	Application	Solution approach
Archetti et al. (21)	split delivery VRP	TS + IP
Archetti et al. (22)	inventory routing problem	TS + MIP
Chouman and Crainic (68)	multicommodity capacitated fixed-charge network design problem	TS + MIP
Vasquez et al. (213)	0-1 multidimensional knapsack problem	TS + Simplex Algorithm
Haouari and Chaouachi Siala (130)	prize collecting Steiner tree problem	GA + Lagrangean Decomposition
Pirkwieser et al. (173)	knapsack constrained maximum spanning tree problem	EA + Lagrangean Decomposition
Ioannou et al. (135)	multi-TSP with Time Windows	GA + DP
Pirkwieser and Raidl (171)	periodic VRPTW	VNS + ILP
Coelho et al. (74)	inventory routing problem	ALNS + MILP
Ljubić et al. (148)	network design problem	multi-start heuristic + CP

3

A Lagrangean Column Generation Heuristic for the CVRP

3.1 Introduction

The VRP is the problem of supplying a set of customers using a fleet of vehicles. It was introduced by Dantzig and Ramser (83) in 1959 and it represents one of the biggest success stories in operations research. Data of a VRP are the set of customers to be served, along with the cost of traveling distances between any pair of them, and the fleet of vehicles to be used for serving customers. A central depot is used as a basis for vehicles. The solution of a VRP asks for the construction of a set of routes starting and ending at the depot, each performed by a single vehicle, such that all customers are serviced, all operational constraints are satisfied and the total cost of the set of routes is minimized.

Many problems belong to the VRP family, according to existing operational constraints; one of the most studied problems is the CVRP. Each customer of a CVRP has a known request of goods and the fleet is composed by identical vehicles with a known capacity to carry goods to customers. The CVRP calls for the design of a set of routes such that each customer is visited exactly once, the total demand of goods of each route does not exceed the capacity of vehicles and the total cost of the set of routes is minimized. The CVRP is *NP*-hard. Given the *NP*-hardness of the problem,

pure exact algorithms can be effectively applied only for solving instances of a limited size; heuristics can be always applied to solve CVRP instances, regardless of their size.

Among the exact algorithms proposed for the CVRP, we mention methods presented by Baldacci et al. (29), Fukasawa et al. (115) and Baldacci et al. (30). Baldacci et al. (29) proposed a B&C algorithm based on a two-commodity network flow formulation. Fukasawa et al. (115) presented an algorithm combining a B&C with a B&C&P based on a two-index and SP formulations. The algorithm proposed by Baldacci et al. (30) was based on a SP formulation strengthened by capacity and clique inequalities.

In this chapter, we describe a matheuristic algorithm to solve the CVRP. The method relies on a CG algorithm based on a SP formulation with additional constraints and on a subgradient optimization method based on a SC model with additional constraints. The pricing step of the CG implements an additive bounding procedure, (see Fischetti and Toth (111) for details about the topic of additive bounding procedures), able to produce new negative reduced costs columns to be added to the current core of columns of the SP model. At the end of the CG procedure, the corresponding core of columns is used to build a SC model with additional constraints, which has to identify a feasible CVRP solution; the SC model is relaxed in a Lagrangean fashion and treated via subgradient optimization; a pruning heuristic is then used to fix infeasibilities of the subgradient solution and to obtain a feasible CVRP solution. In the following we describe the SP and SC formulations, the additive bounding procedure used for pricing and the Lagrangean optimization for the SC model. This chapter is organized as follows: in section 3.2 we describe the SP and SC formulations for the CVRP and some relaxation techniques, in section 3.3 we detail the implemented matheuristic algorithm, in section 3.4 we discuss computational results.

3.2 Mathematical formulations and relaxations

Let $G = (V', A)$ be a complete graph, where $V' = \{0, 1, ..., n\}$ is the set of $n + 1$ vertices and A is the set of arcs. Vertices represent customers to be supplied. Vertex 0 corresponds to the depot and we have the vertex subset $V = V' \setminus \{0\}$ composed by n vertices. Each vertex $i \in V'$ has an associated demand q_i, (we assume $q_0 = 0$). Each arc $(i, j) \in A$ has an associated travel cost d_{ij}. We indicate with D the matrix of travel

costs d_{ij}. We indicate with $\Gamma_i \subseteq V'$ the set of *successors* of i in G and with $\Gamma_i^{-1} \subseteq V'$ the set of *predecessors* of i in G, $\forall i \in V'$.

A fleet of m identical vehicles of capacity Q available at the depot has to serve vertices. We indicate with $R = (0,\ i_1,\ ...,\ i_r,\ 0)$, with $r \geq 1$, a vehicle route; each vehicle route R is a simple circuit in G passing through the depot, visiting vertices $V(R) = \{0,\ i_1,\ ...,\ i_r\}$, $V(R) \subseteq V'$, and such that the total demand of the visited vertices does not exceed the vehicle capacity Q; each vehicle route R has a cost equal to the sum of the travel costs of the arc set, $A(R)$, traversed by route R.

The CVRP asks for the design of a set of m routes of minimum total cost such that each vertex is visited exactly once by exactly one route.

3.2.1 SP formulation

In the following we show the SP formulation with additional constraints for the CVRP. Let $\mathscr{R}$ be the index set of all feasible routes, and let $\mathscr{R}_i \subset \mathscr{R}$ be the index set of routes covering vertex $i \in V'$. The cost associated to each route $\ell \in \mathscr{R}$ is $c_\ell = \sum_{(i,j)\in A(\ell)} d_{ij}$. Let x_ℓ, $\ell \in \mathscr{R}$, be a (0-1) binary variable equal to 1 if and only if route ℓ is in the optimal solution. The CVRP formulation based on the SP model with additional constraints is

$$(SP) \quad z(SP) = \min \sum_{\ell \in \mathscr{R}} c_\ell x_\ell \tag{3.1a}$$

$$\text{s.t.} \quad \sum_{\ell \in \mathscr{R}_i} x_\ell = 1, \quad \forall i \in V \tag{3.1b}$$

$$\sum_{\ell \in \mathscr{R}_0} x_\ell = m, \tag{3.1c}$$

$$x_\ell \in \{0,1\}, \quad \forall \ell \in \mathscr{R} \tag{3.1d}$$

Constraints (3.1b) impose that each vertex $i \in V$ has to be visited by exactly one route. Constraint (3.1c) specifies that exactly m routes have to be selected.

3.2.2 SC formulation

In the following we show the mathematical formulation of the SC model with additional constraints.

$$(SC) \qquad z(SC) = \min \sum_{\ell \in \mathscr{R}} c_\ell x_\ell \tag{3.2a}$$

$$\text{s.t.} \quad \sum_{\ell \in \mathscr{R}_i} x_\ell \geq 1, \quad \forall i \in V \tag{3.2b}$$

$$\sum_{\ell \in \mathscr{R}_0} x_\ell \leq m, \tag{3.2c}$$

$$x_\ell \in \{0, 1\}, \quad \forall \ell \in \mathscr{R} \tag{3.2d}$$

Constraints (3.2b) impose that each vertex $i \in V$ has to be visited by at least one route. Constraint (3.2c) specifies that at most m routes have to be selected.

Let $u = (u_1, ..., u_n)$ be the non-negative vector of dual variables, where u_i $i = 1, ..., n$ are associated to constraints (3.2b). Let v be the non-positive dual variable associated to constraint (3.2c). Hence, the dual problem of the LP relaxation of SC can be defined as follows

$$(DSC) \qquad z(DSC) = \max \sum_{i \in V} u_i + mv \tag{3.3a}$$

$$\text{s.t.} \quad \sum_{i \in \mathrm{V}(\ell) \setminus 0} u_i + v \leq c_\ell, \quad \forall \ell \in \mathscr{R} \tag{3.3b}$$

$$u_i \geq 0, \quad \forall i \in V \tag{3.3c}$$

$$v \leq 0, \tag{3.3d}$$

3.2.3 (q, i)-path and ng-path relaxations

A *forward* path $P = (0, i_1, ..., i_{k-1}, i_k)$ is an elementary path starting from the depot 0, visiting vertices $V(P) = \{0, i_1, ..., i_{k-1}, i_k\}$ and ending at vertex $i_k = \sigma(P)$. Let us denote by $A(P)$ the set of arcs traversed by P and by $c(P) = \sum_{(i,j) \in A(P)} d_{ij}$ the cost of path P.

(q, i)-path and ng-path are two well-known techniques to obtain relaxations of forward paths.

A (q, i)-path is a non-necessarily elementary path starting from the depot 0, visiting a set of vertices of total demand equal to q and ending at the vertex i. The cost $f(q, i)$ of the least cost (q, i)-path can be computed using DP as described by Christofides et al. (71). A (q, i)-route is a $(q, 0)$-path with i as last visited vertex before arriving at the depot 0. (q, i)-path relaxation can easily avoid 2-cycles, i.e. cycles like $(0, i_1, ..., i_{j-1}, i_j, i_{j+1}, ..., i_{k-1}, i_k)$ where $i_{j-1} = i_{j+1}$. It is possible to demonstrate that $f(q, i)$ is a valid lower bound on the cost $c(P)$ of any forward path P, such that $q(P) = q$ and $\sigma(P) = i$.

ng-path relaxation is a technique introduced by Baldacci et al. (31) to obtain a valid lower bound on the cost $c(P)$ of any forward path P; the technique can be described as follows. Let us define $N_i \subseteq V$ as a set of selected vertices for vertex i (according to some criterion) such that $N_i \ni i$ and $|N_i| \leq \Delta(N_i)$, where $\Delta(N_i)$ is a parameter (if $\Delta(N_i) = 4$, $\forall i \in V$, N_i contains i and the three nearest vertices to i). Using sets N_i it is possible to associate to each forward path $P = (0, i_1, ..., i_{k-1}, i_k)$ the subset $\Pi(P) \subseteq V(P)$ containing vertex i_k and every vertex i_r, $r = 1, ..., k$ - 1 of P that belongs to all sets $N_{i_{r+1}}, ..., N_{i_k}$ associated to vertices $i_{r+1}, ..., i_k$ visited after i_r. We can define set $\Pi(P)$ as

$$\Pi(P) = \{i_r : i_r \in \bigcap_{s=r+1}^{k} N_{i_s}, r = 1, ..., k-1\} \bigcup \{i_k\}. \tag{3.4a}$$

A forward ng-path $(NG,\ q,\ i)$ is a non-necessarily elementary path $P = (0, i_1, ..., i_{k-1}, i_k = i)$ starting from the depot 0, visiting a subset of vertices of total demand equal to q such that $NG = \Pi(P)$, ending at vertex i, and such that $i \notin \Pi(P')$, where $P' = (0, i_1, ..., i_{k-1})$. The cost of the least cost forward ng-path $(NG,\ q,\ i)$ is denoted by $f(NG,\ q,\ i)$. We define an $(NG,\ q,\ i)$-route as an $(NG,\ q,\ 0)$-path where i is the last vertex visited before arriving at the depot 0. The cost of the $(NG,\ q,\ i)$-route of minimum cost is given by $f(NG,\ q,\ i) + d_{i0}$. Functions $f(NG, q, i)$ can be computed using DP recursions on the state space graph $\mathscr{H} = (\mathscr{E}, \Psi)$ defined as

$$\mathscr{E} = \{(NG,\ q,\ i) : q_i \leq q \leq Q, \forall NG \subseteq N_i \text{ s.t. } NG \ni i \text{ and } \sum_{j \in NG} q_j \leq q, \forall i \in V'\}, \tag{3.5a}$$

$$\Psi = \{((NG',\ q',\ j), (NG,\ q,\ i)) : \forall (NG',\ q',\ j) \in \Psi^{-1}(NG,\ q,\ i), \forall (NG,\ q,\ i) \in \mathscr{E}\}, \tag{3.6a}$$

where $\Psi^{-1}(NG,\ q,\ i) = \{(NG',\ q\ -\ q_i,\ j) : \forall NG' \subseteq N_j \text{ s.t. } NG' \ni j \text{ and } NG' \cap N_i = NG \setminus \{i\},\ \forall j \in \Gamma_i^{-1}\}$.

The DP recursion for calculating $f(NG,\ q,\ i)$ is the following

$$f(NG,\ q,\ i) = \min_{(NG',q-q_i,j)\in\Psi^{-1}(NG,q,i)} \{f(NG',\ q\ -\ q_i,\ j) + d_{ji}\}, \forall (NG,\ q,\ i) \in \mathscr{E}. \tag{3.7a}$$

It is possible to demonstrate that $f(NG,\ q,\ i)$ is a valid lower bound to the cost $c(P)$ of any forward path P, such that $q(P) = q$ and $\sigma(P) = i$. The quality of the lower bound calculated by ng-path relaxation strongly relies on the definition of sets $N_i\ \forall i \in V$, since the set $\Pi(P)$ represents vertices that cannot be visited along the path $P = (0,\ i_1,\ ...,\ i_{k\ -\ 1},\ i_k)$ immediately after the vertex i_k and $\Pi(P)$ is defined as the intersection of sets N_i associated to vertices visited before i_k plus vertex i_k itself. A proper definition of sets N_i permits to obtain better quality paths, aiming at avoiding loops and, in this way, producing paths "nearer" to elementariness. Figure 3.1 shows an example of expansion of an ng-path for a graph composed by 9 vertices, (8 plus the depot 0). The figure also shows the composition of N_i sets. At the beginning of the expansion, the subset $\Pi(P)$ is empty, then the extension of the path to vertex 1 and the update of the subset $\Pi(P)$ are done, obtaining $\Pi(P) = \{1\}$; then, the extension to vertex 2 is allowed since it does not belong to the subset $\Pi(P)$; subsequently, the corresponding update of the subset $\Pi(P)$ is done, obtaining $\Pi(P) = \{1,\ 2\}$; after this, the extension to vertex 3 is made, obtaining the subset $\Pi(P) = \{1,\ 2,\ 3\}$; hence, the extension to vertex 7 can be done, obtaining $\Pi(P) = \{7\}$.

3.3 The algorithm

In this section we describe the implemented algorithm to solve the CVRP. The algorithm is a matheuristic able to produce feasible CVRP solutions using MP methods as a basis for ameliorating the search of good quality feasible solutions. The objective of the algorithm is that of producing feasible CVRP solutions minimizing total travel

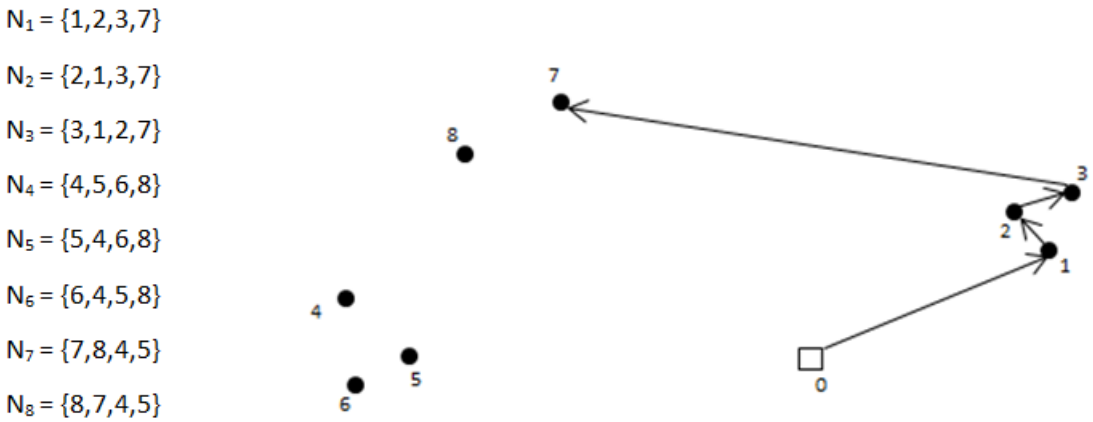

Figure 3.1: Example of expansions of an *ng*-path

costs, while fixing the number of used vehicles to a predefined value, as shown by the SP with additional constraints model 3.1.

The matheuristic can be divided in two main phases. During the first phase, the algorithm applies a CG method relying on an additive bounding procedure to generate a reduced problem RSP obtained from SP model (3.1) by replacing the route set $\mathscr{R}$ with the set $\mathscr{R}'$ composed by (q, i)-routes or ng-routes; at each CG iteration the linear relaxation of the RSP problem is solved by an LP solver. When the CG ends, the second phase of the algorithm starts. The pool of columns of the RSP model is used to build a reduced SC model, RSC, obtained from the SC (3.2) model, to which it is demanded to identify a minimum cost feasible CVRP solution. The RSC model is relaxed in a Lagrangean fashion and solved via subgradient optimization; at each iteration of the algorithm, a pruning heuristic uses the subgradient solution as a basis for building a feasible CVRP solution, by fixing the infeasibilities of the subgradient solution. Hence, the first phase of the matheuristic is made by the solution of the RSP model that possibly produces a valid lower bound on the cost of the optimal CVRP solution, while the second phase is made by the subgradient optimization of the RSC model able to produce a feasible CVRP solution. This is summarized in algorithm 1, where at line 2 the solution of the RSP model is computed, producing as a result the pool of columns of the reduced model; at line 3 the second phase of the method takes place, through the construction of the Lagrangean relaxed RSC model, (using the RSP pool of columns), and the execution of the subgradient optimization; at line 4 the

algorithm returns the best solution found during subgradient optimization.

Algorithm 1 Lagrangean CG Heuristic

1: **procedure** LAGR_CG_HEU(D, n, m, q_i, Q, α)
2: $Pool_C \leftarrow$ Column_Gen_RSP(D, n, m, q_i, Q, LB)
3: $s_{best} \leftarrow$ Lagr_Heu_RSC($Pool_C$, D, m, it_tot LB, α)
4: **return** s_{best}

3.3.1 CG and additive bounding procedure

Algorithm 2 details the main operations done in the first phase of the proposed matheuristic algorithm, i.e. the execution of the CG relying on an additive bounding procedure to obtain a valid CVRP lower bound. Algorithm 2 takes as input, in order, the matrix of travel costs D, the number of vertices n, the number of vehicles to be used m, the vector of demands of vertices q_i, $\forall i \in V'$, and the capacity of vehicles Q; the algorithm gives as output the pool of columns $Pool_F$ composed by $(q,\ i)$-routes or ng-routes identified during the CG, together with the value of the calculated lower bound LB.

Before starting the CG, the pools of, respectively, $(q,\ i)$-routes and ng-routes columns are initialized as empty sets at lines 2-3, while at line 4 the pool of columns $Pool_C$ of the RSP model is initialized as an empty set. Each entry of the matrix of reduced costs $DRed(i,\ j)$ is initialized with travel cost $D(i,\ j)$ $\forall (i, j) \in$ A at line 5. A feasible CVRP solution s calculated by a simple constructive heuristic possibly initializes the core of columns of the master RSP problem at lines 8-9. The heuristic is a simple sequential insertion, that constructs a route at a time, until the capacity of vehicles Q is not violated; the next vertex to be inserted in the currently under construction route is the unrouted vertex that minimizes the extra-mileage. When all vertices have been inserted in exactly one route, an ejection chain procedure, (see Glover (120)), is executed on the resulting solution, if the number of its routes is bigger than the predefined value m. If the ejection chain procedure succeeds in normalizing the number of routes to m, the created feasible CVRP solution is used to initialize the core of columns of the master RSP problem, otherwise single vertex routes $(0,\ i,\ 0)$, $\forall i \in V$ are used as core columns.

Lines 10-31 execute the additive bounding procedure, composed by $(q,\ i)$-route and ng-route pricing. The core of columns of the master RSP problem is first enlarged

Algorithm 2 CG RSP

```
1: procedure Column_Gen_RSP(D, n, m, q_i, Q, LB)
2:     qi_r_pool ← ∅
3:     ng_r_pool ← ∅
4:     Pool_C ← ∅
5:     DRed(i, j) ← D(i, j)   ∀i, j ∈ V'
6:     qi_b_feas ← false
7:     ng_b_feas ← false
8:     s ← Create_UB(D)
9:     Pool_C ← Init_Master(s)
10:    repeat
11:        g ← Solve_Master(Pool_C, z(RSP)_qi)
12:        DRed ← Calc_Red_Costs(D, g)
13:        new_qi_r_pool ← Qi_Route_Pricing(DRed, n, q_i, Q)
14:        qi_r_pool ← qi_r_pool ∪ new_qi_r_pool
15:        Pool_C ← Pool_C ∪ new_qi_r_pool
16:        if new_qi_r_pool = ∅ then
17:            LB = z(RSP)_qi
18:            qi_b_feas ← true
19:    until (new_qi_r_pool ≠ ∅   ||   time_limit_not_exceeded)
20:    if qi_b_feas = true then
21:        Pool_C ← Init_Master(s)
22:        repeat
23:            g ← Solve_Master(Pool_C, z(RSP)_ng)
24:            DRed ← Calc_Red_Costs(D, g)
25:            new_ng_r_pool ← NG_Route_Pricing(DRed, n, q_i, Q)
26:            ng_r_pool ← ng_r_pool ∪ new_ng_r_pool
27:            Pool_C ← Pool_C ∪ new_ng_r_pool
28:            if new_ng_r_pool = ∅ then
29:                LB = z(RSP)_qi + z(RSP)_ng
30:                ng_b_feas ← true
31:        until (new_ng_r_pool ≠ ∅   ||   time_limit_not_exceeded)
32:    if ng_b_feas = true then
33:        Pool_F ← ng_r_pool
34:    else if q_b_feas = true then
35:        Pool_F ← ng_r_pool
36:    else
37:        Pool_F ← qi_r_pool
38:    return Pool_F
```

with (q, i)-routes, (lines 10-19). The master RSP problem is solved with the simplex algorithm, at line 11; reduced costs $\bar{d}_{ij} = d_{ij}$ - $(1/2)(g_i + g_j), \forall(i,\ j) \in$ A are calculated with respect to the current dual solution g, at line 12; hence, we compute functions $f(q,\ i)$ using reduced costs $\bar{d}_{ij}$ instead of d_{ij}, at line 13. Only negative reduced costs (q, i)-routes are used to enlarge the current core of columns of the master RSP problem; these routes are added to the pool *qi_r_pool* of (q, i)-routes, at line 14, and to the pool of columns $Pool_C$ of the master RSP problem, at line 15. The (q, i)-route bounding procedure stops when no negative reduced costs (q, i)-routes can be found or when a user-defined time limit since the beginning of the bounding procedure is exceeded. In case the time limit is exceeded, the solution of the master RSP problem does not produce a valid lower bound on the cost of the optimal CVRP solution, hence the additive bounding procedure ends; otherwise, the solution of the master RSP problem produces a valid lower bound $LB = z(RSP)_{qi}$, and the procedure continues with the execution of the additive component based on the computation of negative reduced costs ng-routes, (lines 22-31). ng-route and sets N_i are computed using reduced costs $\bar{d}_{ij}$ derived from the valid (q, i)-route lower bound, i.e. $\bar{d}_{ij} = d_{ij}$ - $(1/2)(g_i + g_j), \forall(i,j) \in$ A. The core of columns of the master RSP problem is reinitialized in the same manner as before the execution of the (q, i)-route-based bounding procedure, at line 21. The master RSP problem is solved with the simplex algorithm, at line 23 and reduced costs $\bar{d}_{ij} = d_{ij}$ - $(1/2)(g_i + g_j), \forall(i,\ j) \in$ A are calculated with respect to the current dual solution g of the master problem, at line 24; hence, we compute functions $f(NG,\ q,\ i)$ using reduced costs $\bar{d}_{ij}$, at line 25. Negative reduced costs ng-routes are used to enlarge the current core of columns of the master RSP problem; these routes are added to the pool *ng_r_pool* of ng-routes, at line 26, and to the pool of columns $Pool_C$ of the master RSP problem, at line 27. The ng-route bounding procedure stops when no negative reduced costs ng-routes can be found or when a user-defined time limit from the beginning of the ng-route bounding procedure is exceeded; in case no negative reduced costs ng-routes can be found, a valid lower bound $LB = z(RSP)_{qi} + z(RSP)_{ng}$ on the optimal cost of the CVRP solution has been found.

Lines 32-37 decide what pool of columns $Pool_F$ will be used for the construction of the RSC model. In case the ng-route bounding procedure has been executed, the pool of ng-routes is returned; this is done even if no valid ng-route lower bound is

found, since the quality of ng-routes is better than the quality of (q, i)-routes. On the contrary, the pool of (q, i)-routes is returned for the construction of the RSC model.

3.3.2 The Lagrangean heuristic

The second phase of the matheuristic asks for the identification of a minimum cost feasible CVRP solution using information derived from the output of the CG algorithm, i.e. the pool of routes $Pool_F$ and possibly the value of a valid lower bound LB of the optimal cost of the solution of CVRP. (q, i)-routes and ng-routes are relaxations of feasible routes, since they respect capacity constraints, but can contain loops; hence we need to transform all non-elementary routes produced by the additive bounding procedure in elementary ones; moreover, if we want to have a CVRP solution, we need to choose a subset of m elementary routes covering each vertex exactly once. To first gain elementariness and to subsequently have a choice mechanism of elementary routes, we implemented a heuristic method composed by a procedure to make elementary the routes produced by the additive bounding procedure and a Lagrangean heuristic able to produce feasible CVRP solutions via subgradient optimization. The pseudocode of the heuristic procedure is presented in algorithm 3. The method takes as input the pool of columns generated by the additive bounding procedure $Pool_C$, the matrix of travel costs D, the number of vehicles to be used m, the value of the lower bound calculated by the additive bounding procedure LB, α, (a numerical value used to update the value of penalties during the calculation of the subgradient vector) and it_tot, the total number of subgradient iterations; the output of the procedure is the best feasible CVRP solution found s_{best}.

The first step of the heuristic consists in making elementary every non-elementary route in the pool of columns $Pool_C$, at line 2; this is done by generating a new route r', composed by all vertices of the corresponding non-elementary route r in the pool $Pool_C$ repeated exactly once. After its construction, the route r' is optimized to decrease the value of total traveled distance; this is realized through the execution of a 3-opt local search heuristic on r'. New generated routes are added to a new pool, called $Pool_E$.

The heart of the heuristic procedure represented by algorithm 3 is the subgradient optimization, (lines 3-11). $Pool_E$ is used to construct a RSC model obtained from SC (3.2) model by replacing the route set $\mathscr{R}$ with the pool $Pool_E$. Let $Pool_{E_i}$ be the subset of routes covering vertex $i \in V$. We apply the parametric relaxation for the SC model

Algorithm 3 Lagrangean Heuristic RSC

```
1: procedure LAGR_HEU_RSC(Pool_C, D, m, LB, α,it_tot)
2:     Pool_E ← Elem_Routes(Pool_C)
3:     λ_i ← 0   ∀i ∈ V'
4:     cont_it ← 0
5:     repeat
6:         Solve_Lagr_Dual(Pool_E, λ_i, m)
7:         x_sub ← Update_Penalties(Pool_E, λ_i, α, m, LB)
8:         s' ← Pruning_Heuristic(x_sub, D)
9:         s_best ← Update_Sol(s', D)
10:        cont_it ← cont_it + 1
11:    until (cont_it < it_tot   ||   time_limit_not_exceeded)
12:    return s_best
```

shown by Boschetti and Maniezzo (51) to our RSC model; following this relaxation, it is possible to replace each variable x_ℓ of the model by a new set of $|V(\ell) \setminus \{0\}|$ variables $y^i_\ell, i \in V(\ell) \setminus \{0\}$ as follows

$$x_\ell = \sum_{i \in V(\ell)\setminus\{0\}} \frac{w_i}{w(V(\ell))} y^i_\ell, \quad \ell \in Pool_E \tag{3.8a}$$

where w_i is a positive real weight associated with each vertex $i \in V$ and $w(V(\ell)) = \sum_{i \in V(\ell)\setminus\{0\}} w_i$ represents the total weight of column (route) $\ell \in Pool_E$. The resulting mathematical formulation of the parametric relaxation of the RSC problem is

$$(PRSC(w)) \qquad z(PRSC)(w) = \min \sum_{\ell \in Pool_E} \sum_{i \in V(\ell)\setminus\{0\}} c_\ell \frac{w_i}{w(V(\ell))} y^i_\ell \tag{3.9a}$$

$$\text{s.t.} \quad \sum_{\ell \in Pool_{E_i}} \sum_{h \in V(\ell)\setminus\{0\}} \frac{w_h}{w(V(\ell))} y^h_\ell \geq 1, \quad \forall i \in V \tag{3.9b}$$

$$\sum_{\ell \in Pool_{E_0}} \sum_{h \in V(\ell)\setminus\{0\}} \frac{w_h}{w(V(\ell))} y^h_\ell \leq m, \tag{3.9c}$$

$$y^i_\ell \in \{0,1\}, \quad \ell \in Pool_{E_i}, \quad i \in V \tag{3.9d}$$

We relax in a Lagrangean fashion both PRSC(w) constraints (3.9b) and (3.9c). Consider a *penalty* vector $\lambda = (\lambda_1\ ,...,\ \lambda_n,\ \lambda_{n+1})$ of $n+1$ non-negative real numbers, where $\lambda_i \geq 0,\ i = 1,\ ...,\ n$ is a real number associated to constraint (3.9b) for vertex $i \in V$ and $\lambda_{n+1} \geq 0$ is associated to constraint (3.9c). We obtain the following problem

$$(LPRSC(\lambda, w))z(LPRSC)(\lambda, w) = \min \sum_{\ell \in Pool_E} \sum_{i \in V(\ell)\setminus\{0\}} (c_\ell - \lambda'(V(\ell)))\frac{w_i}{w(V(\ell))} y_\ell^i + \sum_{i \in V} \lambda_i - m\lambda_{n+1} \tag{3.10a}$$

$$y_\ell^i \in \{0, 1\}, \quad \ell \in Pool_{E_i}, \quad i \in V \tag{3.10b}$$

where $\lambda'(V(\ell)) = \lambda(\mathrm{V}(\ell)) - \lambda_{n+1}$ and $\lambda(\mathrm{V}(\ell)) = \sum_{h \in V(\ell)\setminus\{0\}} \lambda_h$.

Problem LPRSC(λ, w) is decomposable into n subproblems, one for each row $i \in V$

$$(LPRSC^i(\lambda,\ w)) \qquad z^i(LPRSC)(\lambda,\ w) = \min \sum_{\ell \in Pool_{E_i}} c_\ell^i(\lambda,\ w) y_\ell^i + \lambda_i \tag{3.11a}$$

$$\text{s.t.} \quad y_\ell^i \in \{0,\ 1\}, \quad \ell \in Pool_{E_i} \tag{3.11b}$$

where $c_\ell^i(\lambda,\ w) = (c'_\ell - \lambda(V(\ell)))\frac{w_i}{w(V_\ell)}$ and $c'_\ell = c_\ell + \lambda_{n+1}$.

We set $w_i = \lambda_i$ and add the constraint $\sum_{\ell \in Pool_{E_i}} y_\ell^i = 1, \forall i \in V$. The subproblem LPRSC$^i(\lambda,\ w),\ i \in V$ can be rewritten as follows

$$(LPRSC^i(\lambda)) \qquad z^i(LPRSC)(\lambda) = \min \sum_{\ell \in Pool_{E_i}} c'_\ell \frac{\lambda_i}{\lambda(V(\ell))} y_\ell^i \tag{3.12a}$$

$$\text{s.t.} \quad \sum_{\ell \in Pool_{E_i}} y_\ell^i = 1, \tag{3.12b}$$

$$y_\ell^i \in \{0,\ 1\}, \quad \ell \in Pool_E \tag{3.12c}$$

Hence, the overall value of the Lagrangean problem LPRSC(λ) is

$$z(LPRSC)(\lambda) = \sum_{i \in V} z^i(LPRSC)(\lambda) - m\lambda_{n+1} \tag{3.13a}$$

Boschetti and Maniezzo (51) showed that any optimal solution of problem LPRSC(λ) provides a feasible solution (u, v) of cost $z(LPRSC)(\lambda)$ for the reduced dual problem RDSC, obtained from the DSC model by replacing the route set $\mathscr{R}$ with the pool $Pool_E$. A feasible dual solution (u, v) of cost $z(LPRSC)(\lambda)$ for problem RDSC can be obtained by means of the following expressions

$$u_i = \min_{\ell \in Pool_{E_i}} \{c'_\ell \phi_{i\ell}\} \quad i \in V \tag{3.14a}$$

$$v = -\lambda_{n+1} \tag{3.14b}$$

where $c'_\ell = c_\ell + \lambda_{n+1}$ and

$$\phi_{i\ell} = \begin{cases} \frac{\lambda_i}{\lambda(V(\ell))} & \lambda(V(\ell)) > 0 \\ \frac{1}{|V(\ell)\setminus\{0\}|} & \lambda(V(\ell)) = 0 \end{cases} \tag{3.15a}$$

The best lower bound that can be achieved using expressions (3.14) is equal to the optimal solution cost $z(RDSC)$ of the problem RDSC; this value can be obtained calculating the maximum of the function $z(LPRSC)(\lambda)$ with respect to $\lambda \geq 0$, i.e.

$$\max_{\lambda \geq 0} \{z(LPRSC)(\lambda)\} = z(RDSC) \tag{3.16a}$$

The problem (3.16) is called Lagrangean dual, and we need to solve it to find the optimal (or near-optimal) dual solution of cost $z(RDSC)$. To deal with the Lagrangean dual we implement a subgradient method that searches the space of possible values for λ vectors and obtains the best possible lower bound. Lines 3-11 of algorithm 3 represent the pseudocode related to the execution of the subgradient method. At line 3 each component of the λ vector is initialized to 0, $\forall i \in V'$. At line 4 the counter of subgradient iterations is initialized to 0. Loop 5-11 is the core of subgradient optimization; at line 6 the Lagrangean dual problem (3.16) is solved for the given λ vector. At line 7 the subgradient vector is calculated and used to update λ vector. Let us indicate with $J \subset Pool_E$ the index subset of routes that produce minima of formula (3.14a) $\forall i \in V$, i.e. $J = \{\ell \in Pool_E : \ell = \text{argmin}_{\ell \in Pool_{E_i}}[c'_\ell \phi_{i\ell}], i \in V\}$. Let (u, v) be the dual solution of cost $z(LPRSC)(\lambda)$ computed by expressions (3.14) at point λ. Let x be the corresponding non-necessarily feasible solution of RSC computed as

$$x_\ell = \begin{cases} \sum_{i \in I_\ell} \phi_{i\ell} & \ell \in J \\ 0 & otherwise \end{cases} \tag{3.17a}$$

where $I_\ell = \{i \in V : u_i = c'_\ell \phi_{i\ell}\}$. A valid subgradient of the function $z(LPRSC)(\lambda)$ is given by the vector $\theta = (\theta_1, ..., \theta_n, \theta_{n+1})$, calculated according to the following formulas

$$\theta_i = 1 - \sum_{\ell \in Pool_{E_i}} x_\ell, \quad i \in V \tag{3.18a}$$

$$\theta_{n+1} = m - \sum_{\ell \in Pool_{E_0}} x_\ell \tag{3.18b}$$

The vector of Lagrangean penalties λ is then updated according to the following formulas

$$\lambda_i = \max\{0, \lambda_i + \alpha \frac{0.1LB}{\sum_{j \in V'} \theta_j^2} \theta_i\}, \quad i \in V \tag{3.19a}$$

$$\lambda_{n+1} = \max\{0, \lambda_{n+1} - \alpha \frac{0.1LB}{\sum_{j \in V'} \theta_j^2} \theta_{n+1}\} \tag{3.19b}$$

where LB is the value of the lower bound calculated during the additive bounding procedure and α is a user defined constant. Solution x, calculated following formulas (3.17), is returned at line 7; x is referenced in the pseudocode as x_{sub}.

x_{sub} can be an infeasible solution. Infeasibilities are linked both to the number of occurrences of vertices in the solution and to the number of used vehicles; in fact we can have some vertices that are visited many times by one or more routes, while other vertices are never visited by routes, and we can have that a bigger number of vehicles is currently in use in x_{sub} than the predefined number of vehicles m. The constraint on capacity is, instead, always respected by x_{sub}, since both $(q,\ i)$-routes and ng-routes bounding procedures implicitly respect this constraint. Hence, we implement a pruning heuristic with the aim of fixing infeasibilities of the subgradient solution x_{sub}, giving

Algorithm 4 Pruning Heuristic

```
1: procedure PRUNING_HEURISTIC(x_sub, D)
2:     x' ← Restore_Occurrences(x_sub)
3:     x'' ← Restore_Vehicles(x')
4:     if x''isfeasible then
5:         x_F ← VND(x'', D)
6:     else
7:         x_F ← ∅
8:     return x_F
```

as output a minimum cost feasible CVRP solution. The pruning heuristic is invoked at line 8 of algorithm 3; a pseudocode of the procedure is proposed in algorithm 4.

The first step of the pruning heuristic consists in normalizing the number of occurrences of each vertex $i \in V$, (line 2); this means that only one location will be chosen for vertices occurring many times in the solution x_{sub}, while vertices not present in x_{sub} will be inserted exactly once. The pruning heuristic determines the location of each vertex i occurring many times to be the location with the minimum extra-mileage for visiting i, i.e. the algorithm removes multiple occurrences of vertex i in order of decreasing saving, as follows

$$saving = d_{pred\ i} + d_{i\ succ} - d_{pred\ succ} \tag{3.20a}$$

where *pred* and *succ* are, respectively, the vertex preceding and following i on the considered route. Multiple occurrences of vertex i are removed from x_{sub} by decreasing values of saving, calculated according to the formula (3.20a), until the number of occurrences is equal to 1. After the removal of multiple occurrences, vertices not present in x_{sub} are inserted exactly once; the pruning heuristic assigns non-visited vertices considering one route at a time, until its total demand exceeds the vehicle capacity Q; if all routes exceed Q and we still have unrouted vertices, the algorithm keeps on creating and filling a new route, adding it to x_{sub}, until there are no unrouted vertices.

When all occurrences of vertices are equal to 1, the second step of the pruning heuristic takes place, (line 3), asking to normalize the number of used vehicles of the current solution x'. Since the SP (3.1) formulation asks for a solution with a fixed

number of vehicles m, we implement an ejection chain procedure able to delete routes from solution x', until the number of remaining routes is equal to m. Routes are considered for deletion in ascending order of number of vertices, to have a lower number of vertices to be relocated in x'.

If the ejection chain procedure does not succeed in obtaining a solution with exactly m routes, algorithm 4 returns an empty solution, (line (7)); otherwise, the third step of the pruning heuristic takes place (line (5)). At this step, a VND local search is executed on the current solution x'' to search a feasible solution with a lower total traveled distance. The VND is made by several neighborhoods, both intra-route and inter-route, applied in ascending order of size. The first improvement strategy is adopted during the exploration of each neighborhood. Neighborhoods are not explored exhaustively, concentrating the exploration only on vertices involved in successful moves, (*don't look bits* strategy (Nagata and Bräysy (161))). Each neighborhood exploration is executed for a maximum time limit. Inter-route neighborhoods composing the VND are 2-opt*, neighborhoods based on λ-interchanges (Osman (163)) and Cross-exchange (Taillard et al. (206)). 2-opt* is a neighborhood involving exchanges of couple of arcs between a couple of routes. Considered λ-interchanges are Shift(1,0), Shift(2,1) and Swap(1,1), where, respectively, one vertex is removed from a route r_1 and inserted in another route r_2, two consecutive vertices are removed from a route r_1 and inserted in another route r_2 and a couple of vertices belonging to two different routes r_1 and r_2 are exchanged. Intra-route neighborhoods composing VND are 2-opt and Or-opt2. VND first applies 2-opt and Or-opt2 intra-route neighborhoods on every route of the current solution x''; subsequently, inter-route neighborhoods are applied, in the order, 2-opt*, Shift(1,0), Swap(1,1), Shift(2,1) and Cross-exchange. VND continues its search until an improving solution is found with a percentage of improvement bigger than 1% with respect to the value of the previous best solution found; otherwise, VND local search is stopped.

VND terminates its execution returning the best solution found x_F, at line 5 of algorithm 4. The pruning heuristic returns solution x_F to the Lagrangean heuristic, at line 8 of algorithm 3; here we call this solution s'. The value of the best solution ever found s_{best} by the Lagrangean heuristic is possibly updated to s' at line 9, according to the minimum value of total traveled distance. At line 10 the counter of subgradient iterations is updated.

Subgradient optimization is executed until the maximum number of iterations is not reached or until a user-defined time limit is not exceeded.

The Lagrangean heuristic we propose can be defined to all effects a metaheuristic, as TS or ILS; in fact, it is an iterative higher-level method designed and able to find, generate and use other heuristics with the aim of identifying feasible good-quality solutions. Moreover, it is an example of usage of MP techniques, (such as CG or subgradient optimization), as a basis to define heuristic frameworks and, at the same time, identify feasible minimum cost solutions for the problem to be treated.

3.4 Computational results

In this section we report computational results of the matheuristic described previously in this chapter. The matheuristic was coded in C++ and tests were executed on an Intel$^{\mathrm{R}}$ Core$^{\mathrm{TM}}$ i7 with 3.60GHz and 32 GB of RAM running under Windows Server 2012 64 bits. CPLEX 12.6.1 was used as LP solver for the master RSP problem. To achieve speed up in computation times, we implemented (q, i)-route and ng-route bounding procedures, respectively, using CUDA parallel computing platform and OpenMP, as done by Strappaveccia (202) (see websites (1) and (5) for information related to CUDA and OpenMP).

We set the time limit for the execution of the CG based on (q, i)-route and on ng-route bounding procedures both to 5000 seconds. The time limit for the execution of subgradient optimization is set to 3600 seconds, while its maximum number of iterations is set to 400. The value of α is set to 1.5. The maximum time limit for the exploration of each neighborhood of VND is set to 7 seconds.

We fix the cardinality of sets N_i $\forall i \in V$ of ng-route bounding procedure to 8, i.e. each set N_i is composed by vertex i and by the 7 vertices nearest to i.

To assess performances of our matheuristic related to the quality of both lower bound and feasible solution computed, we use two datasets. One is the relatively new dataset proposed by Uchoa et al. (211) in 2014, available at the website (3). This benchmark is composed by 100 instances, with a size ranging from 100 to 1000 vertices. The name of each instance is formatted as X-nA-kB, where A represents the number of vertices of the instance including the depot, and B is the minimum possible number of vehicles. The average route size is different for every instance. The positioning

of the depot is randomly chosen among a central, eccentric or random position; the distribution of remaining vertices is randomly chosen among a random, clustered or random-clustered distribution. Several options were chosen for the demand distribution to be used. The other dataset used for experimenting the matheuristic consists in a new dataset, generated by us, composed by 6 instances derived from actual practice in freight transportation. Traveled distances, demands of vertices and the capacity of vehicles are all real world data. The size of the instances ranges from 179 to 980 vertices; the capacity is expressed in terms of volume (dm^3), weight (kg) or pallets; distances are expressed in terms of time; the number of vehicles to be used is fixed at runtime to the number of routes belonging to the feasible CVRP solution computed by the constructive heuristic executed at line 8 of algorithm 2. The dataset is available at the website (7).

Computational results of the matheuristic for instances by Uchoa et al. (211) are presented in table A.1. Column *Instance* denotes the name of the solved instance, while n represents the number of vertices excluding the depot. Columns 3-5 are related to the calculated valid lower bound; column 3 is the number of used vehicles, column 4 represents the value of the lower bound and column 5 corresponds to the percentage gap of the lower bound from the related best known solution calculated as $Gap(\%) = ((BestKnown - LowerBound) * 100)/BestKnown$. Columns 6-8 report the best feasible solution identified by the matheuristic during the subgradient optimization; column 6 represents the number of used vehicles, columns 7 is the value of the total traveled distance of the feasible solution and column 8 is the percentage gap of the solution from the corresponding best known solution calculated as $Gap(\%) = ((BestKnown - Heu.) * 100)/BestKnown$. Column 9 is the total execution time of the matheuristic, expressed in seconds. Columns 10-11 report the best known solution of the instance, i.e. the number of used vehicles and the value of the total traveled distance. Computational results of the matheuristic for instances by (7) are presented in table A.2. Column *Instance* denotes the name of the solved instance, while n represents the number of vertices excluding the depot. Columns 3-4 report the calculated lower bound value, i.e. column 3 represents the number of used vehicles, while column 4 is the value of the total traveled distance. Columns 5-6 report the value of the best identified feasible solution during subgradient optimization; column 5 is the number of used vehicles and column 6 is the value of the total traveled distance. Column 7 represents the percentage

gap of the lower bound from the value of the best identified feasible solution. The last column represents the total execution time of the matheuristic, expressed in seconds. Dash entries in columns 3-4 for both tables mean that the corresponding values have not been calculated by the algorithm, because the CG based on (q, i)-route procedure has exceeded the time limit of 5000 seconds, hence not obtaining a valid lower bound.

For what concerns instances by Uchoa et al. (211) the number of vehicles to be used was fixed according to the number of used vehicles in the optimal solution, to have a proper comparison of percentage gaps for both lower bound and heuristic solutions. If we look at table A.1 we can see that the quality of the calculated lower bound of instances is quite good for all instances in the dataset. We have in fact that only for 14 out of 100 instances the matheuristic was not able to calculate a valid lower bound; in these cases even the CG based on (q, i)-route bounding procedure did not succeed in calculating a valid lower bound, because of its time limits. If we consider the remaining 86 instances, we have an average percentage lower bound gap lower than 2%, that represents a good result. For what concerns the quality of the best identified feasible solution of the matheuristic we have higher gaps from best known solutions. For instances with size lower than 200 vertices, the average gap is lower than 4%, while this datum increases to a value lower than 7% if we consider instances with size comprised between 200 and 600 vertices. The percentage gap further increases being comprised between 7% and 9% for instances with size bigger than 600 vertices. If we consider the whole dataset we have an average gap of 6.15%.

Let us look at table A.2 for what concerns instances by (7). We can see that a valid lower bound was not found for the 2 biggest instances. Since we do not know a-priori what is the best solution for these instances, the calculated percentage gaps are all related to the comparison between the valid lower bound and the value of the best identified feasible solution. We have gaps lower than 8% for instances with size lower than 500 vertices, while the gap increases to a 20% for instances with a number of vertices bigger than 500.

4

Parameter tuning of a Lagrangean heuristic and an ILS

4.1 Introduction to the problem of parameter tuning

Many algorithms designed to solve optimization problems base their functioning on the instantiation of a set of parameters. The design of each optimization algorithm is the result of a series of choices, made to obtain the best possible performance. In this context we can hence identify the problem of defining the parameter *configurations* of an optimization algorithm, that permit to obtain optimized empirical performances on a given set of problem instances to be solved. We call this problem the *parameter tuning* problem, defined as follows, as stated by Hoos (133):

Given

- an algorithm A with parameters p_1, ..., p_k that affect its behaviour,
- a space C of configurations, where each configuration $c \in C$ specifies values for A's parameters such that A's behaviour on a given problem instance is completely specified (up to possible randomisation of A),
- a set of problem instances I,
- a performance metric m that measures the performance of A on instance set I for a given configuration c,

find a configuration $c' \in C$ that results in the optimal performance of A on I according to metric m.

Designers and simple users of parameterized algorithms very often encounter the problem of tuning parameters, in order to optimize the empirical performances of algorithms in solving a given set of problem instances. This problem can be faced by the use of automatic methods, that analyze in a clever manner the possible configurations of an optimization algorithm, choosing the best one. The use of automatic procedures aims at finding the configuration of an optimization algorithm giving the best possible performances over a set of instances. The present chapter treats the problem of tuning parameters of optimization algorithms by using automatic methods and presents three applications. The structure of the chapter is as follows. In section 4.2 we show some automatic methods for parameter tuning. In section 4.3 we present the problem of tuning the parameters of a Lagrangean metaheuristic, used to solve both the CVRP and the VRPTW. In section 4.4 we show the problem of tuning the parameters of an ILS heuristic applied to solve the QAP.

4.2 Automatic methods for parameter tuning

Several automatic approaches have been proposed in the literature to deal with the problem of parameter tuning. We introduce *offline* configuration methods. These approaches are made by two different phases. The first phase, called *training*, chooses an algorithm configuration, given a set of instances called *training set*, representative of the particular problem that has to be solved by the target algorithm. During the second phase, called *test*, the chosen candidate algorithm configuration is used to solve an unseen *test set* of instances of the same problem. The aim is to identify, during the training phase, a candidate configuration that minimizes some cost measure over the set of instances that will be seen during the test phase.

Among offline configuration methods there are *racing procedures*. At the basis of racing there is the idea of sequentially evaluating candidate configurations on given benchmark instances, and delete candidates as soon as they are too far behind the current leader candidate, i.e. the candidate with the overall best performance at a certain stage of the race. Birattari et al. (44) proposed the F-Race algorithm that

closely follows the racing procedure. To overcome limitations that affect the basic F-Race approach, some evolutions of F-Race have been proposed. One of these is called Iterative F-Race (I/F-Race) (Balaprakash et al. (26), Birattari et al. (45)). The key idea of this method is that of using an iterative process where, in the first stage of each iteration, configurations are sampled from a probabilistic model M, while in the second stage a standard F-Race is performed on the resulting sample; configurations that survive the race are used to define and update the model M used in the following iteration.

4.2.1 The irace package

The irace package is an automatic offline configurator of optimization algorithms, implementing the *iterated racing* procedure, an extension of the I/F-Race presented by Balaprakash et al. (26) and developed by Birattari et al. (45).

Iterated racing is composed by three steps:

1. sampling new configurations according to a particular distribution
2. selecting the best configurations from the newly sampled ones by means of racing
3. updating the sampling distribution in order to bias the sampling towards the best configurations.

Each parameter to be tuned has its independent sampling distribution. When distributions are updated, sampling distributions are modified for biasing the distributions to increase the probability of sampling, in future iterations, the values of the parameters of the best configurations found. After the sampling of new configurations, the best configurations are selected by means of racing. Each race begins with a finite set of candidate configurations. During each step of the race, the candidate configurations are evaluated on a single instance. After each step, the candidate configurations that perform statistically worse than at least another candidate are discarded; the race continues with the remaining candidates. This iterative procedure continues until a minimum number of surviving candidates is reached, a maximum number of instances has been used or a predefined computational *budget* is reached, (the computational budget may correspond to a computation time or to the number of *experiments*, where an experiment identifies the application of a configuration to an instance). For further

details about the implementation of iterated racing in the irace package we refer to López-Ibánez et al. (151).

irace is distributed as an R package, built upon the race package by Birattari (43). It has been applied for configuring several optimization algorithms. Dubois-Lacoste et al. (100) and Dubois-Lacoste et al. (101) used irace to tune parameters of IG for solving the permutation FSP. A bi-objective TSP was treated by López-Ibáñez and Stützle (150) through automatic configuration of an ACO framework. de Oca et al. (86) automatically configured a particle swarm optimization method for large-scale continuous optimization problems.

4.3 Parameter tuning of a Lagrangean metaheuristic

In this section we present a Lagrangean metaheuristic algorithm to solve both the CVRP and the VRPTW. We show the problem of its tuning, using the irace package with the aim of improving the calculated valid lower bounds.

4.3.1 Target problems

Target problems of the Lagrangean metaheuristic are the CVRP and the VRPTW, (see sections 3.1 and 3.2 of chapter 3 for an introduction and definition of the CVRP). The VRPTW belongs to the family of VRPs and it represents one of the most studied NP-hard problems of the VRP family.

First works on the VRPTW date back to the 1960's, (see Golden and Assad (123), Lenstra et al. (145) and Desrosiers et al. (93) for surveys treating early developments of the VRPTW). The first exact algorithm proposed for the VRPTW was the B&P by Desrochers et al. (92), later improved by Kohl et al. (141) through the addition of 2-path inequalities to the LP relaxation of the SP formulation. Among exact approaches for the VRPTW, we mention the ones proposed by Kohl and Madsen (140), Irnich and Villeneuve (136), Jepsen et al. (137) and Desaulniers et al. (91).

4.3.2 Mathematical formulations and relaxations

The VRPTW is defined on a complete digraph $G = (V', A)$, where $V' = \{0, 1, ..., n\}$ is a set of $n + 1$ vertices and A is the arc set. Vertex 0 corresponds to the depot, and we define the vertex subset $V = V' \setminus \{0\}$ composed by n vertices. Each vertex $i \in V'$ has

an associated demand q_i, (we assume $q_0 = 0$), and a time window $[e_i, l_i]$, where e_i and l_i represent the earliest and latest time to visit i. Each arc $(i, j) \in A$ has an associated travel cost d_{ij} and a travel time $t_{ij} > 0$, the latter including the service time at vertex i, so the departure time at any vertex $i \in V$ coincides with the end of its service. We indicate with D the matrix of travel costs d_{ij}, and with T the matrix of travel times t_{ij}. We indicate with $\Gamma_i \subseteq V'$ the set of *successors* of i in G and with $\Gamma_i^{-1} \subseteq V'$ the set of *predecessors* of i in G, $\forall i \in V'$.

A fleet of m identical vehicles of capacity Q available at the depot has to serve vertices. We indicate with $R = (0, i_1, ..., i_r, 0)$, with $r \geq 1$, a vehicle route; each vehicle route R is a simple circuit in G passing through the depot, visiting vertices $V(R) = \{0, i_1, ..., i_r\}$, $V(R) \subseteq V'$, and such that (i) the total demand of the visited vertices does not exceed the vehicle capacity Q; (ii) the vehicle leaves the depot 0 at time e_0, visits each vertex in $V(R)$ within its time window, and returns to the depot before l_0; (iii) if the vehicle arrives at $i \in V(R)$ before e_i, the service is delayed to time e_i. Each vehicle route R has a cost equal to the sum of the travel costs of the arc set, $A(R)$, traversed by route R. Both the CVRP and the VRPTW ask for the design of a set of at most m routes of minimum total cost, such that each vertex is visited exactly once by exactly one route, respecting all constraints linked to the visit.

Let $\mathscr{R}$ be the index set of all feasible routes, and let $\mathscr{R}_i \subset \mathscr{R}$ be the index set of routes covering vertex $i \in V'$. The cost associated to each route $\ell \in \mathscr{R}$ is $c_\ell = \sum_{(i,j)\in A(\ell)} d_{ij}$. Let x_ℓ, $\ell \in \mathscr{R}$, be a (0-1) binary variable equal to 1 if and only if route ℓ is in the optimal solution. The formulation of both the CVRP and the VRPTW can be based on the following SP model with additional constraints

$$(SP) \quad z(SP) = \min \sum_{\ell \in \mathscr{R}} c_\ell x_\ell \tag{4.1a}$$

$$\text{s.t.} \quad \sum_{\ell \in \mathscr{R}_i} x_\ell = 1, \quad \forall i \in V \tag{4.1b}$$

$$\sum_{\ell \in \mathscr{R}_0} x_\ell \leq m, \tag{4.1c}$$

$$x_\ell \in \{0, 1\}, \quad \forall \ell \in \mathscr{R} \tag{4.1d}$$

Constraints (4.1b) impose that each vertex $i \in V$ has to be visited by exactly one route. Constraint (4.1c) specifies that at most m routes have to be selected.

Let $u = (u_1, ..., u_n)$ be the unrestricted vector of dual variables, where u_i $i = 1, ..., n$ are associated to constraints (4.1b). Let v be the non-positive dual variable associated to constraint (4.1c). Hence, the dual problem of the LP relaxation of SP can be defined as follows

$$(DSP) \qquad z(DSP) = \max \sum_{i \in V} u_i + mv \tag{4.2a}$$

$$\text{s.t.} \quad \sum_{i \in \mathrm{V}(\ell) \setminus 0} u_i + v \leq c_\ell, \quad \forall \ell \in \mathscr{R} \tag{4.2b}$$

$$u_i \text{ unrestricted}, \quad \forall i \in V \tag{4.2c}$$

$$v \leq 0, \tag{4.2d}$$

A *forward* path $P = (0, i_1, ..., i_{k-1}, i_k)$ for the VRPTW is an elementary path starting from the depot 0 at time e_0, visiting vertices $V(P) = \{0, i_1, ..., i_{k-1}, i_k\}$ within their time windows, and ending at vertex $i_k = \sigma(P)$ at time $t(P)$ with $e_{\sigma(P)} \leq t(P) \leq l_{\sigma(P)}$. Let us denote by $A(P)$ the set of arcs traversed by P and by $c(P) = \sum_{(i,j) \in A(P)} d_{ij}$ the cost of path P.

(t, i)-path is a well-known relaxation of forward paths. A (t, i)-path is a non-necessarily elementary path starting from the depot at time e_0, visiting a set of vertices within their time windows, and ending at the vertex i at time $e_i \leq t \leq l_i$. (t, i)-path relaxation ignores the vehicle capacity constraint. The cost $f(t, i)$ of the least cost (t, i)-path can be computed using DP as described by Christofides et al. (71). A (t, i)-route is a $(t, 0)$-path visiting at time t the last vertex i before arriving at the depot. (t, i)-path relaxation can easily avoid 2-cycles, i.e. cycles like $(0, i_1, ..., i_{j-1}, i_j, i_{j+1}, ..., i_{k-1}, i_k)$ where $i_{j-1} = i_{j+1}$. It is possible to demonstrate that $f(t, i)$ is a valid lower bound on the cost $c(P)$ of any forward path P, such that $t(P) = t$ and $\sigma(P) = i$.

We can define the *ng*-path relaxation for the VRPTW. The definition of sets $\Pi(P)$ is the same as (3.4a), where $P = (0, i_1, ..., i_{k-1}, i_k)$ is a forward path. A forward *ng*-path (NG, t, i) is a non-necessarily elementary path $P = (0, i_1, ..., i_{k-1}, i_k = i)$ starting

from the depot at time e_0, visiting a subset of vertices within their time windows such that $NG = \Pi(P)$, ending at vertex i at time $e_i \leq t \leq l_i$, and suche that $i \notin \Pi(P')$, where $P' = (0,\ i_1,\ ...,\ i_{k-1})$. We denote by $f(NG,\ t,\ i)$ the cost of the least cost forward ng-path $(NG,\ t,\ i)$. We define an $(NG,\ t,\ i)$-route as an $(NG,\ t,\ 0)$-path visiting at time t the last vertex i before arriving at the depot. The cost of the $(NG,\ t,\ i)$-route of minimum cost is given by $f(NG,\ t,\ i) + d_{i0}$. We can compute functions $f(NG,\ t,\ i)$ using DP as follows. Let $\Omega(t,\ j,\ i)$ be the subset of departure times from vertex j to arrive at vertex i at time t when j is visited immediately before i, i.e. (i) $\Omega(t,\ j,\ i) = \{t' : e_j \leq t' \leq \min\{l_j, t - t_{ji}\}\}$ if $t = e_i$, and (ii) $\Omega(t,\ j,\ i) = \{t - t_{ji} : e_j \leq t - t_{ji} \leq l_j\}$ if $e_i < t \leq l_i$. The state space graph $\mathscr{H} = (\mathscr{E}, \Psi)$ is defined as

$$\mathscr{E} = \{(NG,\ t,\ i) : \forall NG \subseteq N_i \text{ s.t. } NG \ni i,\ \forall t,\ e_i \leq t \leq l_i,\ \forall i \in V'\}, \tag{4.3a}$$

$$\Psi = \{((NG',\ t',\ j), (NG,\ t,\ i)) : \forall (NG',\ t',\ j) \in \Psi^{-1}(NG,\ t,\ i), \forall (NG,\ t,\ i) \in \mathscr{E}\}, \tag{4.4a}$$

where $\Psi^{-1}(NG,\ t,\ i) = \{(NG', t',\ j) : \forall NG' \subseteq N_j \text{ s.t. } NG' \ni j \text{ and } NG' \cap N_i = NG \setminus \{i\},\ \forall t' \in \Omega(t,\ j,\ i),\ \forall j \in \Gamma_i^{-1}\}$.

The DP recursion for calculating $f(NG,\ t,\ i)$ is the following

$$f(NG,\ t,\ i) = \min_{(NG',\ t',\ j) \in \Psi^{-1}(NG,\ t,\ i)} \{f(NG',\ t',\ j) + d_{ji}\}, \forall (NG,\ t,\ i) \in \mathscr{E}. \tag{4.5a}$$

It is possible to demonstrate that $f(NG,\ t,\ i)$ is a valid lower bound to the cost $c(P)$ of any forward path P, such that $t(P) = t$ and $\sigma(P) = i$, (Baldacci et al. (31)).

4.3.3 The Lagrangean metaheuristic

We describe the Lagrangean metaheuristic. The method is a matheuristic able to produce both a valid lower bound and a feasible solution using MP methods as a basis for ameliorating the search of good quality feasible solutions. The objective of the algorithm is both computing tight lower bounds and feasible solutions minimizing total travel costs, and, at the same time, minimizing the number of used vehicles.

Algorithm 5 reports a pseudocode of the matheuristic. The algorithm is a subgradient iterative method, based on a reduced SP model, RSP, obtained from the SP model 4.1 by replacing the route set $\mathscr{R}$ with the set $\mathscr{R}'$. The subgradient solves the Lagrangean dual problem, computing a near optimal solution of the reduced dual problem, RDSP, obtained from model DSP 4.2 by replacing the route set $\mathscr{R}$ with the set $\mathscr{R}'$. Hence, the subgradient calculates a valid lower bound on the optimal cost $z(SP)$. We relax in a Lagrangean fashion both SP constraints (4.1b) and (4.1c). At each iteration of the subgradient, a CG method relying on a bounding procedure enlarges the set of routes $\mathscr{R}'$, computing negative reduced cost columns, respectively, (q, i)-routes or (NG, q, i)-routes for the CVRP, or (t, i)-routes or (NG, t, i)-routes for the VRPTW. During each iteration of the subgradient, a pruning heuristic is used to fix infeasibilities of the subgradient solution and compute a feasible solution for the problem to be solved. The matheuristic framework is the same for both the CVRP and the VRPTW; only the pruning heuristic and the procedure to calculate a starting feasible solution differ, since they have to solve different problems.

At line 2 of algorithm 5, the pool of columns $Pool_C$ of the master RSP model is initialized to an empty set. Each entry of the matrix of reduced costs $DRed(i, j)$ is initialized with travel cost $D(i, j)$, $\forall (i, j) \in A$ at line 3. The counter of subgradient iterations, the counter of the number of consecutive subgradient iterations without an improvement of the lower bound and a boolean value indicating if a valid lower bound has been computed are initialized, respectively at lines 4, 5 and 6. The value of the Lagrangean dual is initialized to 0, at line 7. At line 8 a starting feasible solution is computed. For what concerns the CVRP, the method is the same as the one used to initialize the core of columns of the master RSP problem in algorithm 2. For the VRPTW, we used a sequential constructive method, relying on the i1 insertion heuristic proposed by Solomon (200) as criterion for inserting vertices in routes. At line 9 we initialize the pool of columns $Pool_C$ of the master RSP problem with single vertex routes (0, i, 0), $\forall i \in V$. Lines 10-37 represent the heart of the Lagrangean metaheuristic. Let $Pool_{C_i} \subset Pool_C$ be the subset of routes covering vertex $i \in V$. We apply the parametric relaxation for the SC model, shown by Boschetti and Maniezzo (51), to our RSP model. Following this relaxation, the resulting mathematical formulation of the RSP problem is

Algorithm 5 Lagrangean Metaheuristic

```
1: procedure LAGR_META(D, T, m, n, q_i, Q, α_in, update_rate, red_rate, it_tot)
2:     Pool_C ← ∅
3:     DRed(i, j) ← D(i, j)   ∀i, j ∈ V'
4:     cont_it ← 0
5:     count_it_no_impr ← 0
6:     valid_lower_bound ← false
7:     lb_b ← 0
8:     s_best ← Create_UB(D, T)
9:     Pool_C ← Init_Master()
10:    α ← α_in
11:    λ_i ← 0   ∀i ∈ V'
12:    repeat
13:        g ← Solve_Lagr_Dual(Pool_C, λ_i, m, lb)
14:        if lb ≤ lb_b then
15:            count_it_no_impr ← count_it_no_impr + 1
16:            if count_it_no_impr = update_rate then
17:                α = α * red_rate
18:                count_it_no_impr ← 0
19:                if α = 0 then
20:                    α ← α_in
21:            if valid_lower_bound = false then
22:                goto 25
23:        else
24:            count_it_no_impr ← 0
25:            DRed ← Calc_Red_Costs(D, g)
26:            new_pool ← Pricing_Cols(DRed, T, n, q_i, Q)
27:            if new_pool = ∅ then
28:                if lb > lb_b then
29:                    valid_lower_bound ← true
30:                    lb_b ← lb
31:            else
32:                Pool_C ← Pool_C ∪ new_pool
33:        x_sub ← Update_Penalties(Pool_C, λ_i, α, m, lb)
34:        s' ← Pruning_Heuristic(x_sub, D, T)
35:        s_best ← Update_Sol(s', D)
36:        cont_it ← cont_it + 1
37:    until (cont_it < it_tot   ||   time_limit_not_exceeded)
38:    return s_best
```

$$(PRSP(w)) \qquad z(PRSP)(w) = \min \sum_{\ell \in Pool_C} \sum_{i \in V(\ell)\setminus\{0\}} c_\ell \frac{w_i}{w(V(\ell))} y_\ell^i \tag{4.6a}$$

$$\text{s.t.} \quad \sum_{\ell \in Pool_{C_i}} \sum_{h \in V(\ell)\setminus\{0\}} \frac{w_h}{w(V(\ell))} y_\ell^h = 1, \quad \forall i \in V \tag{4.6b}$$

$$\sum_{\ell \in Pool_{C_0}} \sum_{h \in V(\ell)\setminus\{0\}} \frac{w_h}{w(V(\ell))} y_\ell^h \leq m, \tag{4.6c}$$

$$y_\ell^i \in \{0,1\}, \quad \ell \in Pool_{C_i}, \quad i \in V \tag{4.6d}$$

We relax in a Lagrangean fashion both PRSP(w) constraints (4.6b) and (4.6c). Consider a *penalty* vector $\lambda = (\lambda_1\ ,...,\ \lambda_n,\ \lambda_{n+1})$, where λ_i, $i = 1, ..., n$ is an unrestricted real number associated to constraint (4.6b) for vertex $i \in V$ and $\lambda_{n+1} \geq 0$ is associated to constraint (4.6c). We obtain the following problem

$$(LPRSP(\lambda, w))z(LPRSP)(\lambda, w) = \min \sum_{\ell \in Pool_C} \sum_{i \in V(\ell)\setminus\{0\}} (c_\ell - \lambda'(V(\ell))) \frac{w_i}{w(V(\ell))} y_\ell^i + \sum_{i \in V} \lambda_i - m\lambda_{n+1} \tag{4.7a}$$

$$y_\ell^i \in \{0,1\}, \quad \ell \in Pool_{C_i}, \quad i \in V \tag{4.7b}$$

where $\lambda'(V(\ell)) = \lambda(\mathrm{V}(\ell)) - \lambda_{n+1}$ and $\lambda(\mathrm{V}(\ell)) = \sum_{h \in V(\ell)\setminus\{0\}} \lambda_h$.

Problem LPRSP(λ, w) is decomposable into n subproblems, one for each row $i \in V$

$$(LPRSP^i(\lambda,\ w)) \qquad z^i(LPRSP)(\lambda,\ w) = \min \sum_{\ell \in Pool_{C_i}} c_\ell^i(\lambda,\ w) y_\ell^i + \lambda_i \tag{4.8a}$$

$$\text{s.t.} \quad y_\ell^i \in \{0,\ 1\}, \quad \ell \in Pool_{C_i} \tag{4.8b}$$

where $c_\ell^i(\lambda,\ w) = (c'_\ell - \lambda(V(\ell)))\frac{w_i}{w(V_\ell)}$ and $c'_\ell = c_\ell + \lambda_{n+1}$.

We set $w_i = \lambda_i$ and add the constraint $\sum_{\ell \in Pool_{C_i}} y_\ell^i = 1, \forall i \in V$. The subproblem LPRSP$^i(\lambda,\ w),\ i \in V$ can be rewritten as follows

$$(LPRSP^i(\lambda)) \qquad z^i(LPRSP)(\lambda) = \min \sum_{\ell \in Pool_{C_i}} c'_\ell \frac{\lambda_i}{\lambda(V(\ell))} y^i_\ell \tag{4.9a}$$

$$\text{s.t.} \quad \sum_{\ell \in Pool_{C_i}} y^i_\ell = 1, \tag{4.9b}$$

$$y^i_\ell \in \{0,\ 1\}, \quad \ell \in Pool_C \tag{4.9c}$$

Hence, the overall value of the Lagrangean problem LPRSP(λ) is

$$z(LPRSP)(\lambda) = \sum_{i \in V} z^i(LPRSP)(\lambda) - m\lambda_{n+1} \tag{4.10a}$$

Any optimal solution of problem LPRSP(λ) provides a feasible solution (u, v) of cost $z(LPRSP)(\lambda)$ for the reduced dual problem RDSP, obtained from the DSP model by replacing the route set $\mathscr{R}$ with the pool $Pool_C$. A feasible dual solution (u, v) of cost $z(LPRSP)(\lambda)$ for problem RDSP can be obtained by means of the following expressions

$$u_i = \min_{\ell \in Pool_{C_i}} \{c'_\ell \phi_{i\ell}\} \quad i \in V \tag{4.11a}$$

$$v = -\lambda_{n+1} \tag{4.11b}$$

where $c'_\ell = c_\ell + \lambda_{n+1}$ and

$$\phi_{i\ell} = \begin{cases} \frac{\lambda_i}{\lambda(V(\ell))} & \lambda(V(\ell)) > 0 \\ \frac{1}{|V(\ell) \setminus \{0\}|} & \lambda(V(\ell)) = 0 \end{cases} \tag{4.12a}$$

The best lower bound that can be achieved using expressions (4.11) is equal to the optimal solution cost $z(RDSP)$ of the problem RDSP; this value can be obtained calculating the maximum of the function $z(LPRSP)(\lambda)$ with respect to λ, i.e.

$$\max_\lambda \{z(LPRSP)(\lambda)\} = z(RDSP) \tag{4.13a}$$

The problem (4.13) is called Lagrangean dual. We need to solve it to find the optimal (or near-optimal) dual solution of cost $z(RDSP)$. The implemented subgradient method deals with the Lagrangean dual, searching the space of possible values for λ vectors and obtaining the best possible lower bound. At line 11 each component of the λ vector is initialized to 0, $\forall i \in V'$. Loop 12-37 is the core of subgradient optimization. At line 13 the Lagrangean dual problem (4.13) is solved for the given λ vector. Lines 14-22 test if the value of the current lower bound is better than the current value of the Lagrangean dual. If it is not, the counter of the number of consecutive iterations without improvement is incremented and, possibly, the value of the parameter α, used to update the vector λ, is updated. Then the algorithm generates new negative reduced cost routes, if a valid lower bound has not yet been found, (line 22). In the case that the current value of the lower bound is better than the current value of the Lagrangean dual, the matrix of reduced costs is updated, (line 25), and the pricing of new negative reduced costs routes is done, (line 26). If no negative reduced costs routes can be found, a valid lower bound has been found, and we update the value of the Lagrangean dual, (line 30). Otherwise, we enlarge the pool of columns $Pool_C$, (line 32). At line 33 the subgradient vector is calculated and used to update the vector λ. Let us indicate with $J \subset Pool_C$ the index subset of routes that produce minima of formula (4.11a) $\forall i \in V$, i.e. $J = \{\ell \in Pool_C : \ell = \text{argmin}_{\ell \in Pool_{C_i}}[c'_\ell \phi_{i\ell}], i \in V\}$. Let (u, v) be the dual solution of cost $z(LPRSP)(\lambda)$ computed by expressions (4.11) at point λ. Let x be the corresponding non-necessarily feasible solution of RSP, computed as

$$x_\ell = \begin{cases} \sum\limits_{i \in I_\ell} \phi_{i\ell} & \ell \in J \\ 0 & otherwise \end{cases} \tag{4.14a}$$

where $I_\ell = \{i \in V : u_i = c'_\ell \phi_{i\ell}\}$. A valid subgradient of the function $z(LPRSP)(\lambda)$ is given by the vector $\theta = (\theta_1, ..., \theta_n, \theta_{n+1})$, calculated according to the following formulas

$$\theta_i = 1 - \sum_{\ell \in Pool_{C_i}} x_\ell, \quad i \in V \tag{4.15a}$$

$$\theta_{n+1} = m - \sum_{\ell \in Pool_{C_0}} x_\ell \tag{4.15b}$$

The vector of Lagrangean penalties λ is then updated according to the following formulas

$$\lambda_i = \lambda_i + \alpha \frac{0.1lb}{\sum_{j \in V'} \theta_j^2} \theta_i, \quad i \in V \tag{4.16a}$$

$$\lambda_{n+1} = \max\{0, \lambda_{n+1} - \alpha \frac{0.1lb}{\sum_{j \in V'} \theta_j^2} \theta_{n+1}\} \tag{4.16b}$$

Solution x, calculated following formulas (4.14), is returned at line 33; x is referenced in the pseudocode as x_{sub}.

At line 34, a pruning heuristic is invoked to fix infeasibilities of the solution x_{sub}.

For what concerns the CVRP, the pruning heuristic works as follows. Using the solution x_{sub}, the heuristic selects the m routes having the corresponding highest values in x_{sub}. Since vertices can appear more than once in the selected m routes, among all the routes in which a vertex appears, the heuristic assigns each vertex to the route with the corresponding highest value in the solution x_{sub}. If some vertices are not present in the selected m routes, they are inserted in the current solution until the capacity constraint of each route is not violated. If the heuristic succeeds in assigning each vertex exactly once, a VND method as the one presented in section 3.3.2 of chapter 3 is executed, to try to improve the current feasible solution.

The pruning heuristic for the VRPTW works as follows. The heuristic selects the m routes of x_{sub} having the corresponding highest values. For each one of these routes, the heuristic deletes multiple occurrences of vertices, maintaining only the first occurrence of each vertex. Multiple occurrences of vertices along different routes are eliminated by maintaining for each vertex, among all the routes in which the vertex appears, only the occurrence corresponding to the route having the highest value in the solution x_{sub}. The respect of the capacity constraint is tested on each of the resulting routes; if the constraint is violated for some routes, the necessary deletions of vertices from the routes themselves are made. The remaining unrouted vertices are inserted in the current partial solution following the i1 criterion by Solomon (200). If the heuristic succeeds in inserting each vertex in the solution exactly once, a VND local search is executed, using the same neighborhoods presented for the CVRP in section 3.3.2 of chapter 3, adjusted for treating time window constraints.

After the execution of the pruning heuristic, at line 35 the current best feasible solution ever found s_{best} is possibly updated to the feasible solution s', if the total traveled distance is lower, for an equal number of used vehicles, or if the number of used vehicles is lower. The subgradient method stops when the maximum number of iterations is reached or when a time limit is exceeded. The Lagrangean metaheuristic returns the best feasible solution found s_{best}.

4.3.4 The parameter tuning problem

Let us consider parameter α of algorithm 5. This parameter is used to update the vector of Lagrangean penalties λ, as shown by formulas (4.16). Since it is involved in the update of λ, α is hence responsible for the solution of the Lagrangean dual problem, that has the aim of computing the highest possible valid lower bound searching the space of the vectors λ. The value of α varies during subgradient iterations, adapting to the improvement of the value of the current lower bound. An initial predefined value is established for α, (line 10). If for a certain consecutive number of iterations the value of the current lower bound does not improve the value of the Lagrangean dual, the value of α is reduced of a predefined rate, (line 17), and, when the value of α is very close to 0, the initial predefined value is reassigned to α, (line 20). Hence, we can identify three higher level parameters that control the updating of α, i.e.

- the initial value of α, α_{in};
- the number of consecutive subgradient iterations without improvement of the lower bound, $count\_it\_no\_impr$;
- the reduction rate of α, red_rate.

We experimented if it could be possible to improve the quality of the valid lower bounds computed by the subgradient method through a proper tuning of these three parameters. The considered applications of the Lagrangean metaheuristic's parameter tuning deal with the solution of the CVRP and the VRPTW. We use the irace package to treat this parameter tuning problem. The Lagrangean metaheuristic was coded in C++ and all computational tests have been executed on an Intel$^{\mathrm{R}}$ Core$^{\mathrm{TM}}$ i3 with 2.40GHz and 4 GB of RAM, running under Windows 7 64 bits.

The first application we analyze is the one related to the CVRP. The subgradient is stopped after 800 iterations, or if the time limit of 6 hours is reached. We fix the cardinality of sets $N_i \; \forall i \in V$ of ng-route bounding procedure to 7, (each set N_i is composed by vertex i and by the 6 vertices nearest to i). Table 4.1 summarizes the characteristics of the three parameters to be tuned, i.e. the name of the parameters, the type, and the ranges of values chosen for tuning each parameter; the table reports the initial chosen ranges of values for the tuning of the parameters.

Table 4.1: Characteristics of the initial chosen ranges for parameters related to α

Name	Type	Range
α_{in}	Real	[1,0, 2,0]
count_it_no_impr	Integral	[15, 30]
red_rate	Real	[0,2, 0,6]

The used default configuration, i.e. the non-tuned configuration, for the three parameters to be tuned is shown in table 4.2.

Table 4.2: Default configuration of the parameters related to α

α_{in}	*count_it_no_impr*	*red_rate*
1,5	15	0,4

The test set of instances of the CVRP we are interested to solve is the one proposed by Uchoa et al. (211), (we consider only instances with a total number of vertices lower than 400, because of memory problems in managing bigger sizes). The training set used by irace during its training phase has been created from scratch, by considering the first 50 vertices for each instance in the test set and using them to generate a new corresponding instance to be added to the training set. We decided not to use the other test sets from the literature of the CVRP since, in our opinion, the created training set can summarize quite well characteristics of the test set, while the other test sets are quite different from each other, hence not providing a proper basis for the tuning. We implemented several pricing procedures for the CG, i.e. the ng-route, the (q, i)-route and the (q, i)-route with 2-cycles bounding procedures, (this last one corresponds to the (q, i)-route procedure not avoiding the creation of 2-cycles in routes). Hence, we

executed three separate parameter tuning sessions of irace, each one dedicated to the tuning of the Lagrangean metaheuristic with the corresponding pricing procedure. The established number of experiments of irace for each session is 1000.

The tuned configurations of the three parameters obtained by irace, using the ranges shown in table 4.1, for, respectively, the *ng*-route, the (q, i)-route and the (q, i)-route with 2-cycles pricing for solving the CVRP did not improve the quality of the valid lower bounds with respect to the use of the default configuration of table 4.2. This is documented in table 4.3, that shows the percentage average lower bound gaps from best known solutions, obtained, respectively, with the default configuration and with the tuned configurations, for every pricing procedure. We can see that the gaps obtained using the tuned configurations are bigger than the gaps obtained by the default configuration.

Table 4.3: Comparison between percentage average lower bound gaps from best known solutions before and after tuning for instances by (211)

Before tuning			**After tuning**		
***ng*-route**	**(q, i)-route**	**(q, i)-route 2-cycles**	***ng*-route**	**(q, i)-route**	**(q, i)-route 2-cycles**
13,56	12,84	5,26	21,44	9,58	6,21

Given the obtained results, we analyze if it could be possible to obtain better quality valid lower bounds, for solving the CVRP, by changing the considered ranges of values, used by irace for the tuning of the Lagrangean metaheuristic. Hence, we tried to enlarge the considered ranges, shown in table 4.1, by the creation of new ranges for the tuning; these ranges are reported in table 4.4.

Table 4.4: Characteristics of the enlarged ranges for parameters related to α

Name	**Type**	**Range**
α_{in}	Real	[0,7, 2,5]
count_it_no_impr	Integral	[10, 30]
red_rate	Real	[0,1, 0,7]

The tuning of the Lagrangean metaheuristic with irace for, respectively, the *ng*-route, the (q, i)-route and the (q, i)-route with 2-cycles pricing for solving the CVRP,

using the enlarged ranges of table 4.4, gave as output the configurations of parameters shown in tables 4.5, 4.6 and 4.7.

Table 4.5: Tuned configuration of the parameters related to α for ng-route pricing for the CVRP

α_{in}	*count_it_no_impr*	*red_rate*
0,73	29	0,65

Table 4.6: Tuned configuration of the parameters related to α for (q, i)-route pricing for the CVRP

α_{in}	*count_it_no_impr*	*red_rate*
0,86	22	0,67

Table 4.7: Tuned configuration of the parameters related to α for (q, i)-route with 2-cycles pricing for the CVRP

α_{in}	*count_it_no_impr*	*red_rate*
1,1	26	0,63

The obtained computational results related to the calculated valid lower bounds before and after the tuning of parameters for, respectively, the ng-route, the (q, i)-route and (q, i)-route with 2-cycles pricing, using the configurations of tables 4.5, 4.6 and 4.7, are reported in tables B.1, B.2 and B.3. The first column of each table shows the name of the solved instance. The second column represents the number of vertices of the instance, (excluding the depot). Columns 3-5 report computational results of the valid lower bound obtained by the Lagrangean metaheuristic before the tuning of parameters, i.e. the number of used vehicles, the value of the total traveled distance and the percentage gap from the best known solution, calculated as $Gap(\%) = ((BestKnown - LowerBound) * 100)/BestKnown$. Columns 6-8 report computational results of the valid lower bound obtained by the Lagrangean metaheuristic after the tuning of parameters, i.e. the number of used vehicles, the value of the total traveled distance and the percentage gap from the best known solution. Column 9 reports the total execution time in seconds of the algorithm. Columns 10-11 show details of the best known solution, i.e. the number of used vehicles and the total traveled

distance, (best known solution values are taken from the website (3)). Underlined entries of the tables identify the values improved by the tuning. To facilitate the comprehension of the results and understand the effectiveness of the tuning process, we summarize the obtained results through correlation plots. The x coordinate of each point in the correlation plot represents the value of the lower bound obtained before the tuning, while the y coordinate is the value of the lower bound after the tuning. Figures B.7, B.8 and B.9 show correlation plots of, respectively, ng-route, (q, i)-route and (q, i)-route with 2-cycles pricing. By looking at correlation plots, we can see that, generically, points lie along or over the bisection line, meaning that performances obtained with tuned configurations are better than results obtained before tuning. We can verify this fact also by looking at table 4.8, that shows percentage average lower bound gaps from best known solutions before and after the tuning for every pricing procedure. We can see that average lower bounds significantly decrease after the tuning of parameters, meaning that performances considerably improve. We can, hence, note that the obtained results are significantly better than the results obtained with the tuned configurations, calculated by irace using the first proposed ranges of values of table 4.1. This demonstrates that the enlarging of the ranges considered by the tuning can potentially find new, better configurations, able to significantly improve the performances of the algorithm to be tuned; moreover, this shows that, often, the human intuition of limiting the search of good configurations to a restricted neighborhood of the default used configuration can be a bad choice, prefering, hence, to consider wider neighborhoods of parameter configurations. For completeness of explanation, tables B.4, B.5 and B.6 report the related heuristic computational results of the Lagrangean metaheuristic, obtained before and after the tuning of the parameters, (dash entries in the tables mean that the pruning heuristic did not succeed in assigning each vertex exactly once, hence not obtaining a feasible solution).

Table 4.8: Comparison between percentage average lower bound gaps from best known solutions before and after tuning for instances by Uchoa et al. (211)

Before tuning			**After tuning**		
***ng*-route**	**(q, i)-route**	**(q, i)-route 2-cycles**	***ng*-route**	**(q, i)-route**	**(q, i)-route 2-cycles**
13,56	12,84	5,26	5,31	4,72	5,53

To assess the performances of the tuned algorithm, we tested the tuned Lagrangean metaheuristic also on instances of test sets A, B, P, (Augerat et al. (24)), E, (Christofides and Eilon (70)), and M, (Christofides et al. (69)); all these instances are available at website (2)). Figures B.10, B.11 and B.12 show correlation plots comparing valid lower bounds computed with the default configuration and tuned configurations, for each of the three pricing procedures. Table 4.9 shows percentage average lower bound gaps before and after the tuning with irace. These results confirm the effectiveness of the tuning process of irace in improving the quality of the valid lower bounds. Corresponding detailed lower bound computational results can be found in tables B.7, B.8 and B.9, while tables B.10, B.11 and B.12 report heuristic results of the Lagrangean metaheuristic.

Table 4.9: Comparison between percentage average lower bound gaps from best known solutions before and after tuning for instances A, B, P, E, M

Before tuning			**After tuning**		
***ng*-route**	**(q, i)-route**	**(q, i)-route 2-cycles**	***ng*-route**	**(q, i)-route**	**(q, i)-route 2-cycles**
3,51	5,6	9,23	3,51	5,61	9,22

The second application of the Lagrangean metaheuristic parameter tuning deals with the solution of the VRPTW. The subgradient is stopped after 700 iterations, or if the time limit of 6 hours is reached. We fix the cardinality of sets $N_i \; \forall i \in V$ of ng-route bounding procedure to 7, i.e. each set N_i is composed by vertex i and by the 6 vertices nearest to i. The default configuration for the three parameters to be tuned is shown in table 4.2. The test sets of instances of the VRPTW we are interested to solve are the 100 vertices test set proposed by Solomon (200) and the 200 vertices test set by Gehring & Homberger's benchmark, available at (8). The training set used by irace is composed by the 25 and the 50 vertices instances derived from instances by Solomon (200); they are composed by, respectively, the first 25 and 50 vertices of each instance by Solomon (200), available at (9). As for the CVRP, we executed three separate parameter tuning sessions of irace, each one dedicated to the tuning of the Lagrangean metaheuristic with the corresponding pricing procedure. The established total number of experiments of irace for each session is 1000. The considered ranges of values for the parameters to be tuned are the ones reported in table 4.1.

The tuning of irace of the Lagrangean metaheuristic with, respectively, the *ng*-route, the (t, i)-route and the (t, i)-route with 2-cycles pricing for solving the VRPTW gave as output the following configurations, shown, respectively, in tables 4.10, 4.11 and 4.12.

Table 4.10: Tuned configuration of the parameters related to α for *ng*-route pricing for the VRPTW

α_{in}	*count_it_no_impr*	*red_rate*
1,29	25	0,33

Table 4.11: Tuned configuration of the parameters related to α for (t, i)-route pricing for the VRPTW

α_{in}	*count_it_no_impr*	*red_rate*
1,17	17	0,63

Table 4.12: Tuned configuration of the parameters related to α for (t, i)-route with 2-cycles pricing for the VRPTW

α_{in}	*count_it_no_impr*	*red_rate*
1,01	17	0,61

Comparisons between the computational results of the valid lower bounds obtained with tuned configurations of parameters and with the default one, previously shown in table 4.2, for the Lagrangean metaheuristic with, respectively, the *ng*-route, the (t, i)-route and the (t, i)-route with 2-cycles pricing for, respectively, tests sets by Solomon (200) and Gehring & Homberger are reported in tables B.13, B.14, B.15, B.16, B.17 and B.18. The first column of each table shows the name of the solved instance. The second column reports the number of vertices of the instance, (excluding the depot). Columns 3-5 report computational results of the valid lower bound obtained by the Lagrangean metaheuristic before the tuning of parameters, i.e. the number of used vehicles, the value of the total traveled distance and the percentage gap from the best known solution, calculated as $Gap(\%) = ((BestKnown - LowerBound) * 100)/BestKnown$. Columns 6-8 report computational results of the valid lower bound obtained by the Lagrangean

metaheuristic after the tuning of parameters, i.e. the number of used vehicles, the value of the total traveled distance and the percentage gap from the best known solution. Column 9 reports the total execution time in seconds of the algorithm. Columns 10-11 show details of the best known solution, i.e. the number of used vehicles and the total traveled distance, (best known solution values for, respectively, the instances by Solomon (200) and by Gehring & Homberger are taken from (9) and (8)). To facilitate the comprehension of the results and to understand the effectiveness of the tuning process, we summarize the obtained results through correlation plots. Figures B.1, B.2 and B.3 show correlation plots for valid lower bounds of the instances by Solomon (200) before and after tuning with irace for, respectively, the ng-route, the $(t,\ i)$-route and the $(t,\ i)$-route with 2-cycles pricing. As you can see by looking at correlation plots, lower bound values obtained with the default configuration, i.e. before tuning, are worse than values obtained after tuning with irace, for each of the three pricing procedures; in fact, we have that the majority of points of each correlation plot lies over or along the bisection line. We can verify this fact also by looking at table 4.13, that shows percentage average lower bound gaps from best known solutions before and after tuning for every pricing procedure.

Table 4.13: Comparison between percentage average lower bound gaps from best known solutions before and after tuning for instances by Solomon (200)

Before tuning			**After tuning**		
***ng*-route**	**$(t,\ i)$-route**	**$(t,\ i)$-route 2-cycles**	***ng*-route**	**$(t,\ i)$-route**	**$(t,\ i)$-route 2-cycles**
4,7	7,3	13,73	4,7	6,47	11,27

The improvement of the quality of the valid lower bound is confirmed also by tests made on 200 vertices instances by Gehring & Homberger. Figures B.4, B.5 and B.6 show the distribution of valid lower bounds, while table 4.14 reports percentage average lower bound gaps from best known solutions before and after tuning for every pricing procedure. For completeness of explanation, tables B.19, B.20, B.21, B.22, B.23, B.24 report the detailed computational results related to the heuristic solution computed by the Lagrangean metaheuristic before and after the tuning of the parameters.

Table 4.14: Comparison between percentage average lower bound gaps from best known solutions before and after tuning for instances by Gehring & Homberger

Before tuning			After tuning		
***ng*-route**	**(t, i)-route**	**(t, i)-route 2-cycles**	***ng*-route**	**(t, i)-route**	**(t, i)-route 2-cycles**
35,41	38,01	53,04	35,43	34,93	43,51

We tested if it could be possible to further improve the obtained results, by executing the tuning of the parameters using the enlarged ranges of values, proposed in table 4.4, used for tuning the Lagrangean metaheuristic for the CVRP. We executed the tuning process with irace, using the enlarged ranges, but the results we obtained did not significantly improve the already obtained results; for this reason, we do not report these results.

4.4 Parameter tuning of an ILS

In this section we present an ILS to solve the QAP. We show the problem of the tuning of ILS, using the irace package, with the aim of improving the value of the feasible solution computed by ILS.

4.4.1 Target problem

The QAP is one of the most difficult combinatorial optimization problems. The problem asks to assign a set of n facilities to a set of n locations, given distances between the locations and flows between the facilities; the aim is that of assigning each facility to exactly one location and viceversa, in such a way that the sum of the product between flows and distances is minimal. Let $\Phi = \{1, ..., n\}$ be an index set of the facilities and $\Lambda = \{1, ..., n\}$ be an index set of the locations. Furthermore, let $D = [d_{ih}]$, $i, h = 1$, ..., n be the matrix of distances between each pair of locations and let $F = [f_{jk}]$, $j, k = 1$, ..., n be the matrix of flows between each pair of facilities. A 0/1 binary variable x_{ij} takes value 1 if facility i is assigned to location j, 0 otherwise. We can define the QAP as the following problem

$$(QAP) \qquad z(QAP) = \min \sum_{i,j=1}^{n} \sum_{h,k=1}^{n} d_{ih} f_{jk} x_{ij} x_{hk} \tag{4.17a}$$

$$\text{s.t.} \quad \sum_{i=1}^{n} x_{ij} = 1, \quad j = 1, ..., n \tag{4.17b}$$

$$\sum_{j=1}^{n} x_{ij} = 1, \quad i = 1, ..., n \tag{4.17c}$$

$$x_{ij} \in \{0, 1\} \tag{4.17d}$$

Many practical problems like backboard wiring (201), hospital layout (105) and many others can be represented as QAPs. The theoretical and practical interest of the problem has given rise during the years to several algorithms, both exact and heuristic. The most effective exact techniques presented in the literature include those of Mautor and Roucairol (158), Hahn et al. (127) and Brüngger et al. (56). Among heuristic approaches we recall the SA of Connolly (76), the TS of Taillard (205) and of Battiti and Tecchiolli (35), the GRASP of Pardalos and Resende (164), the Ant System algorithm of Maniezzo (155) and the ILS of Stützle (204).

4.4.2 The ILS

ILS is an iterative method that uses an embedded heuristic to build a sequence of solutions, leading to far better solutions than if one were to use repeated random trials of that heuristic (152). The key idea of ILS is that of using an iterative mechanism alternating the phases of local search and perturbation, with the aim of achieving a good trade-off between intensification and diversification of the search within the space of solutions of the problem to be treated. Intensification has to be guaranteed by the local search, permitting to identify local minima, while diversification is made by the perturbation, that moves the search towards possibly unexplored areas of the solution space. We can summarize the main steps of ILS by the following algorithm 6.

The algorithm builds an initial solution at line 2; at line 3 a local search procedure is executed to try to improve the quality of the initial solution s_0. Loop 4-8 is the heart of ILS, alternating the perturbation and the local search phase. The acceptance

Algorithm 6 Iterated Local Search

```
1: procedure ILS
2:     s_0 ← Generate_Initial_Solution()
3:     s* ← Local_Search(s_0)
4:     do
5:         s' ← Perturbation(s*, history)
6:         s*' ← Local_Search(s')
7:         s* ← Acceptance_Criterion(s*, s*', history)
8:     while termination condition not met
9:     return s*
```

criterion at line 7 decides if consider or not the current solution as a starting point for the new perturbation/local search iteration. The algorithm ends when the termination condition is satisfied, giving as output the best solution ever found $s*$.

4.4.3 The parameter tuning problem

Let us consider algorithm 6. The design of the ILS depends on the instantiation of three different parameters, that give rise to different implementations of the algorithm:

- the choice of the local search procedure;
- the choice of the perturbation procedure;
- the choice of the acceptance criterion.

Different implementations of the ILS lead to different computed feasible solutions, of better or worse quality. The problem we treat is that of identifying the proper design of the ILS, able to produce the best-quality feasible solutions possible. The target problem of the ILS we are interested to solve is the QAP. As for the Lagrangean metaheuristic, we treat the problem of tuning the ILS using the irace package.

The move that we consider in applying a local search procedure for the QAP is the exchange of two elements of a given solution. The local search parameter can be instantiated using one of the following local search procedures:

- first improvement hill climbing with equal neighbor comparator; it is a first improvement hill climbing that considers as improving neighbor a solution with a fitness lower than or equal to the fitness of the current solution;

- first improvement hill climbing with non equal neighbor comparator; it is a first improvement hill climbing that considers as improving neighbor a solution with a fitness lower than the fitness of the current solution;
- best random hill climbing with equal neighbor comparator; it is a procedure that randomly chooses, at each iteration, one of the best solutions in the neighborhood and updates the current solution if the fitness of the chosen neighbor is lower than or equal to the fitness of the current solution;
- best random hill climbing with non equal neighbor comparator; it is a procedure that randomly chooses, at each iteration, one of the best solutions in the neighborhood and updates the current solution if the fitness of the chosen neighbor is lower than the fitness of the current solution;
- simple hill climbing with equal neighbor comparator; it is a best improvement hill climbing that considers as improving neighbor a solution with a fitness lower than or equal to the fitness of the current solution;
- pure first improvement 2-opt; it executes 2-opt exchanges between elements of a solution, accepting the first found improvement;
- pure best improvement 2-opt; it executes 2-opt exchanges between elements of a solution, accepting the best found improvement;
- tabu search, using best improvement 2-opt.

The perturbation parameter can be instantiated using one of the following procedures:

- restart; this perturbation consists in reinitializing at random the solution when a maximum number of iterations with no improvement is reached;
- multiple exchange; this perturbation is realized by multiple exchanges of elements in the current solution;
- repeated multiple perturbation; it consists in applying a perturbation many times in a row.

The acceptance criterion parameter can be realized by one of the following criteria:

- accept always; it always accepts a solution;
- accept better; it accepts a solution if it is better than the current one;
- accept better or equal; it accepts a solution if it is better than or equal to the current one;
- accept simulated annealing; it permits to accept even worse solutions, only with a certain probability.

All computational tests have been executed on a machine with an Intel^{R} Core^{TM} i3 with 2.40 GHz and 4 GB of RAM, running under Windows 7 64 bits.

The ILS was coded in C++, using the ParadisEO framework, (Cahon et al. (58)), to design the different procedures of local search, perturbation and acceptance criteria. The initial solution s_0 of the ILS corresponds to the permutation (1, ..., n). The termination criterion of the ILS corresponds to a maximum time limit, expressed in seconds, equal to the size of the instance to be solved divided by 5.

The used default configuration of the ILS consists in using:

- pure first improvement 2-opt as local search procedure;
- multiple exchange as perturbation;
- accept simulated annealing as acceptance criterion.

The test set of instances of the QAP we are interested to solve is composed by the instances (tai27e01-tai27e20), (tai45e01-tai45e20), (tai75e01-tai75e20), (tai125e01-tai125e20), (tai175e01-tai175e20), (tai343e01-tai343e20) and (tai729e01-tai729e10) proposed by Taillard, (available at (6)), and by the structured instances proposed by Pellegrini et al. (166).

Two separate tuning sessions of irace have been conducted on the ILS, one addressed to improve the performances of the ILS in solving the test set by Taillard and the other one for the test set by Pellegrini et al. (166).

For what concerns the instances by Taillard, we composed the training set used by irace with a half of the instances of each set (tai27e01-tai27e20), (tai45e01-tai45e20), (tai75e01-tai75e20), (tai125e01-tai125e20), (tai175e01-tai175e20) and (tai343e01-tai343e20).

The test set is composed by the remaining instances. The established number of experiments of irace is 50000.

The tuning of the ILS addressed to solve the instances by Taillard gave as output the following configuration of parameters:

- pure best improvement 2-opt as local search procedure;
- repeated multiple perturbation as perturbation, where the multiple exchange perturbation is executed for a random number of times, (the number of executions is chosen in the interval [6,26]);
- accept simulated annealing as acceptance criterion.

Obtained computational results related to the best identified feasible solutions of the ILS before and after the tuning of the parameters are reported in table B.25. The first column reports the name of the solved instance. The second column shows the number of facilities and locations of the instance. Columns 3-4 report computational results of the best feasible solution found by the non-tuned ILS, i.e., respectively, the value of the objective function and the percentage gap from the best known solution. Columns 5-6 report computational results of the best feasible solution found by the tuned ILS, i.e., respectively, the value of the objective function and the percentage gap from the best known solution. Column 7 shows the value of the best known solution of the corresponding instance, (dash entries mean that the best known solution is not known). To facilitate the understanding of the results and the effectiveness of the tuning process, we summarize the obtained results through the correlation plot, shown in figure B.13. Here, the x coordinate of each point in the correlation plot corresponds to the value of the objective function obtained by the non-tuned ILS, while the y coordinate represents the value of the objective function obtained by the tuned ILS. We can observe that the majority of the points in the correlation plot lies along or under the bisection line; this means that the tuning of the ILS has improved the quality of the best feasible solutions found. This assertion is also confirmed by the data reported in table 4.15, that shows percentage average heuristic gaps from best known solutions obtained before and after the tuning with irace; we have, in fact, that the percentage average gap obtained by the non-tuned ILS is higher than the gap obtained by the tuned ILS, of more than one percent.

Table 4.15: Comparison between percentage average heuristic gaps from best known solutions before and after tuning for instances by Taillard

Before tuning	After tuning
14,64	11,15

For what concerns the structured instances by Pellegrini et al. (166), we composed the training set used by irace with a half of the instances of each set of structured instances of size 60, 80 and 100; the test set is composed by the remaning instances. The established number of experiments of irace is 50000.

The tuning of the ILS addressed to solve the instances by Pellegrini et al. (166) gave as output the following configuration of parameters:

- pure best improvement 2-opt as local search;
- repeated multiple perturbation as perturbation; multiple exchange perturbation is executed for an increasing number of applications, starting from 10 to 14;
- accept always as acceptance criterion.

Figure B.14 and table 4.16 summarize the comparison between heuristic computational results obtained by the ILS before and after the tuning with irace. By looking at the correlation plot we can see that, for the most part, points lie along the bisection line. Table 4.16 shows percentage average heuristic gaps from best known solutions obtained before and after the tuning with irace; we can see that the tuned ILS gives as output slightly better heuristic results with respect to the non-tuned version of the algorithm. The detailed obtained computational results related to the best identified feasible solutions of the ILS before and after the tuning of the parameters are reported in table B.26.

Table 4.16: Comparison between percentage average heuristic gaps from best known solutions before and after tuning for instances by Pellegrini et al. (166)

Before tuning	After tuning
0.92	0.57

5

A scheduling predictive model for the management of a warehouse

5.1 Introduction

In this chapter we deal with a real-world application to the case of a warehouse of tiles located in Thailand. The warehouse is used both as a storage and as a distribution point to commercialize tiles. The problem asks to minimize the duration of the queues of process of a set of resources, operating in the warehouse. We developed a model representing the daily work flow that characterizes the warehouse. The aim of the developed model is that of providing a scenario, in which the utilization of the resources available in the warehouse is maximized, in such a way that the duration of the queues of process can be minimized. To design such a model of the daily work flow, we developed a heuristic algorithm, representing the operating of the resources in the warehouse, with the aim of maximizing the degree of utilization of the available resources.

This chapter is organized as follows. In section 5.2 we report the physical organization of the warehouse, detailing the different areas and resources of the warehouse, and the logical organization of its daily work flow. In section 5.3 we deeply analyze the treated problem and we show the developed heuristic method to deal with the introduced problem. In section 5.4 we comment some output data of the algorithm, in the light of what seen in the previous sections.

5.2 The warehouse: organization and functioning

The warehouse we are interested to deal with stores and organizes the distribution of tiles. Tiles physically lie in the warehouse inside boxes. Boxes are physically grouped together, to form *full pallets*.

The warehouse is subdivided in *storage areas*. Each storage area is logically divided into *blocks*. Each block permits the storage of full pallets in several *positions*. Because of the kind of good, the full pallets of tiles are stored, within each position, as stacks, hence, following a LIFO policy for their removal.

The *preparation area* is the site of the warehouse where pallets are moved for being subjected to the quality checks of the goods, before their delivery to the customers. The preparation area is divided in *locations*, each one with a known capacity to store full pallets, expressed in terms of the number of stockable full pallets.

A so called *ready to ship area* is available in the warehouse; full pallets are moved from the preparation area towards this site to be grouped together to form the shipments of delivery to customers, and to be physically loaded on trucks for the delivery.

In the warehouse, a site called *picking bay* is present; within this place, the composition of full pallets from several non-full pallets happens. The picking bay has associated a so called *picking bay buffer* area, where the non-full pallets requested by the picking bay are temporarily stored.

If some non-full pallets remain in the picking bay buffer and these are useless for other orders, these are moved towards a site called *racks*, where there is the storage of non-full pallets.

The warehouse has also a site where it is possible to manually pick goods, that here we call *trim area*.

The warehouse is equipped with several resources, that guarantee its operating. The great part of the resources is composed by *forklifts*. Each forklift has a set of areas and roles of competence, that define its operating in the warehouse, (e.g., a forklift can operate in the storage areas 1, 2, and 5, for transporting full pallets from one of these areas to the preparation area). The forklifts of the warehouse are often shared by several areas, and a role can be covered by many forklifts. The goods of the warehouse can be also manually picked by human operators, that carry them from the so called trim area to supply the picking bay.

Besides storing the tiles, the warehouse is also responsible for the management of the orders and the organization of the delivery of the goods to the customers. The so called Warehouse Management System, (WMS), is the system that daily deals with the management of the orders and the organization of the deliveries. During the working day, the WMS receives the requests of goods by the customers. The orders are composed by many articles, each one of these specifying the requested quantity of goods, expressed in terms of the number of required boxes of tiles.

To satisfy and complete the articles of the orders, the WMS creates a set of so called *missions*. Each mission corresponds to the movement of goods within the warehouse. The WMS divides the requested quantity of goods by the full pallet quantity of the article; the result corresponds to the necessary number of full pallets, while the remainder is the quantity for picking. The WMS creates as many missions as the number of necessary full pallets of the article; these missions correspond to the carriage of a full pallet, matching the request of the article, from a particular storage area of the warehouse to the preparation area. The remainder quantity is lower than the quantity composing a full pallet of the article, and these goods are not directly carried towards the preparation area.

The assembly of the remainder quantity is made within the picking bay area. In this case, the goods are moved towards the picking bay, passing through the picking bay buffer. The management system of the picking bay buffer verifies the pending picking order requests in the picking bay and the available stock present both in the picking bay buffer and in the picking bay, possibly generating the transfers to supply the picking bay buffer. If the pending picking order requests of the picking bay cannot be satisfied by the currently available stock of both the picking bay buffer and the picking bay, the corresponding missions are created to make the transfers of goods. If the corresponding article is required for the first time, a transfer of a full pallet from the storage areas of the warehouse towards the picking bay buffer happens; otherwise, a transfer from the racks, (the site of the warehouse with non-full pallets), to the picking bay buffer is required.

When the full pallet (or the non-full pallet) arrives at the picking bay buffer, the requested remainder quantity of goods is brought to the picking bay; here, according to the orders, the remainder quantities are assembled together to form a new Unit Load, (UL), that can be carried from the picking bay to the preparation area. If there is

leftover content of the used full pallet (or the non-full pallet) at the picking bay buffer, and this content is useless for the other pending picking order requests, it is carried from the picking bay buffer to the racks.

Besides using the picking bay buffer to provide non-full quantities of goods, the manual picking can happen; here a human operator manually brings the requested quantity of an article from the trim area to the picking bay, where this last one will be assembled together with the other non-full quantity of product, according to the orders.

When the goods arrive at the preparation area, (be it from the picking bay or directly from the storage areas), the products are subjected to quality checks, to assess they are not damaged. At the end of these controls, the goods are ready to be delivered to customers. The WMS organizes the deliveries of the goods creating proper *shipments*. A shipment is composed by many orders grouped together. The composition of the shipments is started at the preparation area; here, the pallets belonging to the same shipment are placed in contiguous locations, following a LIFO policy for their removal, (let us point that each location of the preparation area can contain, at the same time, only pallets that belong to exactly one shipment). When all pallets belonging to the same shipment are present in the preparation area and all quality checks are ended, the pallets of the shipment are carried to the ready to ship area, where the physical loading of the pallets on the trucks is made.

Figure 5.1 provides a schema of the functioning of the warehouse.

The movement of a pallet, (be it full or non-full), involving the storage areas, the picking bay buffer, the racks, the preparation area and the ready to ship area is managed as a mission. Missions are assigned to forklifts. The WMS creates and, successively, assigns the missions to be done to the proper forklifts, according to their area and role of competence. One mission can be executed by exactly one forklift. Each forklift can manage a priority queue of missions, according to which the mission that is the first in the queue is the first mission to be executed, and following the other ones. The queue has a predefined length, beyond which the forklift cannot take charge of other missions.

During the working day, the WMS continuously receives requests of goods by the customers. These requests are scheduled by the WMS through the creation of proper missions, to complete the orders. When a request is scheduled, the created missions enter in a state called *waiting*, since they have to wait for a proper available forklift

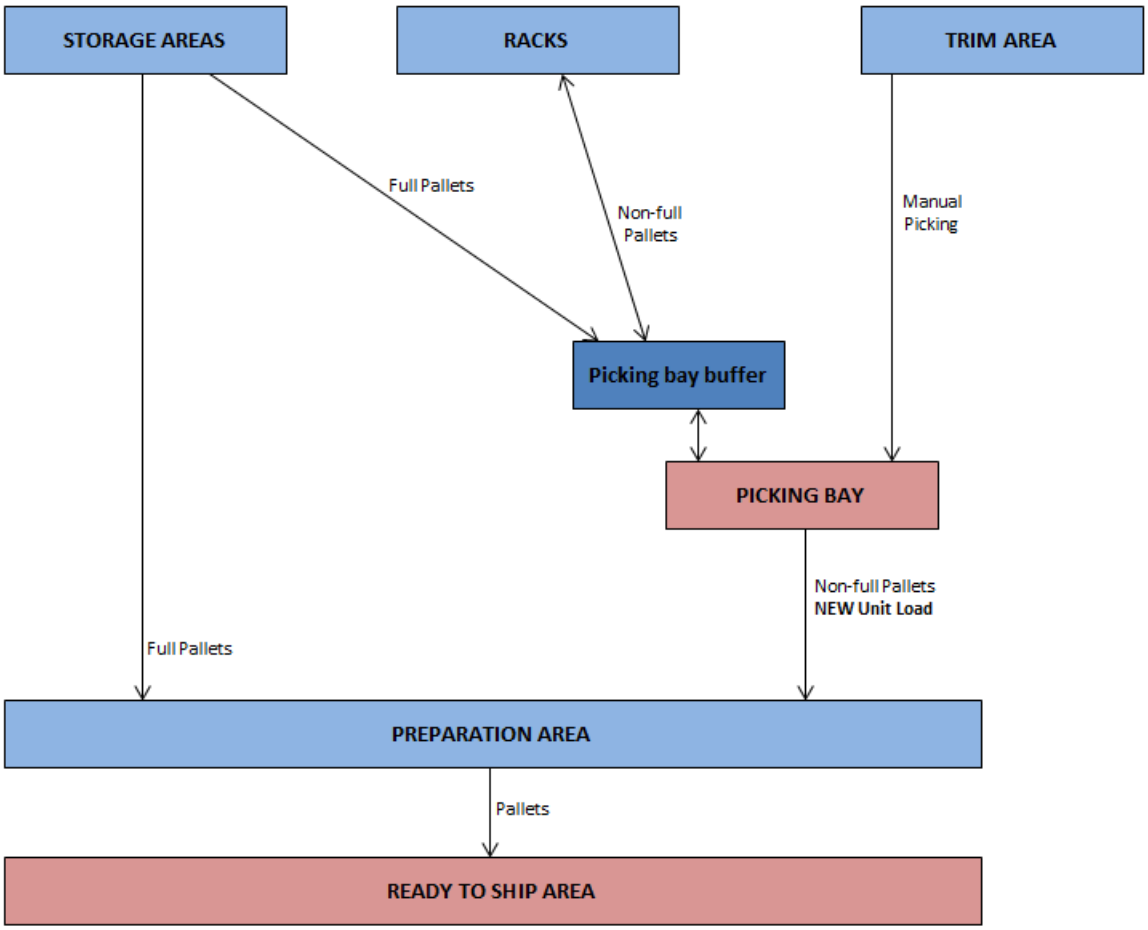

Figure 5.1: Schema of the functioning of the warehouse

to execute them. As soon as the proper forklifts are available, the assignment of the missions to the proper forklifts is made, following the order of scheduling of the related requests, i.e. the missions that are assigned before are the ones associated to the requests scheduled before. When a mission is assigned to a forklift, it enters the last position of the priority queue of the forklift, making a state transition, from waiting to *active*. Each forklift executes the missions according to their order in its priority queue.

Before scheduling the new arriving requests, the WMS executes several checks on the status of the operating of the warehouse. These checks aim at controlling both the current levels of stored goods and the degree of occupation of the locations of the preparation area. If the levels of stored goods are sufficient to satisfy the new arriving requests and the degree of occupation of the preparation area permits the preparing of new shipments, the new arriving requests are scheduled through the creation of missions, (see the paragraph above for the waiting and active missions). If the levels are not sufficient or the degree of occupation of the preparation area is high, the schedule of the requests does not happen; it is postponed to the moment in which there will be the right conditions for both the level of stored goods in the storage areas and the occupation of the preparation area. These non-scheduled requests compose the so called *portfolio* of the orders, i.e. the set of the orders of the customers waiting for their execution. In this case, a priority order is given to all non-scheduled requests of the portfolio, according to their order of arrival to the WMS. The scheduling of these requests will happen when right conditions will be met, following the priority order.

5.3 The problem and the proposed algorithm

In section 5.2 we detailed the internal functioning of the warehouse of interest. One of the issues arising in the context of the operating of the warehouse deals with the minimization of the duration time of the queues of the missions of each available forklift of the warehouse. We developed a model representing the daily work flow that characterizes the warehouse. The aim of the developed model is that of providing a scenario, in which the utilization of the forklifts available in the warehouse is maximized, in such a way that the duration of the queues of missions can be minimized. To design such a

model of the daily work flow, we developed a heuristic algorithm, representing the operating of the forklifts in the warehouse. We tried to minimize the duration time of the queues of missions through the achievement of two basic conditions of the algorithm, i.e.

- the maximization of the degree of use of the available forklifts;
- the early schedule of the non-scheduled requests.

The algorithm must be able to predict, at any time during the working day, the duration of the queues of missions, knowing the operating status of the available forklifts at that time, the set of the current active and waiting missions and the set of non-scheduled requests of the portfolio, present in the WMS at that time. The result of the prediction can be used to measure the real performances of the available resources of the warehouse. This can be a method for identifying possible delays or inefficiencies of the real functioning of the warehouse, hence providing a key performance indicator of the system, for improving the quality of the daily work flow of the warehouse.

The algorithm for predicting the duration of the queues is, in its essence, a pure heuristic approach, dealing with the schedule and the assignment of the missions to the compatible available forklifts, that can perform the requested carriage of goods. At the basis of the algorithm there is the modeling of the concept of mission. Since the algorithm has to take into account also the non-scheduled requests of the portfolio, and the related missions have not yet been created by the WMS, we estimated them by creating as many missions as the number of necessary full pallets to complete the requests. We define the state of these missions as *forecast*. The actual version of the algorithm models the carriage of full pallets both from the storage areas to the preparation area and from the preparation area to the ready to ship area. The algorithm developed so far does not consider the management of the picking bay and the trim areas, hence keeping into account only the forklifts as resources. The actual version of the algorithm considers only the degree of occupation of the preparation area as criterion for the scheduling of the requests of the portfolio, (the algorithm assumes that the levels of stored goods are always sufficient for satisfying the requests). The output of the algorithm consists in the log of the operations made by each forklift, and by the timestamps of the movements of goods within the preparation area.

We base the functioning of the algorithm on the modeling of the mission. This concept can be articulated through the representation of the *life cycle* of the mission. The life cycle of the mission is defined by the different states a mission can assume and by the transitions that regulate each change of state. A summary of the life cycle of the mission can be the following.

"Each forecast mission has to be assigned to a forklift, that performs it. When the mission has been performed through the carriage of the corresponding full pallet to the preparation area, the forklift starts waiting for an empty position in the preparation area, where to store the carried pallet. As soon as a position is found, the forklift stores the pallet and starts the execution of the following mission, while the stored pallet waits for the completion of the related shipment. When the shipment is completed, the locations occupied by the shipment are emptied by appropriate forklifts; they execute the corresponding missions to carry the pallets of the completed shipment towards the ready to ship area. When all the pallets of the shipment arrive there, they are physically loaded on the trucks for the delivery to the customers."

Figure 5.2 shows the detailed life cycle of the mission at the base of the developed algorithm. The rectangles represent the states in which a mission can stand, while the rhombuses correspond to the events that induce the corresponding change of state. The explanation and the understanding of the life cycle permit the explanation of the developed algorithm. In the following subsections, we will discuss the life cycle of the mission, by showing the details of the transitions of the states. Each paragraph is dedicated to the explanation of one or more transitions of the life cycle, shown in figure 5.2.

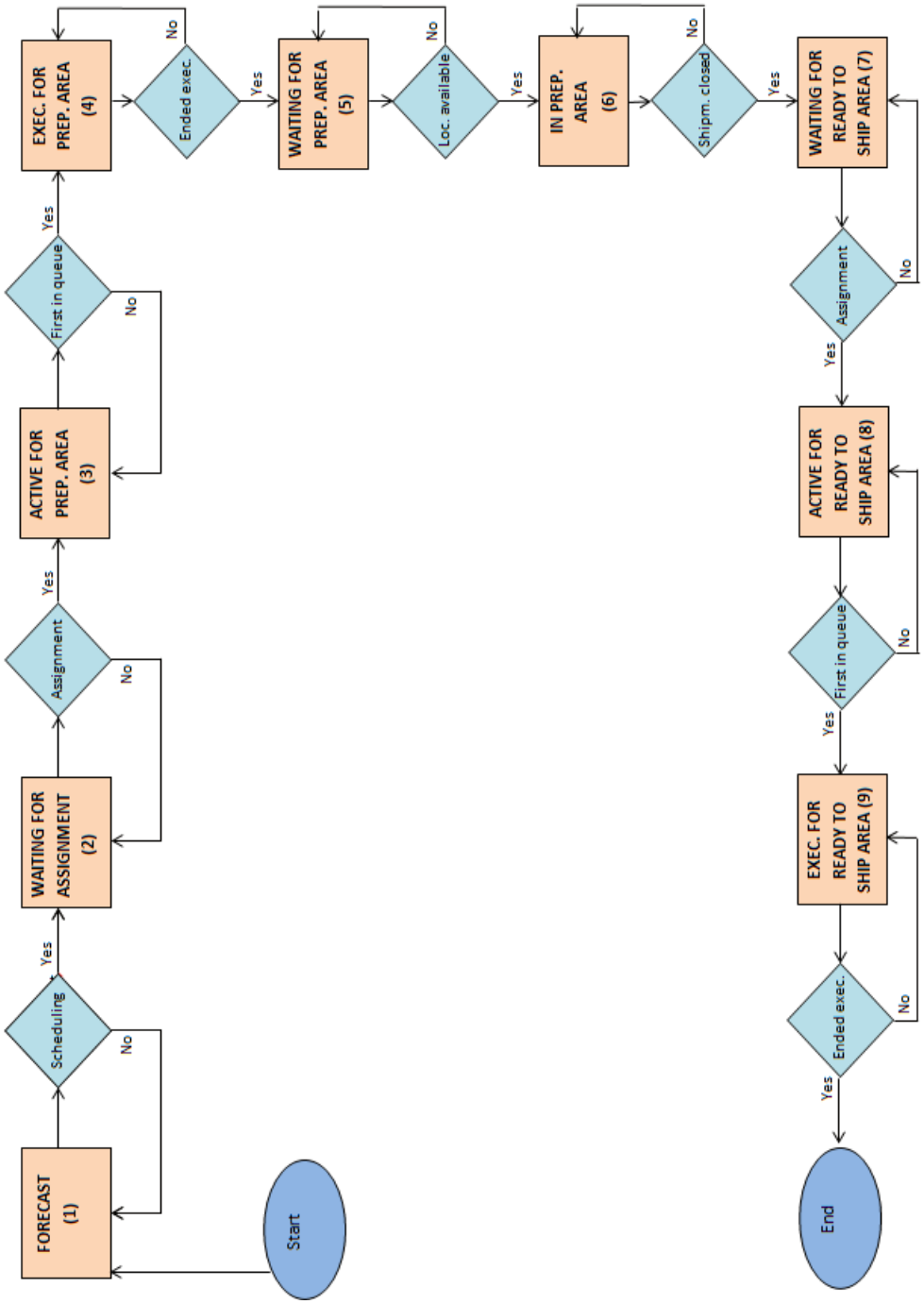

Figure 5.2: Life cycle of a mission

5.3.1 Transition from the forecast state to the waiting for assignment state

The state corresponding to the beginning of the life cycle of a mission is named forecast state, (rectangle with number 1 in figure 5.2). As we explained before, the missions in this state are associated to the requests in the portfolio of the warehouse, waiting for being scheduled for their subsequent execution.

The event of the scheduling of any forecast mission marks the transition from the forecast state to the state, that here we call *waiting for assignment*, (corresponding to the previously introduced waiting state, rectangle with number 2 in figure 5.2), in which the mission starts waiting for an appropriate available forklift to perform it.

Each forecast mission has a priority, associated to the corresponding non-scheduled request, (we recall that a priority is given to all non-scheduled requests). The developed algorithm considers the forecast missions in descending order of priority. When a forecast mission reaches the current highest priority, the algorithm checks the satisfiability of some conditions, before making the transition from the forecast state to the waiting for assignment state. These conditions are related to the level of occupation of the locations of the preparation area.

As we explained before, besides the quality checks of goods, in the preparation area orders are organized and grouped together to form the shipments, ready to be loaded on trucks at the ready to ship area. The pallets belonging to the same shipment are stored within contiguous locations, and each location is reserved for storing pallets belonging to exactly one shipment.

Before executing the transition from the forecast state to the waiting for assignment state for the considered mission, the algorithm verifies if there is at least one *free* location in the preparation area, to be reserved for the shipment to which the considered mission belongs. We define a free location as a location that is completely empty, and that is useless for storing pallets for the currently present shipments in the preparation area.

The developed algorithm checks the uselessness of each empty location by verifying, for all shipments currently present in the preparation area, if and which other contiguous locations they need to be completed. If an empty location is not needed for completing shipments, it can be defined free.

To better understand the concept of free location, figure 5.3 shows an example of empty and useless locations within a preparation area, composed by 10 locations. Each location in the example can contain 6 pallets. Three shipments are already present in the preparation area; these are the shipments 1, 2 and 3, that need in total, respectively, 20, 16 and 21 pallets. The first 4 locations are reserved for shipment 1, the fifth is empty and not assigned, (n.a.), the sixth and the seventh are reserved for shipment 2, and the last 3 locations are dedicated to shipment 3. Let us assume that the shipments 1 and 3 have to be completed by the arrival of other full pallets, while the shipment 2 has already been completed, and the release of its reserved locations has been started from the fifth location. The shipment 1 has to be completed, but it has a sufficient available total capacity for storing all its pallets. On the contrary, the shipment 3 does not have it. When the seventh location, (the green and orange one), will be released by the shipment 2, the shipment 3 will need to occupy it to satisfy its total pallet request. The fifth location, (and also the sixth one), will not, hence, be used by the already present shipments in the preparation area; they are free and they can be reserved for the shipment to which the considered forecast mission belongs.

The presence of free locations for the scheduling of forecast missions avoids waiting times for the forklifts, that will carry the related pallets to the preparation area, (waiting times extend the duration of the queues of missions of the forklifts, hence compromising the aim of the algorithm to minimize the duration of the queues). To have the certainty of the correctness of the control of the free locations, the algorithm checks if the following precondition is satisfied, i.e. if all shipments related to each currently scheduled, (active and waiting), mission are present in the preparation area. If this condition is satisfied, the correctness of the control is immediately assured.

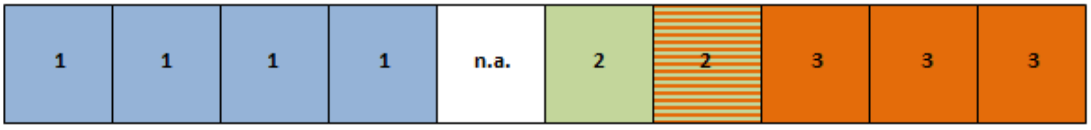

Figure 5.3: Example of useless location in the preparation area

When a forecast mission is considered for scheduling, the developed algorithm controls each empty location in the preparation area, with the aim of identifying a sequence of contiguous free locations sufficient to contain all the pallets related to the shipment

of the considered forecast mission. The found sequence of contiguous free locations is immediately reserved to the interested shipment. If no sequence of free locations is found, the considered forecast mission remains in the forecast state, keeping on waiting for free locations. If the shipment to which the considered mission belongs is already present in the preparation area, the algorithm checks if there is one available position within the reserved locations; in case, the scheduling of the mission happens, otherwise the mission remains in the forecast state, waiting for new free locations to be occupied by its shipment.

The adoption of this criterion permits to schedule a mission when there is the certainty that its execution will not block the work flow of the forklift; in fact, the forklift will not have to wait for an available location in the preparation area, since there are sufficient locations for receiving the requested pallet of good. The avoidance of waiting times is useful to minimize the completion times of the queues of missions of the forklifts.

5.3.2 Transition from the waiting for assignment state to the active for preparation area state

When a mission stands in the state of waiting for assignment, it waits for a compatible forklift, (operating in the area and for the role required by the mission), that can carry the requested pallet of goods from the storage areas to the preparation area. The assignment event marks the transition from the state of waiting for assignment to the state here called *active for preparation area*, (corresponding to the previously introduced active state, rectangle with number 3 in figure 5.2), in which the mission is assigned to a forklift, that will perform it, through the carriage of the requested pallet towards the preparation area.

The assignment criterion has the aim of maximizing the degree of utilization of the forklifts, maintaining, at the same time, a balanced workload among the forklifts, compatibly with the constraints on the maximum number of assignable missions. The developed algorithm considers each mission waiting for assignment, following the time of scheduling of the missions, starting from the currently first scheduled one. Hence, the implemented assignment criterion assigns the currently earliest scheduled mission to the compatible forklift, having the currently lower number of active missions in its queue. If such a forklift is found, the algorithm makes the assignment; otherwise, the

mission remains in the state of waiting for assignment, waiting for the right conditions for the assignment.

The adoption of this criterion for assigning the missions to forklifts has the aim of maximizing the degree of use of the forklifts, avoiding situations in which a forklift is more used than the other ones.

5.3.3 Transitions from the active for preparation area state to the in preparation area state

When a mission has been assigned to a compatible forklift, it enters the last position of its priority queue of active missions. The forklift executes the missions in the queue from the first to the last one. When a mission becomes the first in the queue, the forklift starts its execution, and the mission changes its state from active for preparation area to the state *execution for preparation area*, (rectangle with number 4 in figure 5.2). This state represents the execution of the considered mission.

When the forklift arrives at the preparation area, it has to store the pallet of the considered mission. The state of the mission representing the arrival at the preparation area is *waiting for preparation area*, (rectangle with number 5 in figure 5.2). In this state, the forklift has to wait for an available position in the preparation area, where to store the pallet, according to the shipment to which the considered mission belongs.

The developed algorithm controls each empty location of the preparation area, with the aim of identifying a sequence of contiguous free locations sufficient to contain all the pallets related to the shipment of the considered mission. The found sequence of free contiguous locations is immediately reserved to the interested shipment. If no sequence of free locations is found, the forklift keeps on waiting for a free location where to store the pallet. If the shipment to which the considered mission belongs is already present in the preparation area, the algorithm checks if there is one available position within the reserved locations. In case, the forklift stores the pallet in the available position; this event is modeled through a change of the state of the corresponding mission, from waiting for preparation area to *in preparation area*, (rectangle with number 6 in figure 5.2). At the same time, the forklift can start the execution of the next mission in its priority queue. If there is not an available position within the reserved locations, the forklift keeps on waiting for a free location where to store the pallet.

5.3.4 Closure of a shipment and removal of pallets

Each time a pallet is stored in the preparation area, the developed algorithm verifies if the related shipment has been completed, i.e. if all the requested pallets of the shipment are present in the preparation area. If the shipment has been completed, its reserved locations have to be released, by carrying one pallet at a time from the preparation area towards the ready to ship area. The policy of management of each location of the preparation area is LIFO, i.e. the first pallet to be brought away is the last stored.

As soon as one location is completely released, the algorithm verifies if that location is useful for other shipments, already present in the preparation area, that need other free locations to store their pallets. To favor a useful release of locations of the completed shipment, the algorithm orders the release of the reserved locations, starting from the location juxtaposing the one reserved to the shipment with the earlier scheduling time. Hence, the removal of the pallets from the interested locations starts from the first location in the order, following a LIFO policy to remove the pallets of each location. The possible immediate reservation of empty locations is made to favor both the work flow of the forklifts, and the early schedule of the forecast missions, permitting the reservation of the locations to the interested shipment.

The physical removal of the pallets of a completed shipment is made by the forklifts, that, as for the carriage of pallets from the storage areas to the preparation area, transport the pallets towards the ready to ship area of the warehouse. The release operations follow the established order of the locations, removing the pallets of each location with a LIFO policy; hence, the removal operations of the completed shipment starts from the last stored pallet of the first location in the order.

The *waiting for ready to ship area*, (rectangle with number 7 in figure 5.2), is the state, representing the condition in which a pallet waits for a compatible forklift for the carriage to the ready to ship area. When such a forklift is available, according to the constraints on the maximum number of assignable missions, the considered mission is assigned to the forklift, entering the last position of the priority queue of the forklift. The assignment event corresponds to a transition of state for the mission, from waiting for ready to ship area to *active for ready to ship area*, (rectangle with number 8 in figure 5.2).

When each mission of the completed shipment is assigned, the remaining pallets of the shipment are considered for assignment following the LIFO policy for each location, following the established order of release for the locations. When the assigned mission becomes the first in the priority queue of the active missions of the forklift, the physical removal of the corresponding pallet from the preparation area and the begin of the carriage towards the ready to ship area take place; this event is modeled with a transition of state of the interested mission, from active for ready to ship area to *execution for ready to ship area*, (rectangle with number 9 in figure 5.2). When the forklift arrives at the ready to ship area, it unloads the carried pallet and the life cycle of the related mission ends; at this point, the forklift can start the execution of the following active mission in its queue of priority.

5.3.5 A pseudocode

Algorithm 7 is a basic pseudocode of the designed algorithm, showing the main steps of the method. The algorithm takes as input, in the order, the current time of launch of the algorithm t_cur, the total number of missions n, (including the currently active and waiting missions and the forecast missions associated to the requests in the portfolio), the list of the missions mis, the list of the available resources/forklifts of the warehouse res, the status of the preparation area p_area, the list of the shipments $shipm$, the current maximum priority of the forecast missions max_pr and the current minimum schedule time for the waiting missions min_t_sched.

The designed algorithm is composed by two main nested loops. The external loop, (lines 3-34), models the passing of time. The internal loop, (lines 5-32), controls the state of each mission in the current considered moment of time. Hence, the algorithm controls, at each moment, the state of all the missions.

The granularity of time considered by the algorithm is at the level of seconds, i.e. the progress is made of one second at a time. At line 2 we initialize the time $time$ to the time of launch of the algorithm, and we set the number of ended missions cnt_mis to 0. The external loop starts at line 3. Within this loop, all the missions are analyzed, one second at a time. We initialize the counter of the missions m to 0, at line 4. The internal loop starts at line 5. Within this loop, each mission, $mis[m]$, is analyzed. Lines 6-9 model the transition of a mission in the forecast state to the waiting for assignment state. If the considered forecast mission has a priority equal to

the current maximum priority of the forecast missions, (line 6), the algorithm checks the satisfiability of the conditions shown in paragraph 5.3.1, for the transition of state, (method $Verify\_Schedule$, at line 7). If the conditions are satisfied, the transition to the state waiting for assignment is made, keeping track of the time of scheduling of the mission, (field t_sched, at line 8).

Since the transition of state of $mis[m]$, we possibly update the value of the current maximum priority of the forecast missions, at line 9. Lines 10-13 model the transition of a mission in the waiting for assignment state to the active for preparation area state. If the considered mission has a scheduling time equal to the current minimum schedule time of the waiting missions, (line 10), the algorithm controls if there is a compatible available forklift to which assign the mission, according to what explained in paragraph 5.3.2, (method $Verify\_Assignment$ at line 11). If the assignment is done, the transition to the state active for preparation area is made, (line 12) and the value of the current minimum schedule time of the waiting missions is updated, (line 13).

Lines 14-15 model the transition of a mission in the active for preparation area state to the execution for preparation area state. If the considered active mission is the first active mission in the queue of priority of its forklift, (line 14), the forklift starts the execution of the mission, changing its state to execution for preparation area, and keeping track of the starting execution time, (field t_beg_pa at line 15).

Lines 16-17 model the transition of a mission in the execution for preparation area state to the waiting for preparation area state, corresponding to the arrival of the related pallet to the preparation area. The algorithm verifies if the execution of the mission is ended, by controlling if the passed time from its beginning is equal to the mission execution time, (line 16). If the execution is ended, the algorithm changes the state of the mission to the state waiting for preparation area, (line 17). The waiting for available locations, the storage and the closure of a complete shipment in the preparation area are modeled by the algorithm at lines 18-21.

The algorithm checks the state of occupation of the preparation area for every mission in the waiting for preparation area state, as explained in paragraph 5.3.3, (line 19). If an available position is present, the algorithm executes the storage of the pallet, and immediately controls if the related shipment has been completed, (line 20). If this is the case, the release of the interested locations starts; the state of the mission associated to the last stored pallet of the first location to be released is changed to the

state waiting for ready to ship, (line 21, see paragraph 5.3.4 for the details about this transition).

Lines 22-25 model the transition of a mission in the state waiting for ready to ship area to the state active for ready to ship area. If the assignment to a forklift is done, the state of the mission is changed, (line 24), and the state of the mission associated to the new last stored pallet of the location is changed, to the state waiting for ready to ship area; this means that the pallet is considered for assignment, (line 25). Lines 26-28 model the execution of the active mission for the carriage of the related pallet to the ready to ship area. If the considered mission is the first active mission in the queue of its forklift, it starts the execution of the mission, changing its state to execution for ready to ship area, and keeping track of the starting execution time (line 27).

Each time a pallet is removed from the preparation area, the algorithm checks if the related location is empty; in case, the method verifies if the location can be reserved for other shipments, (method *Extend_Shipment* at line 28).

Lines 29-30 model the end of the life cycle of the mission, coinciding with the arrival at the ready to ship area. In this case, the number of ended missions *cnt_mis* in incremented by 1, (line 30).

At line 31 we increment by 1 the counter of the missions m, to switch to the next mission. The stop condition of the internal loop is given by the analysis of all the missions, (line 32), and it coincides with the end of the analysis of all the missions in a particular moment. At line 33 we increment by 1 the time.

The algorithm stops its execution when all the missions end.

5.4 Output data of the algorithm

In this section, we show a detail of the output of the developed algorithm for an instance of the presented problem. The data of the treated instance are the following:

- 156 missions, (37 active for preparation area, 13 waiting for preparation area and 106 forecast);
- one preparation area, composed by 10 locations, each one with capacity equal to 6;

Algorithm 7 Algorithm for Predicting Duration Times of Queues

```
1: procedure PRED_TIMES_Q(t_cur, n, mis, res, p_area, shipm, max_pr, min_t_sched)
2:     time ← t_cur,   cnt_mis ← 0
3:     repeat
4:         m ← 0
5:         repeat
6:             if mis[m].st = forecast and mis[m].pr = max_pr then
7:                 if Verify_Schedule(mis[m], p_area, time) = true then
8:                     mis[m].st ← wait_ass,   mis[m].t_sched ← time
9:                     Update_Max_Priority(max_pr)
10:            if mis[m].st = wait_ass and mis[m].t_sched = min_t_sched then
11:                if Verify_Assignment(mis[m], res) = true then
12:                    mis[m].st ← active_pa
13:                    Update_Min_T_Sched(min_t_sched)
14:            if mis[m].st = active_pa and mis[m].first_in_q = true then
15:                mis[m].st ← ex_pa,   mis[m].t_beg_pa ← time
16:            if mis[m].st = ex_pa and time−mis[m].t_beg_pa = mis[m].t_ex_pa then
17:                mis[m].st ← wait_pa
18:            if mis[m].st = wait_pa then
19:                if Verify_Prep_Area(mis[m], p_area, shipm) = true then
20:                    if Close_Shipment(mis[m].ship) = true then
21:                        mis[shipm[mis[m].ship].first_mis].st ← wait_rts
22:            if mis[m].st = wait_rts then
23:                if Verify_Assignment(mis[m], res) = true then
24:                    mis[m].st ← active_rts
25:                    mis[shipm[mis[m].ship].next_mis].st ← wait_rts
26:            if mis[m].st = active_rts and mis[m].first_in_q = true then
27:                mis[m].st ← ex_rts,   mis[m].t_beg_rts ← time
28:                Extend_Shipment(p_area, shipm)
29:            if mis[m].st = ex_rts and time − mis[m].t_beg_rts = mis[m].t_ex_rts
    then
30:                cnt_mis ← cnt_mis + 1
31:            m ← m + 1
32:        until (m < n)
33:        time ← time + 1
34:    until (cnt_mis < n)
```

- 7 available forklifts, each one managing a priority queue of active missions with a maximum predefined length equal to 10.

Figure 5.4 shows the log of the operations made by the forklift with the acronym WHD-35. The log identifies the details of the missions executed by the forklift, i.e. in the order, the mission id, the acronym of the shipment to which the mission belongs, the area and the role required by the mission, (the role WHDOM: FULL PALLET corresponds to the carriage of full pallets from the storage areas to the preparation area), the time of scheduling of the mission, the time of assignment of the mission to the forklift, the time in which the execution of the mission has been started and the time of end of the execution of the mission.

The time of launch of the algorithm is 13:01:46. As we can see, the first ten missions in the log have been scheduled before the time of launch; at the beginning of the algorithm, they are waiting missions.

When the algorithm starts, these missions are immediately assigned to the forklift, and their execution respects the order of their scheduling time. We can note a very little delay of less than one minute between the end of the mission 33 and the beginning of the execution of the mission 56. This is due to the lack of sufficient positions in the preparation area for the shipment SH002, to which mission 33 belongs. At time 13:34:46, a contiguous location of the shipment SH002 is still occupied by the pallets belonging to another shipment. This location becomes completely empty at time 13:35:22; hence, at this time, the location can be used by shipment SH002, and the forklift can store the carried pallet of the mission 33; then, the forklift can immediately start the execution of the next mission, the mission 56.

The missions starting from mission 177 until the last mission in the log are forecast missions. We can see that the time of scheduling of these missions is equal to the time of their assignment, meaning that their creation happens when the system can support their execution, to avoid a premature scheduling potentially introducing an overload of scheduled missions.

FORKLIFT		MISSION ID	SHIPMENT	AREA	ROLE	SCHEDULING TIME	ASSIGNMENT TIME	START TIME	END TIME
WHD-35									
		26	SH002	D.L2	WHDOM: FULL PALLET	12:10:00	13:01:46	13:01:46	13:06:49
		31	SH002	D.L4	WHDOM: FULL PALLET	12:10:00	13:01:46	13:06:49	13:16:08
		32	SH002	D.L4	WHDOM: FULL PALLET	12:10:00	13:01:46	13:16:08	13:25:27
		33	SH002	D.L4	WHDOM: FULL PALLET	12:10:00	13:01:46	13:25:27	13:34:46
		56	SH004	D.L4	WHDOM: FULL PALLET	12:45:00	13:01:46	13:35:22	13:44:41
		57	SH004	D.L4	WHDOM: FULL PALLET	12:45:00	13:01:46	13:44:41	13:54:00
		58	SH004	D.L4	WHDOM: FULL PALLET	12:45:00	13:01:46	13:54:00	14:03:19
		73	SH004	D.L3	WHDOM: FULL PALLET	12:45:00	13:01:46	14:03:19	14:08:08
		76	SH004	D.L3	WHDOM: FULL PALLET	12:45:00	13:01:46	14:08:08	14:12:57
		79	SH004	D.L3	WHDOM: FULL PALLET	12:45:00	13:01:46	14:12:57	14:17:46
		177	SH005	D.L1	WHDOM: FULL PALLET	13:54:11	13:54:11	14:17:46	14:23:32
		183	SH005	D.L3	WHDOM: FULL PALLET	14:06:31	14:06:31	14:23:32	14:28:21
		186	SH005	D.L3	WHDOM: FULL PALLET	14:06:31	14:06:31	14:28:21	14:33:10
		154	SH007	D.L3	WHDOM: FULL PALLET	14:12:48	14:12:48	14:33:10	14:37:59
		139	SH006	D.L1	WHDOM: FULL PALLET	14:25:20	14:25:20	14:37:59	14:43:45
		143	SH006	D.L1	WHDOM: FULL PALLET	14:25:20	14:25:20	14:43:45	14:49:31
		115	SH008	D.L3	WHDOM: FULL PALLET	14:44:12	14:44:12	14:49:31	14:54:20
		118	SH008	D.L3	WHDOM: FULL PALLET	14:44:12	14:44:12	14:54:20	14:59:09
		121	SH008	D.L4	WHDOM: FULL PALLET	15:21:55	15:21:55	15:21:55	15:31:14
		128	SH008	D.L4	WHDOM: FULL PALLET	15:28:13	15:28:13	15:31:14	15:40:33
		134	SH008	D.L4	WHDOM: FULL PALLET	15:34:30	15:34:30	15:40:33	15:49:52
		106	SH009	D.L4	WHDOM: FULL PALLET	15:34:30	15:34:30	15:49:52	15:59:11
		113	SH009	D.L4	WHDOM: FULL PALLET	15:34:30	15:34:30	15:59:11	16:08:30
		90	SH011	D.L3	WHDOM: FULL PALLET	15:34:30	15:34:30	16:08:30	16:13:19
		93	SH011	D.L3	WHDOM: FULL PALLET	15:34:30	15:34:30	16:13:19	16:18:08
		84	SH010	D.L1	WHDOM: FULL PALLET	15:34:30	15:34:30	16:18:08	16:23:54

Figure 5.4: Log of the missions executed by forklift WHD-35

6

Conclusions

In the present thesis we investigated the topic of model-based heuristics, applied to the solution of COPs, both from a theoretical and a practical point of view. The current scientific research dealing with the solution of hard COPs is concentrating the efforts on the design of these algorithms, with the aim of improving the state of the art of the solution of COPs, through the design of tailored integrations of exact and heuristic techniques, able to overcome the weaknesses of one another.

We made a survey of the scientific literature, related to the development of model-based heuristics for treating hard COPs. We provided a classification of hybrids of mathematical programming and heuristic techniques, in three main classes: mathematical programming techniques subordinated to heuristics, heuristics subordinated to mathematical programming methods and cooperation between heuristics and mathematical programming. For each identified class, we provided some examples from the scientific literature, dealing with the solution of several COPs.

We proposed a Lagrangean CG heuristic for the solution of the CVRP; the algorithm is able to produce both a valid lower bound and feasible solutions for the treated instances. We introduced the CVRP, showing one possible mathematical formulation for the problem, based on a SP formulation, (where each column corresponds to a route). We introduced some state space relaxation techniques for the problem, ((q, i)-path and ng-path relaxations), and we presented the Lagrangean CG heuristic. The method relies on a CG procedure, where the master problem is solved by a LP solver and the pricing step consists in identifying new negative reduced costs columns/routes through the calculation of (q, i)-path or ng-path relaxations using the reduced costs

of the arcs. At the end of the CG, we have a valid lower bound, and the columns of the obtained master problem are used to build a SC model, solved through subgradient optimization; the infeasibilities of the subgradient solution are fixed, and a feasible CVRP solution is built during each iteration of the subgradient.

We studied the parameter tuning problem, applied to the tuning of a Lagrangean heuristic for solving the CVRP and the VRPTW. We introduced the tuning problem, and some methods from the literature designed to treat this issue; in particular, we introduced racing procedures, and the irace package implementing the iterated racing procedure. We introduced the VRPTW, and one possible mathematical formulation, based on a SP model. We showed some state space relaxation techniques for the problem, ((t, i)-path and ng-path relaxations), and we presented the Lagrangean heuristic. We treated the tuning of the Lagrangean heuristic with the irace package, with the aim of improving the quality of the calculated valid lower bound for the solved instances. The computational results reveal the capability of the irace package of improving the quality of the calculated valid lower bounds, hence confirming our intuition that, with a tailored tuning of the Lagrangean heuristic, we could be able to improve the quality of the valid lower bounds. We further investigated the parameter tuning problem, considering the tuning of an ILS method, applied for solving instances of the QAP. The obtained computational results permit us to improve the quality of the heuristic solutions, calculated by the ILS on the studied QAP instances.

We presented a real-world problem, emerging in the context of the daily functioning of a warehouse commercializing tiles, located in Thailand. The problem asked for the prediction of the duration of the queues of process of the resources, operating in the warehouse. We showed the physical and logical organization of the warehouse, and the resources operating within it. We detailed the problem and the heuristic method we designed to treat it. The developed heuristic does not deal with all the aspects of the daily functioning of the warehouse, (e.g. the management of the picking bay); hence we can imagine future extensions of the algorithm, able to manage all currently non-treated aspects of the operating of the warehouse.

References

[1] CUDA, parallel computing platform. http://www.nvidia.com/object/cuda_home_new.html. 44

[2] VRP problem instances. http://branchandcut.org/VRP/data, . Accessed: 2011-09-11. 65

[3] CVRPLIB capacitated vehicle routing problem library. http://vrp.atd-lab.inf.puc-rio.br/index.php/en/, . 44, 64

[4] COLOR02/03/04: Graph coloring and its generalizations. http://mat.gsia.cmu.edu/COLOR02. 9

[5] OpenMP API specification for parallel programming. http://openmp.org/wp/. 44

[6] QAP, taillard, symmetrical and structures instances. website. http://mistic.heig-vd.ch/taillard/problemes.dir/qap.dir/qap.html. 72

[7] CVRP, symmetric capacitated vehicle routing. http://astarte.csr.unibo.it/data/#CVRP. x, 45, 46, 121

[8] SINTEF: Problems and benchmarks, VRPTW. website. https://www.sintef.no/projectweb/top/vrptw/homberger-benchmark, . 65, 67

[9] SINTEF: Problems and benchmarks, VRPTW. website. https://www.sintef.no/projectweb/top/vrptw/solomon-benchmark/, . 65, 67

[10] Prodhon, c. http://prodhonc.free.fr/. 9

[11] Rodrigo Acuna-Agost, Philippe Michelon, Dominique Feillet, and Serigne Gueye. A MIP-based local search method for the railway rescheduling problem. *Networks*, 57(1):69–86, 2011. 11

[12] Ravindra K Ahuja, James B Orlin, and Dushyant Sharma. Very large-scale neighborhood search. *International Transactions in Operational Research*, 7(4-5):301–317, 2000. 7

[13] Ravindra K Ahuja, James B Orlin, and Dushyant Sharma. A composite very large-scale neighborhood structure for the capacitated minimum spanning tree problem. *Operations Research Letters*, 31(3):185–194, 2003. 9

[14] M Allen, Giridhar Kumaran, and Tong Liu. A combined algorithm for graph-coloring in register allocation. In *Proceedings of the Computational Symposium on Graph Coloring and its Generalizations*, pages 100–111, 2002. 9

[15] Enrico Angelelli, Renata Mansini, and M Grazia Speranza. Kernel search: A general heuristic for the multi-dimensional knapsack problem. *Computers & Operations Research*, 37(11):2017–2026, 2010. 18

[16] Enrico Angelelli, Renata Mansini, and M Grazia Speranza. Kernel search: a new heuristic framework for portfolio selection. *Computational Optimization and Applications*, 51(1): 345–361, 2012. 18

[17] Davide Anghinolfi, Luca Maria Gambardella, Roberto Montemanni, Cristiano Nattero, Massimo Paolucci, and NE Toklu. A matheuristic algorithm for a large-scale energy management problem. In *Large-Scale Scientific Computing*, pages 173–181. Springer, 2012. 26

[18] C Archetti, G Guastaroba, and MG Speranza. An ILP-refined tabu search for the directed profitable rural postman problem. *Discrete Applied Mathematics*, 163:3–16, 2014. 26

[19] Claudia Archetti, Maria Grazia Speranza, and Alain Hertz. A tabu search algorithm for the split delivery vehicle routing problem. *Transportation Science*, 40(1):64–73, 2006. 24

[20] Claudia Archetti, Luca Bertazzi, Gilbert Laporte, and Maria Grazia Speranza. A branch-and-cut algorithm for a vendor-managed inventory-routing problem. *Transportation Science*, 41(3):382–391, 2007. 22, 23

[21] Claudia Archetti, M Grazia Speranza, and Martin WP Savelsbergh. An optimization-based heuristic for the split delivery vehicle routing problem. *Transportation Science*, 42 (1):22–31, 2008. 24, 26

[22] Claudia Archetti, Luca Bertazzi, Alain Hertz, and M Grazia Speranza. A hybrid heuristic for an inventory routing problem. *INFORMS Journal on Computing*, 24(1):101–116, 2012. 22, 26

[23] Claudia Archetti, N Bianchessi, MG Speranza, and A Hertz. The split delivery capacitated team orienteering problem. *Networks*, 63(1):16–33, 2014. 24

[24] Ph Augerat, JM Belenguer, E Benavent, A Corberán, D Naddef, and G Rinaldi. *Computational results with a branch and cut code for the capacitated vehicle routing problem.* IMAG, 1995. 65

[25] Philippe Augerat, José M Belenguer, Enrique Benavent, Angel Corbéran, and Denis Naddef. Separating capacity constraints in the CVRP using tabu search. *European Journal of Operational Research*, 106(2):546–557, 1998. 19, 21

[26] Prasanna Balaprakash, Mauro Birattari, and Thomas Stützle. Improvement strategies for the f-race algorithm: Sampling design and iterative refinement. In *Hybrid Metaheuristics*, pages 108–122. Springer, 2007. 49

[27] Egon Balas. New classes of efficiently solvable generalized traveling salesman problems. *Annals of Operations Research*, 86:529–558, 1999. 9

[28] Egon Balas and Neil Simonetti. Linear time dynamic-programming algorithms for new classes of restricted TSPs: A computational study. *INFORMS Journal on Computing*, 13(1):56–75, 2001. 9

[29] Roberto Baldacci, Eleni Hadjiconstantinou, and Aristide Mingozzi. An exact algorithm for the capacitated vehicle routing problem based on a two-commodity network flow formulation. *Operations Research*, 52(5):723–738, 2004. 28

[30] Roberto Baldacci, Nicos Christofides, and Aristide Mingozzi. An exact algorithm for the vehicle routing problem based on the set partitioning formulation with additional cuts. *Mathematical Programming*, 115(2):351–385, 2008. 28

[31] Roberto Baldacci, Aristide Mingozzi, and Roberto Roberti. New route relaxation and pricing strategies for the vehicle routing problem. *Operations research*, 59(5):1269–1283, 2011. 31, 53

[32] Francisco Barahona and Ranga Anbil. The volume algorithm: producing primal solutions with a subgradient method. *Mathematical Programming*, 87(3):385–399, 2000. 25

[33] Cynthia Barnhart, Ellis L Johnson, George L Nemhauser, Martin WP Savelsbergh, and Pamela H Vance. Branch-and-price: Column generation for solving huge integer programs. *Operations research*, 46(3):316–329, 1998. 20

[34] Nicolas Barnier and Pascal Brisset. Graph coloring for air traffic flow management. *Annals of operations research*, 130(1-4):163–178, 2004. 9

[35] Roberto Battiti and Giampietro Tecchiolli. The reactive tabu search. *ORSA journal on computing*, 6(2):126–140, 1994. 69

[36] J.E. Beasley. Lagrangean heuristics for location problems. *European Journal of Operational Research*, 65(3):383 – 399, 1993. 18

[37] John E Beasley. OR-library: distributing test problems by electronic mail. *Journal of the operational research society*, pages 1069–1072, 1990. 25

[38] John E Beasley. Obtaining test problems via internet. *Journal of Global Optimization*, 8 (4):429–433, 1996. 25

[39] Gaetan Belvaux and Laurence A Wolsey. bcprod: A specialized branch-and-cut system for lot-sizing problems. *Management Science*, 46(5):724–738, 2000. 20

[40] Jacques F Benders. Partitioning procedures for solving mixed-variables programming problems. *Numerische mathematik*, 4(1):238–252, 1962. 18

[41] Russell Bent and Pascal Van Hentenryck. A two-stage hybrid local search for the vehicle routing problem with time windows. *Transportation Science*, 38(4):515–530, 2004. 25

[42] Luca Bertazzi, Giuseppe Paletta, and M Grazia Speranza. Deterministic order-up-to level policies in an inventory routing problem. *Transportation Science*, 36(1):119–132, 2002. 22

[43] Mauro Birattari. The race package for r. racing methods for the selection of the best. 2003. 50

[44] Mauro Birattari, Thomas Stützle, Luis Paquete, Klaus Varrentrapp, et al. A racing algorithm for configuring metaheuristics. In *GECCO*, volume 2, pages 11–18, 2002. 48

[45] Mauro Birattari, Zhi Yuan, Prasanna Balaprakash, and Thomas Stützle. F-race and iterated f-race: An overview. In *Experimental methods for the analysis of optimization algorithms*, pages 311–336. Springer, 2010. 49

[46] Jacek Blazewicz, Ceyda Oguz, Aleksandra Swiercz, and Jan Weglarz. DNA sequencing by hybridization via genetic search. *Operations Research*, 54(6):1185–1192, 2006. 12

[47] Christian Blum, Mateu Yábar Vallès, and Maria J Blesa. An ant colony optimization algorithm for DNA sequencing by hybridization. *Computers & Operations Research*, 35 (11):3620–3635, 2008. 1

[48] Christian Blum, Jakob Puchinger, Günther R Raidl, and Andrea Roli. Hybrid metaheuristics in combinatorial optimization: A survey. *Applied Soft Computing*, 11(6):4135–4151, 2011. 6

[49] P Borisovsky, Alexandre Dolgui, and A Eremeev. Genetic algorithms for a supply management problem: MIP-recombination vs greedy decoder. *European Journal of Operational Research*, 195(3):770–779, 2009. 16, 17

[50] Marco Boschetti and Vittorio Maniezzo. Benders decomposition, lagrangean relaxation and metaheuristic design. *Journal of Heuristics*, 15(3):283–312, 2009. 18

[51] Marco Boschetti and Vittorio Maniezzo. A set covering based matheuristic for a real-world city logistics problem. *International Transactions in Operational Research*, 22(1): 169–195, 2015. 38, 40, 54

[52] Marco Boschetti, Vittorio Maniezzo, and Matteo Roffilli. Decomposition techniques as metaheuristic frameworks. In *Matheuristics*, pages 135–158. Springer, 2010. 18

[53] Marco A Boschetti and Vittorio Maniezzo. Combining exact methods and heuristics. *Wiley Encyclopedia of Operations Research and Management Science*, 2011. 6

[54] Marco A Boschetti, Vittorio Maniezzo, Matteo Roffilli, and Antonio Bolufé Röhler. Matheuristics: Optimization, simulation and control. In *Hybrid Metaheuristics*, pages 171–177. Springer, 2009. 5

[55] Marco A Boschetti, Vittorio Maniezzo, Matteo Roffilli, and Antonio José Bolufé Röhler. Matheuristics for traffic counter location. In *Proc. of the VII ALIO/EURO Workshop on Applied Combinatiorial Optimization, Porto, Portugal, May 4-6, 2011*, pages 77–80. 2011. 19

[56] Adrian Brüngger, Ambros Marzetta, Jens Clausen, and Michael Perregaard. Joining forces in solving large-scale quadratic assignment problems in parallel. In *Parallel Processing Symposium, 1997. Proceedings., 11th International*, pages 418–427. IEEE, 1997. 69

[57] Guillermo Cabrera G, Enrique Cabrera, Ricardo Soto, L Rubio, Broderick Crawford, and Fernando Paredes. A hybrid approach using an artificial bee algorithm with mixed integer programming applied to a large-scale capacitated facility location problem. *Mathematical Problems in Engineering*, 2012, 2012. 17

[58] Sébastien Cahon, Nordine Melab, and E-G Talbi. Paradiseo: A framework for the reusable design of parallel and distributed metaheuristics. *Journal of Heuristics*, 10(3):357–380, 2004. 72

[59] J Carlier and P Villon. A new heuristic for the traveling salesman problem. *RAIRO. Recherche opérationnelle*, 24(3):245–253, 1990. 9

[60] Marco Caserta and Stefan Voß. A corridor method-based algorithm for the premarshalling problem. In *Applications of Evolutionary Computing*, pages 788–797. Springer, 2009. 12

[61] Marco Caserta and Stefan Voß. A math-heuristic algorithm for the DNA sequencing problem. In *Learning and Intelligent Optimization*, pages 25–36. Springer, 2010. 1, 12

[62] Marco Caserta and Stefan Voß. A math-heuristic Dantzig-Wolfe algorithm for the capacitated lot sizing problem. In *Learning and Intelligent Optimization*, pages 31–41. Springer, 2012. 20

[63] Marco Caserta and Stefan Voß. A hybrid algorithm for the DNA sequencing problem. *Discrete Applied Mathematics*, 163:87–99, 2014. 1, 12

[64] Marco Caserta, Adriana Ramirez, and Stefan Voß. A math-heuristic for the multi-level capacitated lot sizing problem with carryover. In *Applications of Evolutionary Computation*, pages 462–471. Springer, 2010. 12

[65] Marco Caserta, Stefan Voß, and Moshe Sniedovich. Applying the corridor method to a blocks relocation problem. *OR spectrum*, 33(4):915–929, 2011. 11

[66] Alberto Ceselli and Giovanni Righini. A branch-and-price algorithm for the capacitated p-median problem. *Networks*, 45(3):125–142, 2005. 15

[67] Marco Chiarandini, Irina Dumitrescu, and Thomas Stützle. Local search for the colouring graph problem. A computational study. *TU Darmstadt, FB Intellektik, FG Informatik Hochschulstr. 10, 64289 Darmstadt, German*, 2003. 8

[68] Mervat Chouman and Teodor Crainic. *A MIP-tabu search hybrid framework for multi-commodity capacitated fixed-charge network design.* CIRRELT, 2010. 22, 26

[69] N Christofides, A Mingozzi, P Toth, and C Sandi. Combinatorial optimization, 1979. 65

[70] Nicos Christofides and Samuel Eilon. An algorithm for the vehicle-dispatching problem. *Or*, pages 309–318, 1969. 65

[71] Nicos Christofides, Aristide Mingozzi, and Paolo Toth. Exact algorithms for the vehicle routing problem, based on spanning tree and shortest path relaxations. *Mathematical programming*, 20(1):255–282, 1981. 31, 52

[72] Paul C Chu and John E Beasley. A genetic algorithm for the multidimensional knapsack problem. *Journal of heuristics*, 4(1):63–86, 1998. 25

[73] Paul CH Chu. *A genetic algorithm approach for combinatorial optimisation problems.* PhD thesis, Imperial College London, 1997. 25

[74] Leandro C Coelho, Jean-François Cordeau, and Gilbert Laporte. Consistency in multi-vehicle inventory-routing. *Transportation Research Part C: Emerging Technologies*, 24: 270–287, 2012. 23, 26

[75] Richard K Congram, Chris N Potts, and Steef L van de Velde. An iterated dynasearch algorithm for the single-machine total weighted tardiness scheduling problem. *INFORMS Journal on Computing*, 14(1):52–67, 2002. 9, 10

[76] David T Connolly. An improved annealing scheme for the qap. *European Journal of Operational Research*, 46(1):93–100, 1990. 69

[77] Pedro J Copado-Méndez, Christian Blum, Gonzalo Guillén-Gosálbez, and Laureano Jiménez. Large neighbourhood search applied to the efficient solution of spatially explicit strategic supply chain management problems. *Computers & Chemical Engineering*, 49:114–126, 2013. 8

[78] Carlos Cotta and José M Troya. Embedding branch and bound within evolutionary algorithms. *Applied Intelligence*, 18(2):137–153, 2003. 15

[79] Teodor Gabriel Crainic, Ye Li, and Michel Toulouse. A first multilevel cooperative algorithm for capacitated multicommodity network design. *Computers & Operations Research*, 33(9):2602–2622, 2006. 11

[80] HAJ Crauwels, Chris N Potts, and Luk N Van Wassenhove. Local search heuristics for the single machine total weighted tardiness scheduling problem. *INFORMS Journal on computing*, 10(3):341–350, 1998. 10

[81] André Renato Villela da Silva and Luiz Satoru Ochi. Hybrid heuristics for dynamic resource-constrained project scheduling problem. In *Hybrid Metaheuristics*, pages 73–87. Springer, 2010. 22

[82] Emilie Danna, Edward Rothberg, and Claude Le Pape. Exploring relaxation induced neighborhoods to improve MIP solutions. *Mathematical Programming*, 102(1):71–90, 2005. 13

[83] George B Dantzig and John H Ramser. The truck dispatching problem. *Management science*, 6(1):80–91, 1959. 27

[84] George B Dantzig and Philip Wolfe. Decomposition principle for linear programs. *Operations research*, 8(1):101–111, 1960. 18

[85] Roberto De Franceschi, Matteo Fischetti, and Paolo Toth. A new ILP-based refinement heuristic for vehicle routing problems. *Mathematical Programming*, 105(2-3):471–499, 2006. 8

[86] Marco A Montes de Oca, Doğan Aydın, and Thomas Stützle. An incremental particle swarm for large-scale continuous optimization problems: an example of tuning-in-the-loop (re) design of optimization algorithms. *Soft Computing*, 15(11):2233–2255, 2011. 50

[87] Zeger Degraeve and Raf Jans. A new dantzig-wolfe reformulation and branch-and-price algorithm for the capacitated lot-sizing problem with setup times. *Operations Research*, 55(5):909–920, 2007. 20

[88] Federico Della Croce and Fabio Salassa. A variable neighborhood search based matheuristic for nurse rostering problems. *Annals of Operations Research*, pages 1–15, 2010. 26

[89] Federico Della Croce, Marco Ghirardi, and Roberto Tadei. Recovering beam search: enhancing the beam search approach for combinatorial optimization problems. *Journal of Heuristics*, 10(1):89–104, 2004. 25

[90] Federico Della Croce, Andrea Grosso, and Fabio Salassa. A matheuristic approach for the two-machine total completion time flow shop problem. *Annals of Operations Research*, pages 1–12, 2011. 25

[91] Guy Desaulniers, François Lessard, and Ahmed Hadjar. Tabu search, partial elementarity, and generalized k-path inequalities for the vehicle routing problem with time windows. *Transportation Science*, 42(3):387–404, 2008. 50

[92] Martin Desrochers, Jacques Desrosiers, and Marius Solomon. A new optimization algorithm for the vehicle routing problem with time windows. *Operations research*, 40(2): 342–354, 1992. 50

[93] Jacques Desrosiers, Yvan Dumas, Marius M Solomon, and François Soumis. Time constrained routing and scheduling. *Handbooks in operations research and management science*, 8:35–139, 1995. 50

[94] Karl F Doerner, Guenther Fuellerer, Richard F Hartl, Manfred Gronalt, and Manuel Iori. Metaheuristics for the vehicle routing problem with loading constraints. *Networks*, 49(4): 294–307, 2007. 20

[95] Alexandre Dolgui, Anton Eremeev, and Olga Guschinskaya. MIP-based GRASP and genetic algorithm for balancing transfer lines. In *Matheuristics*, pages 189–208. Springer, 2010. 16, 17

[96] Marco Dorigo, Vittorio Maniezzo, and Alberto Colorni. Ant system: optimization by a colony of cooperating agents. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, 26(1):29–41, 1996. 16

[97] André Gustavo dos Santos and Geraldo Robson Mateus. Hybrid approach to solve a crew scheduling problem: an exact column generation algorithm improved by metaheuristics. In *Hybrid Intelligent Systems, 2007. HIS 2007. 7th International Conference on*, pages 107–112. IEEE, 2007. 20, 21

[98] Alexandre R Duarte, Celso C Ribeiro, and Sebastián Urrutia. A hybrid ILS heuristic to the referee assignment problem with an embedded MIP strategy. In *Hybrid Metaheuristics*, pages 82–95. Springer, 2007. 16, 17

[99] Alexandre R Duarte, Celso C Ribeiro, Sebastián Urrutia, and Edward H Haeusler. Referee assignment in sports leagues. In *Practice and Theory of Automated Timetabling VI*, pages 158–173. Springer, 2007. 16

[100] Jérémie Dubois-Lacoste, Manuel López-Ibáñez, and Thomas Stützle. A hybrid tp+ pls algorithm for bi-objective flow-shop scheduling problems. *Computers & Operations Research*, 38(8):1219–1236, 2011. 50

[101] Jérémie Dubois-Lacoste, Manuel López-Ibáñez, and Thomas Stützle. Improving the anytime behavior of two-phase local search. *Annals of mathematics and artificial intelligence*, 61(2):125–154, 2011. 50

[102] Irina Dumitrescu and Thomas Stützle. Combinations of local search and exact algorithms. In *Applications of Evolutionary Computing*, pages 211–223. Springer, 2003. 6

[103] Irina Dumitrescu and Thomas Stützle. Usage of exact algorithms to enhance stochastic local search algorithms. In Vittorio Maniezzo, Thomas Sttzle, and Stefan Vo, editors, *Matheuristics*, pages 103–134. Springer US, 2010. 6

[104] Jack Edmonds. Paths, trees, and flowers. *Canadian Journal of mathematics*, 17(3): 449–467, 1965. 15

[105] Alwalid N Elshafei. Hospital layout as a quadratic assignment problem. *Operational Research Quarterly*, pages 167–179, 1977. 69

[106] Anton V Eremeev. On complexity of optimal recombination for binary representations of solutions. *Evolutionary Computation*, 16(1):127–147, 2008. 15

[107] Thomas A Feo and Mauricio GC Resende. Greedy randomized adaptive search procedures. *Journal of global optimization*, 6(2):109–133, 1995. 17

[108] Susana Fernandes and Helena R Lourenço. A GRASP and branch-and-bound metaheuristic for the job-shop scheduling. In *Evolutionary Computation in Combinatorial Optimization*, pages 60–71. Springer, 2007. 17

[109] Susana Fernandes and Helena R. Lourenço. Optimised search heuristic combining valid inequalities and tabu search. In *Hybrid Metaheuristics*, pages 87–101. Springer Berlin Heidelberg, 2008. 24

[110] Matteo Fischetti and Andrea Lodi. Local branching. *Mathematical programming*, 98(1-3): 23–47, 2003. 10, 13, 14

[111] Matteo Fischetti and Paolo Toth. An additive bounding procedure for combinatorial optimization problems. *Operations Research*, 37(2):319–328, 1989. 28

[112] Matteo Fischetti, Carlo Polo, and Massimo Scantamburlo. A local branching heuristic for mixed-integer programs with 2-level variables, with an application to a telecommunication network design problem. *Networks*, 44(2):61–72, 2004. 11

[113] Matteo Fischetti, Andrea Lodi, and Domenico Salvagnin. Just MIP it! In *Matheuristics*, pages 39–70. Springer, 2010. 11

[114] E Feo Flushing and Gianni A Di Caro. Exploiting synergies between exact and heuristic methods in optimization: an application to the relay placement problem in wireless sensor networks. *Proc. of BIONETICS*, 2012. 16, 17

[115] Ricardo Fukasawa, Humberto Longo, Jens Lysgaard, Marcus Poggi de Aragão, Marcelo Reis, Eduardo Uchoa, and Renato F Werneck. Robust branch-and-cut-and-price for the

capacitated vehicle routing problem. *Mathematical programming*, 106(3):491–511, 2006. 28

[116] Michel Gendreau, Alain Hertz, and Gilbert Laporte. A tabu search heuristic for the vehicle routing problem. *Management science*, 40(10):1276–1290, 1994. 8

[117] Ilfat Ghamlouche, Teodor Gabriel Crainic, and Michel Gendreau. Cycle-based neighbourhoods for fixed-charge capacitated multicommodity network design. *Operations research*, 51(4):655–667, 2003. 8, 11, 22

[118] Ilfat Ghamlouche, Teodor Gabriel Crainic, and Michel Gendreau. Path relinking, cycle-based neighbourhoods and capacitated multicommodity network design. *Annals of Operations research*, 131(1-4):109–133, 2004. 8, 11

[119] Diptesh Ghosh. Solving medium to large sized euclidean generalized minimum spanning tree problems. 2003. 15

[120] Fred Glover. Ejection chains, reference structures and alternating path methods for traveling salesman problems. *Discrete Applied Mathematics*, 65(1):223–253, 1996. 34

[121] Fred Glover and M Laguna. Tabu search, 1997. *Kluwer Academic Publishers*, 1997. 14

[122] Bruce Golden, Subramanian Raghavan, and Daliborka Stanojević. Heuristic search for the generalized minimum spanning tree problem. *INFORMS Journal on Computing*, 17 (3):290–304, 2005. 15

[123] Bruce L Golden and Arjang A Assad. Or forumperspectives on vehicle routing: Exciting new developments. *Operations Research*, 34(5):803–810, 1986. 50

[124] Thiago M Gomes, Haroldo G Santos, and Marcone JF Souza. A pre-processing aware RINS based MIP heuristic. In *Hybrid Metaheuristics*, pages 1–11. Springer, 2013. 13

[125] Andrea Grosso, Federico Della Croce, and Roberto Tadei. An enhanced dynasearch neighborhood for the single-machine total weighted tardiness scheduling problem. *Operations Research Letters*, 32(1):68–72, 2004. 10

[126] Martin Gruber and Günther R. Raidl. (Meta-)heuristic separation of jump cuts in a branch&cut approach for the bounded diameter minimum spanning tree problem. In Vittorio Maniezzo, Thomas Stützle, and Stefan Voß, editors, *Matheuristics*, Annals of Information Systems, pages 209–229. Springer US, 2010. 19, 21

[127] Peter Hahn, Thomas Grant, and Nat Hall. A branch-and-bound algorithm for the quadratic assignment problem based on the hungarian method. *European Journal of Operational Research*, 108(3):629–640, 1998. 69

[128] Pierre Hansen and Nenad Mladenović. Variable neighborhood search: principles and applications. *European journal of operational research*, 130(3):449–467, 2001. 9

[129] Pierre Hansen, Nenad Mladenović, and Dragan Urošević. Variable neighborhood search and local branching. *Computers & Operations Research*, 33(10):3034–3045, 2006. 11, 14

[130] Mohamed Haouari and Jouhaina Chaouachi Siala. A hybrid lagrangian genetic algorithm for the prize collecting steiner tree problem. *Computers & Operations Research*, 33(5): 1274–1288, 2006. 25, 26

[131] Mike Hewitt, George L Nemhauser, and Martin WP Savelsbergh. Combining exact and heuristic approaches for the capacitated fixed-charge network flow problem. *INFORMS Journal on Computing*, 22(2):314–325, 2010. 8, 22

[132] Robert Hinterding. Mapping order-independent genes and the knapsack problem. In *Proceedings of the first IEEE Conference on Evolutionary Computation*. Citeseer, 1994. 25

[133] Holger H Hoos. Automated algorithm configuration and parameter tuning. In *Autonomous search*, pages 37–71. Springer, 2012. 47

[134] Bin Hu, Markus Leitner, and Günther R Raidl. Combining variable neighborhood search with integer linear programming for the generalized minimum spanning tree problem. *Journal of Heuristics*, 14(5):473–499, 2008. 15, 17

[135] Petros A Ioannou, Anastasios Chassiakos, Hossein Jula, and Ricardo Unglaub. Dynamic optimization of cargo movement by trucks in metropolitan areas with adjacent ports. Technical report, METRANS Transportation Center, 2002. 26

[136] Stefan Irnich and Daniel Villeneuve. The shortest-path problem with resource constraints and k-cycle elimination for k= 3. *INFORMS Journal on Computing*, 18(3):391–406, 2006. 50

[137] Mads Jepsen, Bjørn Petersen, Simon Spoorendonk, and David Pisinger. Subset-row inequalities applied to the vehicle-routing problem with time windows. *Operations Research*, 56(2):497–511, 2008. 50

[138] Kap Hwan Kim and Gyu-Pyo Hong. A heuristic rule for relocating blocks. *Computers & Operations Research*, 33(4):940–954, 2006. 12

[139] Scott Kirkpatrick, MP Vecchi, et al. Optimization by simulated annealing. *science*, 220 (4598):671–680, 1983. 25

[140] Niklas Kohl and Oli BG Madsen. An optimization algorithm for the vehicle routing problem with time windows based on lagrangian relaxation. *Operations Research*, 45(3): 395–406, 1997. 50

[141] Niklas Kohl, Jacques Desrosiers, Oli BG Madsen, Marius M Solomon, and Francois Soumis. 2-path cuts for the vehicle routing problem with time windows. *Transportation Science*, 33(1):101–116, 1999. 50

[142] Jasmina Lazić, Saïd Hanafi, Nenad Mladenović, and Dragan Urošević. Variable neighbourhood decomposition search for 0–1 mixed integer programs. *Computers & Operations Research*, 37(6):1055–1067, 2010. 14

[143] François-Xavier Le Louarn, Michel Gendreau, and Jean-Yves Potvin. GENI ants for the traveling salesman problem. *Annals of Operations Research*, 131(1-4):187–201, 2004. 17

[144] Markus Leitner and Günther R Raidl. Lagrangian decomposition, metaheuristics, and hybrid approaches for the design of the last mile in fiber optic networks. In *Hybrid Metaheuristics*, pages 158–174. Springer, 2008. 26

[145] JK Lenstra, M Desroches, MWP Savelbergh, and F Soumis. *Vehicle routing with time windows: optimization and approximation*. Elsevier Science Publishers, North-Holland, 1988. 50

[146] Gary Lewandowski and Anne Condon. Experiments with parallel graph coloring heuristics and applications of graph coloring. *Johnson & Trick*, page 309, 1996. 9

[147] Leo Liberti, Giacomo Nannicini, and Nenad Mladenović. A good recipe for solving MINLPs. In *Matheuristics*, pages 231–244. Springer, 2010. 11

[148] Ivana Ljubić, Peter Putz, and Juan-José Salazar-González. A MIP-based approach to solve the prize-collecting local access network design problem. *European Journal of Operational Research*, 2013. 23, 26

[149] Ramon Lopes, Vinicius WC Morais, Thiago F Noronha, and Vitor AA Souza. Heuristics and matheuristics for a real-life machine reassignment problem. *International Transactions in Operational Research*, 2014. 16, 17

[150] Manuel López-Ibáñez and Thomas Stützle. Automatic configuration of multi-objective aco algorithms. In *Swarm Intelligence*, pages 95–106. Springer, 2010. 50

[151] Manuel López-Ibánez, Jérémie Dubois-Lacoste, Thomas Stützle, and Mauro Birattari. The irace package: Iterated racing for automatic algorithm configuration. Technical report, Citeseer, 2011. 50

[152] Helena R Lourenço, Olivier C Martin, and Thomas Stützle. *Iterated local search*. Springer, 2003. 16, 69

[153] Daniele Manerba and Renata Mansini. An exact algorithm for the capacitated total quantity discount problem. *European Journal of Operational Research*, 222(2):287–300, 2012. 9

[154] Daniele Manerba and Renata Mansini. An effective matheuristic for the capacitated total quantity discount problem. *Computers & Operations Research*, 41:1–11, 2014. 9

[155] Vittorio Maniezzo. Exact and approximate nondeterministic tree-search procedures for the quadratic assignment problem. *INFORMS journal on computing*, 11(4):358–369, 1999. 69

[156] Vladislav Maraš, Jasmina Lazić, Tatjana Davidović, and Nenad Mladenović. Routing of barge container ships by mixed-integer programming heuristics. *Applied Soft Computing*, 13(8):3515–3528, 2013. 14

[157] Renaud Masson, Thibaut Vidal, Julien Michallet, Puca Huachi Vaz Penna, Vinicius Petrucci, Anand Subramanian, and Hugues Dubedout. An iterated local search heuristic for multi-capacity bin packing and machine reassignment problems. *Expert Systems with Applications*, 40(13):5266–5275, 2013. 16

[158] Thierry Mautor and Catherine Roucairol. A new exact algorithm for the solution of quadratic assignment problems. *Discrete Applied Mathematics*, 55(3):281–293, 1994. 69

[159] Snežana Mitrović-Minić and Abraham P Punnen. Variable intensity local search. In *Matheuristics*, pages 245–252. Springer, 2010. 13

[160] Nenad Mladenović and Pierre Hansen. Variable neighborhood search. *Computers & Operations Research*, 24(11):1097–1100, 1997. 9

[161] Yuichi Nagata and Olli Bräysy. Efficient local search limitation strategies for vehicle routing problems. In *Evolutionary Computation in Combinatorial Optimization*, pages 48–60. Springer, 2008. 43

[162] Sandra Ulrich Ngueveu, Christian Prins, and Roberto Wolfler Calvo. A hybrid tabu search for the m-peripatetic vehicle routing problem. In *Matheuristics*, pages 253–266. Springer, 2010. 15, 17

[163] Ibrahim Hassan Osman. Metastrategy simulated annealing and tabu search algorithms for the vehicle routing problem. *Annals of operations research*, 41(4):421–451, 1993. 43

[164] L Pardalos and MGC Resende. A greedy randomized adaptive search procedure for the quadratic assignment problem. *Quadratic Assignment and Related Problems, DIMACS Series on Discrete Mathematics and Theoretical Computer Science*, 16:237–261, 1994. 69

[165] Joao Pedro Pedroso and Mikio Kubo. Hybrid tabu search for lot sizing problems. In *Hybrid Metaheuristics*, pages 66–77. Springer, 2005. 22

[166] Paola Pellegrini, Franco Mascia, Thomas Stützle, and Mauro Birattari. On the sensitivity of reactive tabu search to its meta-parameters. *Soft Computing*, 18(11):2177–2190, 2014. viii, x, xi, 72, 74, 180, 181, 182, 183, 184, 185, 186, 194

[167] Guido Perboli, Roberto Tadei, and Daniele Vigo. The two-echelon capacitated vehicle routing problem: models and math-based heuristics. *Transportation Science*, 45(3):364–380, 2011. 14

[168] Pavel A Pevzner. 1-tuple DNA sequencing: computer analysis. *Journal of Biomolecular structure and dynamics*, 7(1):63–73, 1989. 1

[169] Pavel A Pevzner and Robert J Lipshutz. Towards DNA sequencing chips. In *Mathematical Foundations of Computer Science 1994*, pages 143–158. Springer, 1994. 1

[170] DT Pham, A Ghanbarzadeh, E Koc, S Otri, S Rahim, and M Zaidi. The bees algorithm-a novel tool for complex optimisation problems. In *Proceedings of the 2nd Virtual International Conference on Intelligent Production Machines and Systems (IPROMS 2006)*, pages 454–459, 2006. 17

[171] Sandro Pirkwieser and Günther R Raidl. Multiple variable neighborhood search enriched with ILP techniques for the periodic vehicle routing problem with time windows. In *Hybrid Metaheuristics*, pages 45–59. Springer, 2009. 23, 26

[172] Sandro Pirkwieser and Günther R Raidl. Variable neighborhood search coupled with ILP-based very large neighborhood searches for the (periodic) location-routing problem. In *Hybrid Metaheuristics*, pages 174–189. Springer, 2010. 9

[173] Sandro Pirkwieser, Günther R Raidl, and Jakob Puchinger. Combining lagrangian decomposition with an evolutionary algorithm for the knapsack constrained maximum spanning tree problem. In *Evolutionary Computation in Combinatorial Optimization*, pages 176–187. Springer, 2007. 25, 26

[174] Sandro Pirkwieser, Günther R Raidl, and Jakob Puchinger. A lagrangian decomposition/evolutionary algorithm hybrid for the knapsack constrained maximum spanning tree problem. In *Recent Advances in Evolutionary Computation for Combinatorial Optimization*, pages 69–85. Springer, 2008. 25

[175] Rapeepan Pitakaso, Christian Almeder, Karl F Doerner, and Richard F Hartl. Combining population-based and exact methods for multi-level capacitated lot-sizing problems. *International journal of production research*, 44(22):4755–4771, 2006. 26

[176] Yves Pochet and Mathieu Van Vyve. A general heuristic for production planning problems. *INFORMS Journal on Computing*, 16(3):316–327, 2004. 22

[177] Chandra A Poojari and John E Beasley. Improving Benders decomposition using a genetic algorithm. *European Journal of Operational Research*, 199(1):89–97, 2009. 20

[178] Petrica Claudiu Pop. *The generalized minimum spanning tree problem*. Twente University Press, 2002. 15

[179] Matthias Prandtstetter and Günther R Raidl. An integer linear programming approach and a hybrid variable neighborhood search for the car sequencing problem. *European Journal of Operational Research*, 191(3):1004–1022, 2008. 15, 17

[180] Jakob Puchinger and Günther R Raidl. Combining metaheuristics and exact algorithms in combinatorial optimization: a survey and classification. In *Artificial intelligence and knowledge engineering applications: a bioinspired approach*, pages 41–53. Springer, 2005. 6

[181] Jakob Puchinger and Günther R Raidl. Models and algorithms for three-stage two-dimensional bin packing. *European Journal of Operational Research*, 183(3):1304–1327, 2007. 20, 21

[182] Jakob Puchinger, Günther R Raidl, and Sandro Pirkwieser. Metaboosting: enhancing integer programming techniques by metaheuristics. In *Matheuristics*, pages 71–102. Springer, 2010. 19

[183] Günther R Raidl. An improved genetic algorithm for the multiconstrained 0-1 knapsack problem. In *Evolutionary Computation Proceedings, 1998. IEEE World Congress on Computational Intelligence., The 1998 IEEE International Conference on*, pages 207–211. IEEE, 1998. 25

[184] Günther R Raidl and Jakob Puchinger. Combining (integer) linear programming techniques and metaheuristics for combinatorial optimization. In *Hybrid Metaheuristics*, pages 31–62. Springer, 2008. 6

[185] Colin R Reeves. Feature article-genetic algorithms for the operations researcher. *INFORMS Journal on Computing*, 9(3):231–250, 1997. 14

[186] Walter Rei, Jean-François Cordeau, Michel Gendreau, and Patrick Soriano. Accelerating Benders decomposition by local branching. *INFORMS Journal on Computing*, 21(2): 333–345, 2009. 20

[187] Marc Reimann. Guiding ACO by problem relaxation: a case study on the symmetric TSP. In *Hybrid Metaheuristics*, pages 45–56. Springer, 2007. 16, 17

[188] Geraldo Ribeiro Filho and LA Nogueira Lorena. Constructive genetic algorithm and column generation: an application to graph coloring. In *Proceedings of APORS*, 2000. 20, 21

[189] Inmaculada Rodríguez-Martín and Juan José Salazar-González. A local branching heuristic for the capacitated fixed-charge network design problem. *Computers & Operations Research*, 37(3):575–581, 2010. 10

[190] Andrea Roli, Stefano Benedettini, Thomas Stützle, and Christian Blum. Large neighbourhood search algorithms for the founder sequence reconstruction problem. *Computers & operations research*, 39(2):213–224, 2012. 9

[191] Stefan Ropke and David Pisinger. An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. *Transportation science*, 40(4):455–472, 2006. 23

[192] Majid Salari, Paolo Toth, and Andrea Tramontani. An ILP improvement procedure for the open vehicle routing problem. *Computers & Operations Research*, 37(12):2106–2120, 2010. 8

[193] HG Santos, TAM Toffolo, S Ribas, and RAM Gomes. Integer programming techniques for the nurse rostering problem. In *Proceedings of PATAT*, pages 256–283, 2012. 9

[194] VI Sarvanov and NN Doroshko. Approximate solution of the traveling salesman problem by a local algorithm with scanning neighborhoods of factorial cardinality in cubic time. *Software: Algorithms and Programs*, 31:11–13, 1981. 8

[195] VI Sarvanov and NN Doroshko. The approximate solution of the traveling salesman problem by a local algorithm that searches neighborhoods of exponential cardinality in quadratic time. *Software: Algorithms and Programs*, 31:8–11, 1981. 9

[196] Verena Schmid, Karl F Doerner, Richard F Hartl, Martin WP Savelsbergh, and Wolfgang Stoecher. A hybrid solution approach for ready-mixed concrete delivery. *Transportation Science*, 43(1):70–85, 2009. 23

[197] Verena Schmid, Karl F Doerner, Richard F Hartl, and Juan-José Salazar-González. Hybridization of very large neighborhood search for ready-mixed concrete delivery problems. *Computers & Operations Research*, 37(3):559–574, 2010. 9

[198] Paul Shaw. Using constraint programming and local search methods to solve vehicle routing problems. In *Principles and Practice of Constraint ProgrammingCP98*, pages 417–431. Springer, 1998. 25

[199] Moshe Sniedovich and S Voß. The corridor method: a dynamic programming inspired metaheuristic. *Control and Cybernetics*, 35:551–578, 2006. 11

[200] Marius M Solomon. Algorithms for the vehicle routing and scheduling problems with time window constraints. *Operations research*, 35(2):254–265, 1987. vii, x, xi, 54, 59, 65, 66, 67, 154, 155, 156, 157, 158, 159, 166, 167, 168, 169, 170, 171, 187, 188

[201] Leon Steinberg. The backboard wiring problem: A placement algorithm. *Siam Review*, 3(1):37–50, 1961. 69

[202] Francesco Strappaveccia. *Many-core Algorithms for Combinatorial Optimization.* PhD thesis, Alma Mater Studiorum, University of Bologna, 2015. 44

[203] Johannes Strodl, Karl F Doerner, Fabien Tricoire, and Richard F Hartl. On index structures in hybrid metaheuristics for routing problems with hard feasibility checks: an application to the 2-dimensional loading vehicle routing problem. In *Hybrid Metaheuristics*, pages 160–173. Springer, 2010. 15, 17

[204] Thomas Stützle. Iterated local search for the quadratic assignment problem. *European Journal of Operational Research*, 174(3):1519–1539, 2006. 69

[205] E Taillard. Robust taboo search for the quadratic assignment problem. *Parallel computing*, 17(4):443–455, 1991. 69

[206] Éric Taillard, Philippe Badeau, Michel Gendreau, François Guertin, and Jean-Yves Potvin. A tabu search heuristic for the vehicle routing problem with soft time windows. *Transportation science*, 31(2):170–186, 1997. 43

[207] Éric D Taillard. A heuristic column generation method for the heterogeneous fleet VRP. *Revue française d'automatique, d'informatique et de recherche opérationnelle. Recherche opérationnelle*, 33(1):1–14, 1999. 25

[208] Fabien Tricoire, Karl F Doerner, Richard F Hartl, and Manuel Iori. Heuristic and exact algorithms for the multi-pile vehicle routing problem. *OR spectrum*, 33(4):931–959, 2011. 19

[209] William W Trigeiro, L Joseph Thomas, and John O McClain. Capacitated lot sizing with setup times. *Management science*, 35(3):353–366, 1989. 20

[210] Theodore Tsiligirides. Heuristic methods applied to orienteering. *Journal of the Operational Research Society*, pages 797–809, 1984. 12

[211] Eduardo Uchoa, Diego Pecin, Artur Pessoa, Marcus Poggi, Anand Subramanian, and Thibaut Vidal. New benchmark instances for the capacitated vehicle routing problem. vii, viii, ix, x, 44, 45, 46, 61, 62, 64, 118, 119, 120, 124, 125, 126, 127, 128, 129, 130, 131, 132, 133, 134, 135, 190, 191

[212] Shunji Umetani, Mutsunoti Yagiura, and Toshihide Ibaraki. One-dimensional cutting stock problem with a given number of setups: a hybrid approach of metaheuristics and linear programming. *Journal of Mathematical Modelling and Algorithms*, 5(1):43–64, 2006. 16

[213] Michel Vasquez, Jin-Kao Hao, et al. A hybrid approach for the 0-1 multidimensional knapsack problem. In *IJCAI*, pages 328–333, 2001. 24, 26

[214] Jakob Walla, Mario Ruthmair, and Günther R Raidl. Solving a video-server load rebalancing problem by mixed integer programming and hybrid variable neighborhood search. In *Hybrid Metaheuristics*, pages 84–99. Springer, 2009. 15, 17

[215] Masoud Yaghini, Mohammad Karimi, and Mohadeseh Rahbar. A hybrid metaheuristic approach for the capacitated p-median problem. *Applied Soft Computing*, 13(9):3922–3930, 2013. 14, 17

Appendices

Appendix A

Chapter 3

A.1 Computational results

A. CHAPTER 3

Table A.1: Lagrangean column generation computational results for instances by Uchoa et al. (211)

Instance	n	Lower Bound			Heuristic			Time(s)	Best known	
		Veh.	Dist	Gap(%)	Veh.	Dist.	Gap(%)		Veh.	Dist.
X-n101-k25	100	26	27236,77	1,28	26	28201	2,21	20	26	27591
X-n106-k14	105	14	26187,32	0,66	14	26947	2,22	40	14	26362
X-n110-k13	109	13	14694,68	1,85	13	15296	2,17	18	13	14971
X-n115-k10	114	10	12540,72	1,62	10	13068	2,52	111	10	12747
X-n120-k6	119	6	13011,53	2,40	6	13889	4,18	62	6	13332
X-n125-k30	124	30	55115,93	0,76	30	57399	3,35	26	30	55539
X-n129-k18	128	18	28676,22	0,91	18	30264	4,57	24	18	28940
X-n134-k13	133	13	10646,85	2,47	13	11462	5,00	135	13	10916
X-n139-k10	138	10	13302,41	2,12	10	13826	1,74	41	10	13590
X-n143-k7	142	7	15350,37	2,23	7	16510	5,16	944	7	15700
X-n148-k46	147	47	42938,17	1,17	47	44624	2,71	13	47	43448
X-n153-k22	152	23	20812,73	1,92	23	21775	2,62	119	23	21220
X-n157-k13	156	13	16721,29	0,92	13	17086	1,24	31	13	16876
X-n162-k11	161	11	13682,19	3,22	11	14417	1,97	258	11	14138
X-n167-k10	166	10	20284,21	1,33	10	22043	7,23	85	10	20557
X-n172-k51	171	53	45075,16	1,17	53	46542	2,05	33	53	45607
X-n176-k26	175	26	47166,72	1,35	26	49878	4,32	92	26	47812
X-n181-k23	180	23	25256,09	1,22	23	26034	1,82	27	23	25569
X-n186-k15	185	15	23771,67	1,55	15	25789	6,81	192	15	24145
X-n190-k8	189	8	16675,39	1,79	8	17737	4,46	568	8	16980
X-n195-k51	194	53	43720,09	1,14	53	45777	3,51	71	53	44225
X-n200-k36	199	36	58130,54	0,76	36	60630	3,50	90	36	58578
X-n204-k19	203	19	19235,99	1,68	19	20249	3,50	195	19	19565
X-n209-k16	208	16	30056,56	1,96	16	32686	6,62	77	16	30656
X-n214-k11	213	11	10680,91	1,61	11	12342	13,69	1597	11	10856
X-n219-k73	218	73	117210,87	0,33	73	117918	0,27	10	73	117595
X-n223-k34	222	34	39889,25	1,35	34	42334	4,69	78	34	40437
X-n228-k23	227	23	25032,67	2,76	23	27221	5,75	472	23	25742
X-n233-k16	232	17	18851,61	1,97	17	20427	6,22	1641	17	19230
X-n237-k14	236	14	26749,28	1,08	14	29009	7,27	138	14	27042
X-n242-k48	241	48	81955,57	0,96	48	86136	4,09	89	48	82751
X-n247-k47	246	51	36806,79	1,25	51	38242	2,60	512	51	37274
X-n251-k28	250	28	38218,06	1,20	28	40506	4,71	69	28	38684
X-n256-k16	255	17	18513,56	1,94	17	19344	2,46	910	17	18880

- datum not available.

Table A.1: Lagrangean column generation computational results for instances by Uchoa et al. (211)

Instance	n	Lower Bound			Heuristic			Time(s)	Best known	
		Veh.	Dist	Gap(%)	Veh.	Dist.	Gap(%)		Veh.	Dist.
X-n261-k13	260	13	26129,19	1,61	13	28572	7,58	4172	13	26558
X-n266-k58	265	58	74718,07	1,01	58	79356	5,14	104	58	75478
X-n270-k35	269	36	34710,82	1,64	36	36823	4,34	171	36	35291
X-n275-k28	274	28	21041,72	0,96	28	22104	4,04	66	28	21245
X-n280-k17	279	17	32706,02	2,38	17	36550	9,09	1970	17	33503
X-n284-k15	283	15	19994,7	1,14	15	21764	7,60	1619	15	20226
X-n289-k60	288	61	94300,62	0,89	61	100784	5,92	342	61	95151
X-n294-k50	293	51	46427,53	1,57	51	50046	6,10	564	51	47167
X-n298-k31	297	31	33795,06	1,27	31	37502	9,56	391	31	34231
X-n303-k21	302	21	21111,32	2,91	21	22856	5,11	4635	21	21744
X-n308-k13	307	13	25154,86	2,72	13	27728	7,23	4254	13	25859
X-n313-k71	312	72	93215,66	0,88	72	98118	4,33	375	72	94044
X-n317-k53	316	53	78120,53	0,30	53	79660	1,67	42	53	78355
X-n322-k28	321	28	29301,72	1,85	28	32535	8,98	722	28	29854
X-n327-k20	326	20	27083,79	1,71	20	30120	9,30	417	20	27556
X-n331-k15	330	15	30696,13	1,31	15	33368	7,28	625	15	31103
X-n336-k84	335	86	137850,69	0,94	86	144727	4,00	400	86	139165
X-n344-k43	343	43	41442,08	1,50	43	45720	8,67	141	43	42073
X-n351-k40	350	40	25569,23	1,41	40	30068	15,93	2572	40	25936
X-n359-k29	358	29	50988,46	1,01	29	56420	9,53	1442	29	51509
X-n367-k17	366	17	22336,13	2,09	17	24302	6,52	7385	17	22814
X-n376-k94	375	94	147246,7	0,32	94	148988	0,86	39	94	147713
X-n384-k52	383	53	65220,92	1,21	53	69932	5,92	396	53	66021
X-n393-k38	392	38	37803,61	1,22	38	41213	7,69	294	38	38269
X-n401-k29	400	29	65493,43	1,13	29	70494	6,42	7040	29	66243
X-n411-k19	410	19	19160,84	2,83	19	21550	9,29	9395	19	19718
X-n420-k130	419	130	106817,76	0,91	130	111683	3,60	184	130	107798
X-n429-k61	428	62	64717,81	1,20	62	69531	6,15	418	62	65501
X-n439-k37	438	37	35837,32	1,53	37	38200	4,96	291	37	36395
X-n449-k29	448	29	54236,22	2,03	29	64131	15,85	9772	29	55358
X-n459-k26	458	26	23582,03	2,48	26	26825	10,93	7390	26	24181
X-n469-k138	468	140	220596,76	0,66	140	228232	2,77	2797	140	222070
X-n480-k70	479	70	88619,73	1,02	70	94417	5,45	262	70	89535
X-n491-k59	490	60	65802,44	1,25	60	72049	8,13	6800	60	66633
X-n502-k39	501	39	68579,55	0,97	39	70912	2,40	657	39	69253

- datum not available.

A. CHAPTER 3

Table A.1: Lagrangean column generation computational results for instances by Uchoa et al. (211)

Instance	n	Lower Bound			Heuristic			Time(s)	Best known	
		Veh.	Dist	Gap(%)	Veh.	Dist.	Gap(%)		Veh.	Dist.
X-n513-k21	512	21	23270,19	3,85	21	26356	8,90	10532	21	24201
X-n524-k153	523	156	153794,37	0,59	156	156168	0,94	2583	156	154711
X-n536-k96	535	97	94188,89	0,98	97	99274	4,36	2292	97	95122
X-n548-k50	547	50	86230,91	0,68	50	93861	8,11	488	50	86822
X-n561-k42	560	-	-	100,00	42	46610	9,01	6406	42	42756
X-n573-k30	572	-	-	100,00	30	53207	4,78	7598	30	50780
X-n586-k159	585	159	189145,47	0,73	159	196469	3,11	3691	159	190543
X-n599-k92	598	94	107423,07	1,28	94	115246	5,91	873	94	108813
X-n613-k62	612	-	-	100,00	62	66313	10,93	7862	62	59778
X-n627-k43	626	43	61538,37	1,33	43	69987	12,22	2641	43	62366
X-n641-k35	640	35	62640,92	1,88	35	71426	11,88	6408	35	63839
X-n655-k131	654	131	106512,99	0,25	131	108278	1,40	168	131	106780
X-n670-k130	669	134	145436,75	0,86	134	152275	3,80	8368	134	146705
X-n685-k75	684	-	-	100,00	75	76869	12,34	7450	75	68425
X-n701-k44	700	-	-	100,00	44	90359	9,80	7224	44	82292
X-n716-k35	715	-	-	100,00	35	47917	10,09	8055	35	43525
X-n733-k159	732	160	134667,47	1,25	160	144023	5,62	2395	160	136366
X-n749-k98	748	-	-	100,00	98	86245	11,00	8057	98	77700
X-n766-k71	765	-	-	100,00	71	123763	7,92	7999	71	114683
X-n783-k48	782	-	-	100,00	48	80622	10,86	8635	48	72727
X-n801-k40	800	40	72827,58	1,03	40	82690	12,37	6251	40	73587
X-n819-k171	818	173	157049,94	0,98	173	165758	4,51	2950	173	158611
X-n837-k142	836	142	192506,36	0,91	142	204607	5,32	1367	142	194266
X-n856-k95	855	95	88472,16	0,72	95	93017	4,38	1279	95	89118
X-n876-k59	875	-	-	100,00	59	108414	8,72	8630	59	99715
X-n895-k37	894	-	-	100,00	38	60878	12,38	7139	38	54172
X-n916-k207	915	208	327370,25	0,75	208	339828	3,03	1766	208	329836
X-n936-k151	935	-	-	100,00	159	141636	6,41	8624	159	133105
X-n957-k87	956	87	84956,7	0,83	87	92238	7,66	2911	87	85672
X-n979-k58	978	-	-	100,00	58	130721	9,67	8629	58	119194
X-n1001-k43	1000	-	-	100,00	43	81520	12,07	8623	43	72742

- datum not available.

Table A.2: Lagrangean column generation computational results for instances of CVRP available at (7)

Instance	n	Lower Bound		Heuristic		Gap(%)	Time(s)
		Veh.	Dist	Veh.	Dist.		
mat179	178	9	662989,81	9	701141	5,44	445
mat344	343	10	83601,23	10	90657	7,78	6746
mat382	381	22	12021,46	22	12989	7,45	5433
mat544	543	31	16682,05	31	20857	20,02	10544
mat827	826	-	-	41	27731	-	7493
mat980	979	-	-	47	53346	-	8589

- datum not available.

Appendix B

Chapter 4

B.1 Computational results

Table B.1: Valid lower bounds before and after tuning for *ng*-route pricing for instances by Uchoa et al. (211)

Instance	n	Before tuning			After tuning			Time(s)	Best known	
		Veh.	Dist.	Gap(%)	Veh.	Dist.	Gap(%)		Veh.	Dist.
X-n101-k25	100	26	27221,01	1,34	26	**27221,41**	1,34	36	26	27591
X-n106-k14	105	14	25480,58	3,34	14	25477,81	3,35	277	14	26362
X-n110-k13	109	13	14676,09	1,97	13	**14676,88**	1,96	49	13	14971
X-n115-k10	114	10	12483,50	2,07	10	**12488,28**	2,03	318	10	12747
X-n120-k6	119	6	12938,80	2,95	6	12938,78	2,95	144	6	13332
X-n125-k30	124	30	55286,81	0,45	30	55279,49	0,47	155	30	55539
X-n129-k18	128	18	28410,94	1,83	18	**28411,23**	1,83	82	18	28940
X-n134-k13	133	13	10640,59	2,52	13	**10640,77**	2,52	686	13	10916
X-n139-k10	138	10	13301,16	2,13	10	**13302,43**	2,12	186	10	13590
X-n143-k7	142	7	15269,44	2,74	7	15250,41	2,86	5190	7	15700
X-n148-k46	147	47	43114,87	0,77	47	**43115,99**	0,76	36	47	43448
X-n153-k22	152	23	20939,89	1,32	23	20870,93	1,65	329	23	21220
X-n157-k13	156	13	16722,80	0,91	13	16721,21	0,92	166	13	16876
X-n162-k11	161	11	13680,91	3,23	11	**13681,12**	3,23	2288	11	14138
X-n167-k10	166	10	20194,17	1,76	10	**20194,33**	1,76	805	10	20557
X-n172-k51	171	53	45224,67	0,84	53	45224,64	0,84	250	53	45607
X-n176-k26	175	26	44819,49	6,26	26	**46312,86**	3,14	490	26	47812
X-n181-k23	180	23	25103,26	1,82	23	25099,90	1,83	137	23	25569
X-n186-k15	185	15	23775,82	1,53	15	**23777,15**	1,52	1279	15	24145
X-n190-k8	189	8	14506,31	14,57	8	**16280,24**	4,12	2844	8	16980
X-n195-k51	194	53	43808,96	0,94	53	**43810,05**	0,94	198	53	44225
X-n200-k36	199	36	58029,24	0,94	36	58028,02	0,94	379	36	58578
X-n204-k19	203	19	19181,41	1,96	19	**19182,64**	1,95	1453	19	19565
X-n209-k16	208	16	30056,44	1,96	16	30047,06	1,99	419	16	30656
X-n214-k11	213	11	10675,48	1,66	11	10672,62	1,69	7018	11	10856
X-n219-k73	218	73	117207,45	0,33	73	117206,88	0,33	72	73	117595
X-n223-k34	222	34	39886,39	1,36	34	**39887,40**	1,36	209	34	40437
X-n228-k23	227	23	0,00	100,00	23	**25093,05**	2,52	2324	23	25742
X-n233-k16	232	17	18839,29	2,03	17	18802,37	2,22	7860	17	19230
X-n237-k14	236	14	26691,01	1,30	14	26404,93	2,36	538	14	27042
X-n242-k48	241	48	82030,46	0,87	48	82025,02	0,88	217	48	82751
X-n247-k47	246	51	0,00	100,00	51	**36618,24**	1,76	1405	51	37274
X-n251-k28	250	28	38106,77	1,49	28	38103,94	1,50	450	28	38684
X-n256-k16	255	17	18203,80	3,58	17	18203,56	3,58	9541	17	18880
X-n261-k13	260	13	25520,57	3,91	13	**25884,86**	2,53	18012	13	26558

Table B.1: Valid lower bounds before and after tuning for *ng*-route pricing for instances by Uchoa et al. (211)

Instance	n	Before tuning			After tuning			Time(s)	Best known	
		Veh.	Dist.	Gap(%)	Veh.	Dist.	Gap(%)		Veh.	Dist.
X-n266-k58	265	58	74853,66	0,83	58	**74855,61**	0,82	81	58	75478
X-n270-k35	269	36	34658,54	1,79	36	**34660,21**	1,79	929	36	35291
X-n275-k28	274	28	20751,07	2,32	28	**21023,69**	1,04	475	28	21245
X-n280-k17	279	17	0,00	100,00	17	0,00	100,00	5298	17	33503
X-n284-k15	283	15	18105,29	10,49	15	**19614,97**	3,02	3103	15	20226
X-n289-k60	288	61	94583,59	0,60	61	94156,44	1,05	1281	61	95151
X-n294-k50	293	51	46461,69	1,50	51	**46464,20**	1,49	626	51	47167
X-n298-k31	297	31	33776,89	1,33	31	**33777,29**	1,33	730	31	34231
X-n303-k21	302	21	20885,40	3,95	21	**20979,26**	3,52	12529	21	21744
X-n308-k13	307	13	0,00	100,00	13	**24100,19**	6,80	8639	13	25859
X-n313-k71	312	72	91329,35	2,89	72	**93311,31**	0,78	1226	72	94044
X-n317-k53	316	53	76314,62	2,60	53	75932,14	3,09	285	53	78355
X-n322-k28	321	28	29294,24	1,87	28	**29294,85**	1,87	2356	28	29854
X-n327-k20	326	20	26800,25	2,74	20	**27066,64**	1,78	2056	20	27556
X-n331-k15	330	15	30625,47	1,54	15	30479,36	2,01	1607	15	31103
X-n336-k84	335	86	130765,49	6,04	86	**138232,57**	0,67	1268	86	139165
X-n344-k43	343	43	41458,68	1,46	43	41448,61	1,48	273	43	42073
X-n351-k40	350	40	0,00	100,00	40	**25582,89**	1,36	7246	40	25936
X-n359-k29	358	29	0,00	100,00	29	**49578,75**	3,75	1984	29	51509
X-n367-k17	366	17	8014,62	64,87	17	0,00	100,00	13144	17	22814
X-n376-k94	375	94	147246,11	0,32	94	**147251,11**	0,31	234	94	147713
X-n384-k52	383	53	65248,26	1,17	53	65235,16	1,19	1749	53	66021
X-n393-k38	392	38	37628,94	1,67	38	**37768,58**	1,31	1035	38	38269

Table B.2: Valid lower bounds before and after tuning for (q, i)-route pricing for instances by Uchoa et al. (211)

Instance	n	Before tuning			After tuning			Time(s)	Best known	
		Veh.	Dist.	Gap(%)	Veh.	Dist.	Gap(%)		Veh.	Dist.
X-n101-k25	100	26	27077,67	1,86	26	**27079,31**	1,85	35	26	27591
X-n106-k14	105	14	25409,14	3,61	14	**25409,72**	3,61	94	14	26362
X-n110-k13	109	13	14509,36	3,08	13	**14510,49**	3,08	55	13	14971
X-n115-k10	114	10	12345,75	3,15	10	**12346,23**	3,14	105	10	12747
X-n120-k6	119	6	12783,15	4,12	6	**12783,69**	4,11	159	6	13332
X-n125-k30	124	30	55210,67	0,59	30	55197,68	0,61	68	30	55539
X-n129-k18	128	18	28364,52	1,99	18	28363,93	1,99	84	18	28940
X-n134-k13	133	13	10561,83	3,24	13	**10562,53**	3,24	182	13	10916
X-n139-k10	138	10	13051,59	3,96	10	**13052,04**	3,96	141	10	13590
X-n143-k7	142	7	14902,30	5,08	7	14896,25	5,12	499	7	15700
X-n148-k46	147	47	43016,96	0,99	47	**43018,51**	0,99	41	47	43448
X-n153-k22	152	23	19780,67	6,78	23	**20856,93**	1,71	151	23	21220
X-n157-k13	156	13	16580,74	1,75	13	**16581,08**	1,75	137	13	16876
X-n162-k11	161	11	13390,30	5,29	11	**13391,08**	5,28	361	11	14138
X-n167-k10	166	10	19787,86	3,74	10	**19788,15**	3,74	240	10	20557
X-n172-k51	171	53	45115,97	1,08	53	45115,86	1,08	120	53	45607
X-n176-k26	175	26	44822,83	6,25	26	**45855,27**	4,09	307	26	47812
X-n181-k23	180	23	25009,27	2,19	23	25009,13	2,19	130	23	25569
X-n186-k15	185	15	23424,27	2,99	15	**23424,49**	2,98	410	15	24145
X-n190-k8	189	8	14637,63	13,79	8	**15855,11**	6,62	759	8	16980
X-n195-k51	194	53	43683,18	1,23	53	**43684,85**	1,22	100	53	44225
X-n200-k36	199	36	57944,39	1,08	36	**57944,42**	1,08	128	36	58578
X-n204-k19	203	19	19038,00	2,69	19	**19038,74**	2,69	347	19	19565
X-n209-k16	208	16	29698,14	3,12	16	29697,65	3,13	266	16	30656
X-n214-k11	213	11	10025,89	7,65	11	**10515,55**	3,14	991	11	10856
X-n219-k73	218	73	117207,35	0,33	73	**117207,51**	0,33	86	73	117595
X-n223-k34	222	34	39657,59	1,93	34	**39659,39**	1,92	189	34	40437
X-n228-k23	227	23	23688,58	7,98	23	**24690,83**	4,08	465	23	25742
X-n233-k16	232	17	18506,87	3,76	17	**18523,66**	3,67	802	17	19230
X-n237-k14	236	14	26381,26	2,44	14	26172,14	3,22	439	14	27042
X-n242-k48	241	48	81857,25	1,08	48	**81859,65**	1,08	181	48	82751
X-n247-k47	246	51	0,00	100,00	51	**35992,07**	3,44	556	51	37274
X-n251-k28	250	28	37806,23	2,27	28	37799,95	2,29	303	28	38684
X-n256-k16	255	17	17947,60	4,94	17	**17948,22**	4,94	1122	17	18880
X-n261-k13	260	13	25154,94	5,28	13	25063,32	5,63	1914	13	26558

Table B.2: Valid lower bounds before and after tuning for (q, i)-route pricing for instances by Uchoa et al. (211)

Instance	n	Before tuning			After tuning			Time(s)	Best known	
		Veh.	Dist.	Gap(%)	Veh.	Dist.	Gap(%)		Veh.	Dist.
X-n266-k58	265	58	74436,30	1,38	58	**74440,50**	1,37	63	58	75478
X-n270-k35	269	36	34390,66	2,55	36	**34391,31**	2,55	447	36	35291
X-n275-k28	274	28	20826,14	1,97	28	20737,26	2,39	294	28	21245
X-n280-k17	279	17	23407,31	30,13	17	**30911,58**	7,73	1411	17	33503
X-n284-k15	283	15	0,00	100,00	15	**19144,66**	5,35	1041	15	20226
X-n289-k60	288	61	89595,11	5,84	61	**94265,59**	0,93	398	61	95151
X-n294-k50	293	51	46098,00	2,27	51	**46098,53**	2,27	400	51	47167
X-n298-k31	297	31	33431,58	2,34	31	**33432,58**	2,33	445	31	34231
X-n303-k21	302	21	19578,47	9,96	21	**20762,38**	4,51	1733	21	21744
X-n308-k13	307	13	0,00	100,00	13	**23534,10**	8,99	2130	13	25859
X-n313-k71	312	72	93352,58	0,74	72	92155,36	2,01	306	72	94044
X-n317-k53	316	53	77628,01	0,93	53	75267,30	3,94	229	53	78355
X-n322-k28	321	28	28855,82	3,34	28	**28857,64**	3,34	642	28	29854
X-n327-k20	326	20	25248,24	8,37	20	**26651,53**	3,28	894	20	27556
X-n331-k15	330	15	29566,89	4,94	15	**29642,95**	4,69	1126	15	31103
X-n336-k84	335	86	128453,26	7,70	86	**137486,82**	1,21	624	86	139165
X-n344-k43	343	43	41193,63	2,09	43	**41194,81**	2,09	208	43	42073
X-n351-k40	350	40	0,00	100,00	40	**25065,60**	3,36	1349	40	25936
X-n359-k29	358	29	0,00	100,00	29	**49598,41**	3,71	1008	29	51509
X-n367-k17	366	17	15442,91	32,31	17	0,00	100,00	2722	17	22814
X-n376-k94	375	94	147063,60	0,44	94	147061,89	0,44	199	94	147713
X-n384-k52	383	53	64790,46	1,86	53	64786,50	1,87	726	53	66021
X-n393-k38	392	38	36705,81	4,08	38	**37457,98**	2,12	331	38	38269

Table B.3: Valid lower bounds before and after tuning for (q, i)-route with 2-cycles pricing for instances by Uchoa et al. (211)

Instance	n	Before tuning			After tuning			Time(s)	Best known	
		Veh.	Dist.	Gap(%)	Veh.	Dist.	Gap(%)		Veh.	Dist.
X-n101-k25	100	26	26786,56	2,92	26	**26787,44**	2,91	68	26	27591
X-n106-k14	105	14	25184,70	4,47	14	**25185,92**	4,46	102	14	26362
X-n110-k13	109	13	13923,46	7,00	13	**13924,20**	6,99	68	13	14971
X-n115-k10	114	10	11976,00	6,05	10	**11976,97**	6,04	124	10	12747
X-n120-k6	119	6	12307,12	7,69	6	12303,68	7,71	157	6	13332
X-n125-k30	124	30	54995,84	0,98	30	54981,77	1,00	51	30	55539
X-n129-k18	128	18	27829,73	3,84	18	**27830,12**	3,84	73	18	28940
X-n134-k13	133	13	10171,78	6,82	13	10171,26	6,82	161	13	10916
X-n139-k10	138	10	12157,73	10,54	10	**12158,07**	10,54	148	10	13590
X-n143-k7	142	7	14302,39	8,90	7	14268,62	9,12	341	7	15700
X-n148-k46	147	47	42340,09	2,55	47	**42343,67**	2,54	44	47	43448
X-n153-k22	152	23	20790,19	2,03	23	20715,76	2,38	134	23	21220
X-n157-k13	156	13	16299,50	3,42	13	16245,04	3,74	139	13	16876
X-n162-k11	161	11	12734,46	9,93	11	**12734,76**	9,93	302	11	14138
X-n167-k10	166	10	19087,12	7,15	10	19084,62	7,16	230	10	20557
X-n172-k51	171	53	44607,29	2,19	53	44579,33	2,25	106	53	45607
X-n176-k26	175	26	46084,92	3,61	26	**46229,80**	3,31	180	26	47812
X-n181-k23	180	23	24541,94	4,02	23	24399,76	4,57	116	23	25569
X-n186-k15	185	15	22797,90	5,58	15	**22798,15**	5,58	321	15	24145
X-n190-k8	189	8	15868,46	6,55	8	15575,89	8,27	529	8	16980
X-n195-k51	194	53	43353,08	1,97	53	**43354,01**	1,97	166	53	44225
X-n200-k36	199	36	57663,10	1,56	36	57162,44	2,42	131	36	58578
X-n204-k19	203	19	17960,92	8,20	19	17960,74	8,20	302	19	19565
X-n209-k16	208	16	28768,90	6,16	16	**28769,52**	6,15	261	16	30656
X-n214-k11	213	11	9860,26	9,17	11	**9872,17**	9,06	826	11	10856
X-n219-k73	218	73	116507,04	0,93	73	**116507,19**	0,93	85	73	117595
X-n223-k34	222	34	38970,08	3,63	34	**38971,07**	3,63	175	34	40437
X-n228-k23	227	23	24528,62	4,71	23	24103,28	6,37	335	23	25742
X-n233-k16	232	17	17786,97	7,50	17	17677,71	8,07	724	17	19230
X-n237-k14	236	14	25630,33	5,22	14	25617,73	5,27	506	14	27042
X-n242-k48	241	48	81058,69	2,05	48	81055,76	2,05	207	48	82751
X-n247-k47	246	51	36642,74	1,69	51	36329,94	2,53	420	51	37274
X-n251-k28	250	28	36915,15	4,57	28	36909,83	4,59	271	28	38684
X-n256-k16	255	17	17159,80	9,11	17	**17161,25**	9,10	874	17	18880
X-n261-k13	260	13	24753,49	6,79	13	24512,32	7,70	1316	13	26558

Table B.3: Valid lower bounds before and after tuning for (q, i)-route with 2-cycles pricing for instances by Uchoa et al. (211)

Instance	n	Before tuning			After tuning			Time(s)	Best known	
		Veh.	Dist.	Gap(%)	Veh.	Dist.	Gap(%)		Veh.	Dist.
X-n266-k58	265	58	73690,95	2,37	58	**73693,95**	2,36	53	58	75478
X-n270-k35	269	36	33613,33	4,75	36	**33614,51**	4,75	237	36	35291
X-n275-k28	274	28	20148,93	5,16	28	20120,35	5,29	294	28	21245
X-n280-k17	279	17	30016,15	10,41	17	29655,25	11,48	1257	17	33503
X-n284-k15	283	15	18073,43	10,64	15	**18432,00**	8,87	958	15	20226
X-n289-k60	288	61	93590,43	1,64	61	93226,09	2,02	213	61	95151
X-n294-k50	293	51	45188,54	4,19	51	**45191,25**	4,19	345	51	47167
X-n298-k31	297	31	32646,74	4,63	31	32582,70	4,82	461	31	34231
X-n303-k21	302	21	20121,87	7,46	21	19870,56	8,62	1317	21	21744
X-n308-k13	307	13	22731,99	12,09	13	**22910,70**	11,40	2159	13	25859
X-n313-k71	312	72	92444,56	1,70	72	92359,04	1,79	287	72	94044
X-n317-k53	316	53	72934,39	6,92	53	**76678,01**	2,14	294	53	78355
X-n322-k28	321	28	27952,08	6,37	28	**27952,40**	6,37	417	28	29854
X-n327-k20	326	20	25916,72	5,95	20	25812,32	6,33	1025	20	27556
X-n331-k15	330	15	29390,75	5,51	15	29203,70	6,11	1345	15	31103
X-n336-k84	335	86	137468,60	1,22	86	137078,70	1,50	441	86	139165
X-n344-k43	343	43	40217,74	4,41	43	**40219,96**	4,40	111	43	42073
X-n351-k40	350	40	24734,83	4,63	40	21787,00	16,00	1012	40	25936
X-n359-k29	358	29	49210,90	4,46	29	47775,60	7,25	937	29	51509
X-n367-k17	366	17	20182,22	11,54	17	**20863,33**	8,55	2915	17	22814
X-n376-k94	375	94	146294,28	0,96	94	146269,38	0,98	254	94	147713
X-n384-k52	383	53	63702,76	3,51	53	**63705,63**	3,51	583	53	66021
X-n393-k38	392	38	36340,89	5,04	38	36331,35	5,06	257	38	38269

Table B.4: Feasible solution values before and after tuning for *ng*-route pricing for instances by Uchoa et al. (211)

Instance	n	Before tuning			After tuning			Best known	
		Veh.	Dist.	Gap(%)	Veh.	Dist.	Gap(%)	Veh.	Dist.
X-n101-k25	100	26	28185	2,15	26	27978	1,40	26	27591
X-n106-k14	105	14	26810	1,70	14	26932	2,16	14	26362
X-n110-k13	109	13	15118	0,98	13	15127	1,04	13	14971
X-n115-k10	114	10	12767	0,16	10	12767	0,16	10	12747
X-n120-k6	119	6	13569	1,78	6	13533	1,51	6	13332
X-n125-k30	124	30	56669	2,03	30	56463	1,66	30	55539
X-n129-k18	128	18	29480	1,87	18	29693	2,60	18	28940
X-n134-k13	133	13	11260	3,15	13	11265	3,20	13	10916
X-n139-k10	138	10	13806	1,59	10	13812	1,63	10	13590
X-n143-k7	142	7	16143	2,82	7	16299	3,82	7	15700
X-n148-k46	147	46	44209	1,75	46	44290	1,94	47	43448
X-n153-k22	152	22	22277	4,98	23	21511	1,37	23	21220
X-n157-k13	156	13	17089	1,26	13	17082	1,22	13	16876
X-n162-k11	161	11	14279	1,00	11	14270	0,93	11	14138
X-n167-k10	166	10	21409	4,14	10	21308	3,65	10	20557
X-n172-k51	171	52	47145	3,37	52	47260	3,62	53	45607
X-n176-k26	175	26	49135	2,77	26	48813	2,09	26	47812
X-n181-k23	180	23	25862	1,15	23	25936	1,44	23	25569
X-n186-k15	185	15	24926	3,23	15	24864	2,98	15	24145
X-n190-k8	189	8	17603	3,67	8	17573	3,49	8	16980
X-n195-k51	194	52	46351	4,81	52	46732	5,67	53	44225
X-n200-k36	199	36	59981	2,40	36	60165	2,71	36	58578
X-n204-k19	203	19	20114	2,81	19	20075	2,61	19	19565
X-n209-k16	208	16	32058	4,57	16	32129	4,80	16	30656
X-n214-k11	213	11	12010	10,63	11	12177	12,17	11	10856
X-n219-k73	218	73	117802	0,18	73	117779	0,16	73	117595
X-n223-k34	222	34	41408	2,40	34	41513	2,66	34	40437
X-n228-k23	227	23	26604	3,35	23	26529	3,06	23	25742
X-n233-k16	232	17	20074	4,39	17	19991	3,96	17	19230
X-n237-k14	236	14	28047	3,72	14	28222	4,36	14	27042
X-n242-k48	241	48	85108	2,85	48	85153	2,90	48	82751
X-n247-k47	246	50	38476	3,22	50	38359	2,91	51	37274
X-n251-k28	250	28	40272	4,11	28	40190	3,89	28	38684
X-n256-k16	255	17	19201	1,70	17	19288	2,16	17	18880
X-n261-k13	260	13	28151	6,00	13	28031	5,55	13	26558

Table B.4: Feasible solution values before and after tuning for *ng*-route pricing for instances by Uchoa et al. (211)

Instance	n	Before tuning			After tuning			Best known	
		Veh.	Dist.	Gap(%)	Veh.	Dist.	Gap(%)	Veh.	Dist.
X-n266-k58	265	58	77742	3,00	58	78410	3,88	58	75478
X-n270-k35	269	36	36669	3,90	36	36565	3,61	36	35291
X-n275-k28	274	28	22004	3,57	28	21836	2,78	28	21245
X-n280-k17	279	17	35887	7,12	17	35289	5,33	17	33503
X-n284-k15	283	15	21608	6,83	15	21383	5,72	15	20226
X-n289-k60	288	61	100153	5,26	61	99571	4,65	61	95151
X-n294-k50	293	50	54616	15,79	50	55336	17,32	51	47167
X-n298-k31	297	31	35320	3,18	31	35607	4,02	31	34231
X-n303-k21	302	21	22764	4,69	21	22783	4,78	21	21744
X-n308-k13	307	13	27238	5,33	13	27461	6,20	13	25859
X-n313-k71	312	72	98949	5,22	72	97357	3,52	72	94044
X-n317-k53	316	53	79575	1,56	53	79484	1,44	53	78355
X-n322-k28	321	28	32291	8,16	28	32432	8,64	28	29854
X-n327-k20	326	20	29624	7,50	20	29164	5,84	20	27556
X-n331-k15	330	15	32574	4,73	15	33028	6,19	15	31103
X-n336-k84	335	85	145875	4,82	85	145038	4,22	86	139165
X-n344-k43	343	43	46139	9,66	43	49335	17,26	43	42073
X-n351-k40	350	40	28073	8,24	40	28594	10,25	40	25936
X-n359-k29	358	29	55088	6,95	29	54571	5,94	29	51509
X-n367-k17	367	17	24434	7,10	17	24284	6,44	17	22814
X-n376-k94	375	94	148414	0,47	94	148430	0,49	94	147713
X-n384-k52	383	53	68917	4,39	53	69320	5,00	53	66021
X-n393-k38	392	38	40885	6,84	38	40772	6,54	38	38269

Table B.5: Feasible solution values before and after tuning for (q, i)-route pricing for instances by Uchoa et al. (211)

Instance	n	Before tuning			After tuning			Best known	
		Veh.	Dist.	Gap(%)	Veh.	Dist.	Gap(%)	Veh.	Dist.
X-n101-k25	100	26	28133	1,96	26	28037	1,62	26	27591
X-n106-k14	105	14	26903	2,05	14	26812	1,71	14	26362
X-n110-k13	109	13	15155	1,23	13	15143	1,15	13	14971
X-n115-k10	114	10	12827	0,63	10	12860	0,89	10	12747
X-n120-k6	119	6	13815	3,62	6	13750	3,14	6	13332
X-n125-k30	124	30	56982	2,60	30	56791	2,25	30	55539
X-n129-k18	128	18	29735	2,75	18	29669	2,52	18	28940
X-n134-k13	133	13	11208	2,67	13	11276	3,30	13	10916
X-n139-k10	138	10	13881	2,14	10	13889	2,20	10	13590
X-n143-k7	142	7	16382	4,34	7	16481	4,97	7	15700
X-n148-k46	147	46	44778	3,06	46	44927	3,40	47	43448
X-n153-k22	152	22	22408	5,60	22	22129	4,28	23	21220
X-n157-k13	156	13	17212	1,99	13	17184	1,83	13	16876
X-n162-k11	161	11	14303	1,17	11	14269	0,93	11	14138
X-n167-k10	166	10	21653	5,33	10	21567	4,91	10	20557
X-n172-k51	171	52	46619	2,22	52	47014	3,09	53	45607
X-n176-k26	175	26	49198	2,90	26	49000	2,48	26	47812
X-n181-k23	180	23	25771	0,79	23	25933	1,42	23	25569
X-n186-k15	185	15	25438	5,36	15	25371	5,08	15	24145
X-n190-k8	189	8	17528	3,23	8	17566	3,45	8	16980
X-n195-k51	194	52	46358	4,82	52	46732	5,67	53	44225
X-n200-k36	199	36	61005	4,14	36	61166	4,42	36	58578
X-n204-k19	203	19	20249	3,50	19	20088	2,67	19	19565
X-n209-k16	208	16	32399	5,69	16	32430	5,79	16	30656
X-n214-k11	213	11	12675	16,76	11	12458	14,76	11	10856
X-n219-k73	218	73	117810	0,18	73	117795	0,17	73	117595
X-n223-k34	222	34	41741	3,22	34	41887	3,59	34	40437
X-n228-k23	227	23	26529	3,06	23	26597	3,32	23	25742
X-n233-k16	232	17	20110	4,58	17	20014	4,08	17	19230
X-n237-k14	236	14	28543	5,55	14	28625	5,85	14	27042
X-n242-k48	241	48	85314	3,10	48	85193	2,95	48	82751
X-n247-k47	246	50	38261	2,65	50	38036	2,04	51	37274
X-n251-k28	250	28	40210	3,94	28	40499	4,69	28	38684
X-n256-k16	255	17	19407	2,79	17	19438	2,96	17	18880

- datum not available.

Table B.5: Feasible solution values before and after tuning for (q, i)-route pricing for instances by Uchoa et al. (211)

Instance	n	Before tuning			After tuning			Best known	
		Veh.	Dist.	Gap(%)	Veh.	Dist.	Gap(%)	Veh.	Dist.
X-n261-k13	260	13	28967	9,07	13	28900	8,82	13	26558
X-n266-k58	265	-	-	-	-	-	-	58	75478
X-n270-k35	269	36	36872	4,48	36	37030	4,93	36	35291
X-n275-k28	274	28	21990	3,51	28	21965	3,39	28	21245
X-n280-k17	279	17	35772	6,77	17	35497	5,95	17	33503
X-n284-k15	283	15	21661	7,09	15	21633	6,96	15	20226
X-n289-k60	288	61	100562	5,69	61	98797	3,83	61	95151
X-n294-k50	293	50	55862	18,43	50	56079	18,89	51	47167
X-n298-k31	297	31	36866	7,70	31	36466	6,53	31	34231
X-n303-k21	302	21	22957	5,58	21	22713	4,46	21	21744
X-n308-k13	307	13	27428	6,07	13	27495	6,33	13	25859
X-n313-k71	312	72	97659	3,84	72	98302	4,53	72	94044
X-n317-k53	316	53	79399	1,33	53	79329	1,24	53	78355
X-n322-k28	321	28	32969	10,43	28	32881	10,14	28	29854
X-n327-k20	326	20	29901	8,51	20	29584	7,36	20	27556
X-n331-k15	330	15	33381	7,32	15	33381	7,32	15	31103
X-n336-k84	335	85	147660	6,10	85	145798	4,77	86	139165
X-n344-k43	343	-	-	-	-	-	-	43	42073
X-n351-k40	350	40	29168	12,46	40	29335	13,11	40	25936
X-n359-k29	358	29	55625	7,99	29	55518	7,78	29	51509
X-n367-k17	366	17	24466	7,24	17	24403	6,97	17	22814
X-n376-k94	375	94	148710	0,67	94	148598	0,60	94	147713
X-n384-k52	383	53	69328	5,01	53	69530	5,31	53	66021
X-n393-k38	392	38	41367	8,10	38	41329	8,00	38	38269

- datum not available.

Table B.6: Feasible solution values before and after tuning for (q, i)-route with 2-cycles pricing for instances by Uchoa et al. (211)

Instance	n	Before tuning			After tuning			Best known	
		Veh.	Dist.	Gap(%)	Veh.	Dist.	Gap(%)	Veh.	Dist.
X-n101-k25	100	26	28136	1,98	26	28077	1,76	26	27591
X-n106-k14	105	14	26885	1,98	14	26986	2,37	14	26362
X-n110-k13	109	13	15233	1,75	13	15269	1,99	13	14971
X-n115-k10	114	10	12892	1,14	10	12901	1,21	10	12747
X-n120-k6	119	6	13878	4,10	6	13795	3,47	6	13332
X-n125-k30	124	30	56990	2,61	30	56477	1,69	30	55539
X-n129-k18	128	18	30124	4,09	18	30393	5,02	18	28940
X-n134-k13	133	13	11446	4,86	13	11393	4,37	13	10916
X-n139-k10	138	10	14248	4,84	10	14150	4,12	10	13590
X-n143-k7	142	7	16509	5,15	7	16519	5,22	7	15700
X-n148-k46	147	46	44698	2,88	46	45433	4,57	47	43448
X-n153-k22	152	23	21661	2,08	22	24344	14,72	23	21220
X-n157-k13	156	13	17376	2,96	13	17494	3,66	13	16876
X-n162-k11	161	11	14381	1,72	11	14476	2,39	11	14138
X-n167-k10	166	10	22036	7,19	10	21920	6,63	10	20557
X-n172-k51	171	52	47503	4,16	52	47755	4,71	53	45607
X-n176-k26	175	26	49690	3,93	26	49251	3,01	26	47812
X-n181-k23	180	23	26058	1,91	23	26069	1,96	23	25569
X-n186-k15	185	15	25665	6,30	15	25797	6,84	15	24145
X-n190-k8	189	8	17652	3,96	8	17675	4,09	8	16980
X-n195-k51	194	52	47250	6,84	52	47013	6,30	53	44225
X-n200-k36	199	-	-	-	36	62932	7,43	36	58578
X-n204-k19	203	19	20435	4,45	19	20384	4,19	19	19565
X-n209-k16	208	16	32558	6,20	16	32454	5,87	16	30656
X-n214-k11	213	11	13044	20,15	11	13138	21,02	11	10856
X-n219-k73	218	73	117851	0,22	73	117869	0,23	73	117595
X-n223-k34	222	34	42292	4,59	34	42605	5,36	34	40437
X-n228-k23	227	23	26817	4,18	23	26687	3,67	23	25742
X-n233-k16	232	17	20348	5,81	17	20255	5,33	17	19230
X-n237-k14	236	14	28825	6,59	14	28792	6,47	14	27042
X-n242-k48	241	48	85621	3,47	48	85650	3,50	48	82751
X-n247-k47	246	50	38494	3,27	50	38532	3,38	51	37274
X-n251-k28	250	28	40466	4,61	28	40676	5,15	28	38684
X-n256-k16	255	17	19491	3,24	17	19588	3,75	17	18880

- datum not available.

Table B.6: Feasible solution values before and after tuning for (q, i)-route with 2-cycles pricing for instances by Uchoa et al. (211)

Instance	n	Before tuning			After tuning			Best known	
		Veh.	Dist.	Gap(%)	Veh.	Dist.	Gap(%)	Veh.	Dist.
X-n261-k13	260	13	28976	9,10	13	28981	9,12	13	26558
X-n266-k58	265	-	-	-	-	-	-	58	75478
X-n270-k35	269	36	37557	6,42	36	37314	5,73	36	35291
X-n275-k28	274	28	21960	3,37	28	22067	3,87	28	21245
X-n280-k17	279	17	36179	7,99	17	36251	8,20	17	33503
X-n284-k15	283	15	21704	7,31	15	21752	7,54	15	20226
X-n289-k60	288	61	100213	5,32	61	100141	5,24	61	95151
X-n294-k50	293	51	49713	5,40	51	49744	5,46	51	47167
X-n298-k31	297	31	38081	11,25	31	37594	9,82	31	34231
X-n303-k21	302	21	22956	5,57	21	22753	4,64	21	21744
X-n308-k13	307	13	27613	6,78	13	27711	7,16	13	25859
X-n313-k71	312	72	98693	4,94	72	98108	4,32	72	94044
X-n317-k53	316	53	79528	1,50	53	79508	1,47	53	78355
X-n322-k28	321	28	34585	15,85	28	33734	13,00	28	29854
X-n327-k20	326	20	30100	9,23	20	30104	9,25	20	27556
X-n331-k15	330	15	33134	6,53	15	33260	6,94	15	31103
X-n336-k84	335	85	146387	5,19	85	147930	6,30	86	139165
X-n344-k43	343	-	-	-	-	-	-	43	42073
X-n351-k40	350	40	30002	15,68	40	30083	15,99	40	25936
X-n359-k29	358	29	56122	8,96	29	55904	8,53	29	51509
X-n367-k17	366	17	24655	8,07	17	25020	9,67	17	22814
X-n376-k94	375	94	148850	0,77	94	148999	0,87	94	147713
X-n384-k52	383	53	70549	6,86	53	70796	7,23	53	66021
X-n393-k38	392	38	41574	8,64	38	42034	9,84	38	38269

- datum not available.

Table B.7: Valid lower bounds before and after tuning for *ng*-route pricing for instances A, B, P, E, M

Instance	n	Before tuning			After tuning			Time(s)	Best known	
		Veh.	Dist.	Gap(%)	Veh.	Dist.	Gap(%)		Veh.	Dist.
A-n32-k5	31	5	757,62	3,36	5	**757,71**	3,35	1	5	784
A-n33-k5	32	5	651,94	1,37	5	651,94	1,37	1	5	661
A-n33-k6	32	6	726,56	2,08	6	**726,57**	2,08	2	6	742
A-n34-k5	33	5	742,69	4,54	5	**742,87**	4,52	2	5	778
A-n36-k5	35	5	772,10	3,37	5	**772,15**	3,36	2	5	799
A-n37-k5	36	5	656,27	1,90	5	**656,32**	1,90	2	5	669
A-n37-k6	36	6	922,57	2,79	6	**922,60**	2,78	2	6	949
A-n38-k5	37	5	694,70	4,84	5	**694,77**	4,83	2	5	730
A-n39-k5	38	5	799,48	2,74	5	**799,59**	2,73	3	5	822
A-n39-k6	38	6	800,20	3,71	6	**800,32**	3,69	10	6	831
A-n44-k6	43	6	925,46	1,23	6	**925,59**	1,22	7	6	937
A-n45-k6	44	6	924,41	2,08	6	924,07	2,11	5	6	944
A-n45-k7	44	7	1111,84	2,98	7	**1111,99**	2,97	7	7	1146
A-n46-k7	45	7	898,45	1,70	7	**898,54**	1,69	10	7	914
A-n48-k7	47	7	1044,79	2,63	7	**1044,92**	2,62	9	7	1073
A-n53-k7	52	7	988,42	2,14	7	**988,53**	2,13	14	7	1010
A-n54-k7	53	7	1132,62	2,95	7	**1132,71**	2,94	13	7	1167
A-n55-k9	54	9	1054,22	1,75	9	**1054,41**	1,73	10	9	1073
A-n60-k9	59	9	1322,22	2,35	9	**1322,27**	2,34	16	9	1354
A-n61-k9	60	9	1009,86	2,33	9	**1010,01**	2,32	9	9	1034
A-n62-k8	61	8	1249,88	2,96	8	**1249,90**	2,96	20	8	1288
A-n63-k9	62	9	1578,85	2,30	9	**1578,90**	2,30	16	9	1616
A-n63-k10	62	10	1284,42	2,25	10	**1284,46**	2,25	18	10	1314
A-n64-k9	63	9	1366,70	2,45	9	**1366,80**	2,44	20	9	1401
A-n65-k9	64	9	1146,02	2,38	9	**1146,07**	2,38	12	9	1174
A-n69-k9	68	9	1125,97	2,85	9	**1126,10**	2,84	20	9	1159
A-n80-k10	79	10	1724,33	2,19	10	**1724,50**	2,18	35	10	1763
B-n31-k5	30	5	611,60	8,99	5	**611,61**	8,99	6	5	672
B-n34-k5	33	5	744,10	5,57	5	**744,12**	5,57	6	5	788
B-n35-k5	34	5	825,30	13,58	5	825,30	13,58	6	5	955
B-n38-k6	37	6	712,98	11,43	6	**713,00**	11,43	6	6	805
B-n39-k5	38	5	509,29	7,23	5	**509,32**	7,23	9	5	549
B-n41-k6	40	6	797,71	3,77	6	**797,72**	3,77	7	6	829
B-n43-k6	42	6	698,14	5,91	6	**698,16**	5,91	9	6	742
B-n44-k7	43	7	858,87	5,51	7	**858,91**	5,51	9	7	909

Table B.7: Valid lower bounds before and after tuning for *ng*-route pricing for instances A, B, P, E, M

Instance	n	Before tuning			After tuning			Time(s)	Best known	
		Veh.	Dist.	Gap(%)	Veh.	Dist.	Gap(%)		Veh.	Dist.
B-n45-k5	44	5	684,90	8,80	5	684,08	8,91	14	5	751
B-n45-k6	44	6	652,23	3,80	6	**652,25**	3,80	8	6	678
B-n50-k7	49	7	664,24	10,36	7	**664,26**	10,36	13	7	741
B-n50-k8	49	8	1254,78	4,36	8	1254,47	4,38	14	8	1312
B-n51-k7	50	7	960,82	6,90	7	**960,85**	6,89	13	7	1032
B-n52-k7	51	7	675,74	9,54	7	**675,78**	9,53	17	7	747
B-n56-k7	55	7	628,83	11,06	7	**628,85**	11,05	19	7	707
B-n57-k7	56	7	1125,00	2,43	7	1124,99	2,43	20	7	1153
B-n57-k9	56	9	1498,88	6,20	9	**1499,01**	6,19	16	9	1598
B-n63-k10	62	10	1444,52	3,44	10	**1444,57**	3,44	20	10	1496
B-n64-k9	63	9	805,86	6,40	9	**805,91**	6,40	17	9	861
B-n66-k9	65	9	1250,78	4,96	9	1250,51	4,98	35	9	1316
B-n67-k10	66	10	985,04	4,55	10	**985,10**	4,54	20	10	1032
B-n68-k9	67	9	1179,34	7,28	9	1179,32	7,29	27	9	1272
B-n78-k10	77	10	1157,98	5,16	10	**1158,05**	5,16	34	10	1221
E-n22-k4	21	4	373,49	0,40	4	**373,50**	0,40	120	4	375
E-n23-k3	22	3	555,50	2,37	3	555,50	2,37	113	3	569
E-n30-k3	29	3	478,19	10,45	3	478,19	10,45	200	3	534
E-n33-k4	32	4	802,60	3,88	4	**802,63**	3,88	577	4	835
E-n51-k5	50	5	516,94	0,78	5	**516,99**	0,77	15	5	521
E-n76-k7	75	7	661,80	2,96	7	**661,82**	2,96	56	7	682
E-n76-k8	75	8	717,15	2,43	8	**717,19**	2,42	34	8	735
E-n76-k10	75	10	811,68	2,21	10	**811,70**	2,20	22	10	830
E-n76-k14	75	14	1001,65	1,90	14	**1001,69**	1,89	12	14	1021
E-n101-k8	100	8	785,36	3,64	8	**785,45**	3,63	143	8	815
E-n101-k14	100	14	1044,52	2,11	14	**1044,64**	2,10	43	14	1067
M-n101-k10	100	10	804,56	1,88	10	**804,58**	1,88	25	10	820
M-n121-k7	120	7	1024,60	0,91	7	1024,53	0,92	303	7	1034
M-n151-k12	150	12	989,57	2,51	12	**989,78**	2,48	435	12	1015
M-n200-k16	199	16	1249,67	1,91	16	**1249,74**	1,90	530	16	1274
M-n200-k17	199	17	1249,62	1,99	17	1249,62	1,99	610	17	1275
P-n20-k2	19	2	210,00	2,78	2	210,00	2,78	2	2	216
P-n21-k2	20	2	209,96	0,49	2	**210,01**	0,47	3	2	211
P-n22-k2	21	2	214,50	0,69	2	214,50	0,69	3	2	216
P-n22-k8	21	8	601,25	0,29	8	601,25	0,29	16	8	603
P-n23-k8	22	8	527,89	0,21	8	527,47	0,29	1	8	529

Table B.7: Valid lower bounds before and after tuning for *ng*-route pricing for instances A, B, P, E, M

Instance	n	Before tuning			After tuning			Time(s)	Best known	
		Veh.	Dist.	Gap(%)	Veh.	Dist.	Gap(%)		Veh.	Dist.
P-n40-k5	39	5	448,19	2,14	5	**448,21**	2,14	7	5	458
P-n45-k5	44	5	499,63	2,03	5	**499,68**	2,02	10	5	510
P-n50-k7	49	7	542,01	2,16	7	**542,04**	2,16	10	7	554
P-n50-k8	49	8	614,36	2,64	8	614,09	2,68	4	8	631
P-n50-k10	49	10	686,34	1,39	10	**686,35**	1,39	7	10	696
P-n51-k10	50	10	732,84	1,10	10	**732,88**	1,10	6	10	741
P-n55-k7	54	7	549,41	3,27	7	**549,44**	3,27	14	7	568
P-n55-k10	54	10	674,58	2,80	10	**674,63**	2,79	9	10	694
P-n55-k15	54	15	966,99	2,23	15	965,33	2,39	3	15	989
P-n60-k10	59	10	734,75	1,24	10	**734,78**	1,24	12	10	744
P-n60-k15	59	15	957,97	1,04	15	**958,07**	1,03	7	15	968
P-n65-k10	64	10	780,57	1,44	10	**780,61**	1,44	15	10	792
P-n70-k10	69	10	808,87	2,19	10	**808,99**	2,18	16	10	827
P-n76-k4	75	4	586,05	1,17	4	**586,09**	1,17	203	4	593
P-n76-k5	75	5	614,32	2,02	5	**614,36**	2,02	97	5	627
P-n101-k4	100	4	666,70	2,10	4	**666,76**	2,09	583	4	681

Table B.8: Valid lower bounds before and after tuning for (q, i)-route pricing for instances A, B, P, E, M

Instance	n	Before tuning			After tuning			Time(s)	Best known	
		Veh.	Dist.	Gap(%)	Veh.	Dist.	Gap(%)		Veh.	Dist.
A-n32-k5	31	5	728,13	7,13	5	**728,15**	7,12	1	5	784
A-n33-k5	32	5	643,39	2,66	5	**643,41**	2,66	2	5	661
A-n33-k6	32	6	702,63	5,31	6	702,63	5,31	4	6	742
A-n34-k5	33	5	729,27	6,26	5	**729,31**	6,26	4	5	778
A-n36-k5	35	5	763,27	4,47	5	**763,35**	4,46	4	5	799
A-n37-k5	36	5	638,74	4,52	5	**638,82**	4,51	5	5	669
A-n37-k6	36	6	906,81	4,45	6	**906,90**	4,44	4	6	949
A-n38-k5	37	5	671,00	8,08	5	**671,06**	8,07	5	5	730
A-n39-k5	38	5	794,20	3,38	5	**794,23**	3,38	5	5	822
A-n39-k6	38	6	786,16	5,40	6	**786,23**	5,39	5	6	831
A-n44-k6	43	6	911,68	2,70	6	**911,74**	2,70	6	6	937
A-n45-k6	44	6	895,91	5,09	6	895,64	5,12	3	6	944
A-n45-k7	44	7	1103,08	3,75	7	**1103,10**	3,74	6	7	1146
A-n46-k7	45	7	890,68	2,55	7	**890,74**	2,54	7	7	914
A-n48-k7	47	7	1030,36	3,97	7	**1030,42**	3,97	9	7	1073
A-n53-k7	52	7	973,86	3,58	7	**973,96**	3,57	11	7	1010
A-n54-k7	53	7	1113,51	4,58	7	**1113,55**	4,58	11	7	1167
A-n55-k9	54	9	1020,40	4,90	9	**1020,42**	4,90	9	9	1073
A-n60-k9	59	9	1304,44	3,66	9	**1304,59**	3,65	11	9	1354
A-n61-k9	60	9	996,71	3,61	9	996,71	3,61	7	9	1034
A-n62-k8	61	8	1222,65	5,07	8	**1222,72**	5,07	17	8	1288
A-n63-k9	62	9	1564,69	3,18	9	**1564,74**	3,17	12	9	1616
A-n63-k10	62	10	1266,04	3,65	10	**1266,11**	3,64	13	10	1314
A-n64-k9	63	9	1351,58	3,53	9	**1351,61**	3,53	17	9	1401
A-n65-k9	64	9	1131,64	3,61	9	**1131,79**	3,60	10	9	1174
A-n69-k9	68	9	1111,09	4,13	9	**1111,13**	4,13	17	9	1159
A-n80-k10	79	10	1706,18	3,22	10	**1706,26**	3,22	25	10	1763
B-n31-k5	30	5	525,89	21,74	5	**525,92**	21,74	4	5	672
B-n34-k5	33	5	720,05	8,62	5	**720,07**	8,62	4	5	788
B-n35-k5	34	5	805,58	15,65	5	**805,63**	15,64	5	5	955
B-n38-k6	37	6	671,42	16,59	6	671,37	16,60	8	6	805
B-n39-k5	38	5	476,79	13,15	5	**476,80**	13,15	6	5	549
B-n41-k6	40	6	732,34	11,66	6	732,34	11,66	5	6	829
B-n43-k6	42	6	650,43	12,34	6	650,43	12,34	7	6	742
B-n44-k7	43	7	821,38	9,64	7	821,30	9,65	6	7	909

Table B.8: Valid lower bounds before and after tuning for (q, i)-route pricing for instances A, B, P, E, M

Instance	n	Before tuning			After tuning			Time(s)	Best known	
		Veh.	Dist.	Gap(%)	Veh.	Dist.	Gap(%)		Veh.	Dist.
B-n45-k5	44	5	649,26	13,55	5	**649,29**	13,54	9	5	751
B-n45-k6	44	6	592,45	12,62	6	**592,47**	12,62	5	6	678
B-n50-k7	49	7	631,93	14,72	7	**632,00**	14,71	9	7	741
B-n50-k8	49	8	1205,35	8,13	8	**1205,37**	8,13	10	8	1312
B-n51-k7	50	7	913,01	11,53	7	**913,04**	11,53	8	7	1032
B-n52-k7	51	7	628,69	15,84	7	**628,70**	15,84	10	7	747
B-n56-k7	55	7	599,23	15,24	7	599,23	15,24	13	7	707
B-n57-k7	56	7	1058,40	8,20	7	**1058,45**	8,20	6	7	1153
B-n57-k9	56	9	1428,38	10,61	9	**1428,41**	10,61	10	9	1598
B-n63-k10	62	10	1406,60	5,98	10	**1406,71**	5,97	14	10	1496
B-n64-k9	63	9	766,00	11,03	9	766,00	11,03	13	9	861
B-n66-k9	65	9	1216,50	7,56	9	1216,47	7,56	15	9	1316
B-n67-k10	66	10	967,07	6,29	10	**967,12**	6,29	14	10	1032
B-n68-k9	67	9	1142,65	10,17	9	**1142,74**	10,16	17	9	1272
B-n78-k10	77	10	1117,70	8,46	10	**1117,78**	8,45	23	10	1221
E-n22-k4	21	4	371,22	1,01	4	371,22	1,01	6	4	375
E-n23-k3	22	3	539,41	5,20	3	**539,51**	5,18	11	3	569
E-n30-k3	29	3	464,25	13,06	3	**464,27**	13,06	11	3	534
E-n33-k4	32	4	793,04	5,03	4	**793,13**	5,01	23	4	835
E-n51-k5	50	5	512,54	1,62	5	**512,63**	1,61	12	5	521
E-n76-k7	75	7	660,73	3,12	7	**660,77**	3,11	32	7	682
E-n76-k8	75	8	716,04	2,58	8	**716,11**	2,57	27	8	735
E-n76-k10	75	10	811,22	2,26	10	**811,28**	2,26	19	10	830
E-n76-k14	75	14	999,48	2,11	14	**999,52**	2,10	11	14	1021
E-n101-k8	100	8	782,54	3,98	8	**782,59**	3,98	64	8	815
E-n101-k14	100	14	1042,12	2,33	14	**1042,27**	2,32	34	14	1067
M-n101-k10	100	10	780,63	4,80	10	**780,71**	4,79	51	10	820
M-n121-k7	120	7	1012,60	2,07	7	1012,57	2,07	122	7	1034
M-n151-k12	150	12	986,98	2,76	12	**987,06**	2,75	130	12	1015
M-n200-k16	199	16	1240,22	2,65	16	**1240,36**	2,64	97	16	1274
M-n200-k17	199	17	1240,32	2,72	17	**1240,38**	2,72	202	17	1275
P-n20-k2	19	2	209,51	3,00	2	**209,54**	2,99	2	2	216
P-n21-k2	20	2	207,87	1,48	2	207,87	1,48	2	2	211
P-n22-k2	21	2	212,31	1,71	2	212,31	1,71	2	2	216
P-n22-k8	21	8	601,25	0,29	8	601,25	0,29	3	8	603
P-n23-k8	22	8	528,34	0,12	8	527,33	0,32	1	8	529

Table B.8: Valid lower bounds before and after tuning for (q, i)-route pricing for instances A, B, P, E, M

Instance	n	Before tuning			After tuning			Time(s)	Best known	
		Veh.	Dist.	Gap(%)	Veh.	Dist.	Gap(%)		Veh.	Dist.
P-n40-k5	39	5	444,39	2,97	5	**444,43**	2,96	6	5	458
P-n45-k5	44	5	495,03	2,94	5	**495,06**	2,93	9	5	510
P-n50-k7	49	7	539,76	2,57	7	**539,81**	2,56	9	7	554
P-n50-k8	49	8	612,28	2,97	8	611,95	3,02	3	8	631
P-n50-k10	49	10	683,23	1,83	10	683,23	1,83	6	10	696
P-n51-k10	50	10	729,28	1,58	10	**729,30**	1,58	4	10	741
P-n55-k7	54	7	547,15	3,67	7	**547,19**	3,66	12	7	568
P-n55-k10	54	10	672,68	3,07	10	**672,70**	3,07	9	10	694
P-n55-k15	54	15	961,96	2,73	15	959,34	3,00	2	15	989
P-n60-k10	59	10	731,75	1,65	10	**731,81**	1,64	12	10	744
P-n60-k15	59	15	954,39	1,41	15	**954,42**	1,40	8	15	968
P-n65-k10	64	10	776,86	1,91	10	**776,88**	1,91	13	10	792
P-n70-k10	69	10	807,82	2,32	10	**807,85**	2,32	14	10	827
P-n76-k4	75	4	585,17	1,32	4	**585,19**	1,32	67	4	593
P-n76-k5	75	5	613,08	2,22	5	**613,17**	2,21	47	5	627
P-n101-k4	100	4	662,12	2,77	4	662,09	2,78	150	4	681

Table B.9: Valid lower bounds before and after tuning for (q, i)-route with 2-cycles pricing for instances A, B, P, E, M

Instance	n	Before tuning			After tuning			Time(s)	Best known	
		Veh.	Dist.	Gap(%)	Veh.	Dist.	Gap(%)		Veh.	Dist.
A-n32-k5	31	5	691,08	11,85	5	691,04	11,86	4	5	784
A-n33-k5	32	5	608,55	7,93	5	**608,61**	7,93	4	5	661
A-n33-k6	32	6	671,77	9,46	6	**671,87**	9,45	4	6	742
A-n34-k5	33	5	672,64	13,54	5	**672,65**	13,54	4	5	778
A-n36-k5	35	5	732,06	8,38	5	**732,08**	8,38	6	5	799
A-n37-k5	36	5	615,84	7,95	5	**615,87**	7,94	6	5	669
A-n37-k6	36	6	868,69	8,46	6	**868,84**	8,45	5	6	949
A-n38-k5	37	5	628,86	13,85	5	**628,89**	13,85	5	5	730
A-n39-k5	38	5	770,43	6,27	5	**770,53**	6,26	6	5	822
A-n39-k6	38	6	736,52	11,37	6	**736,61**	11,36	6	6	831
A-n44-k6	43	6	880,82	6,00	6	**880,85**	5,99	7	6	937
A-n45-k6	44	6	847,40	10,23	6	847,37	10,24	4	6	944
A-n45-k7	44	7	1065,73	7,00	7	**1065,74**	7,00	7	7	1146
A-n46-k7	45	7	851,56	6,83	7	**851,59**	6,83	7	7	914
A-n48-k7	47	7	997,22	7,06	7	**997,26**	7,06	8	7	1073
A-n53-k7	52	7	916,05	9,30	7	**916,15**	9,29	12	7	1010
A-n54-k7	53	7	1077,92	7,63	7	**1078,00**	7,63	11	7	1167
A-n55-k9	54	9	975,13	9,12	9	**975,15**	9,12	9	9	1073
A-n60-k9	59	9	1251,41	7,58	9	**1251,50**	7,57	11	9	1354
A-n61-k9	60	9	955,76	7,57	9	**955,80**	7,56	7	9	1034
A-n62-k8	61	8	1182,12	8,22	8	**1182,27**	8,21	15	8	1288
A-n63-k9	62	9	1517,65	6,09	9	**1517,77**	6,08	12	9	1616
A-n63-k10	62	10	1185,44	9,78	10	**1185,83**	9,75	14	10	1314
A-n64-k9	63	9	1277,99	8,78	9	**1278,10**	8,77	15	9	1401
A-n65-k9	64	9	1097,05	6,55	9	**1097,11**	6,55	12	9	1174
A-n69-k9	68	9	1048,50	9,53	9	**1048,60**	9,53	18	9	1159
A-n80-k10	79	10	1652,50	6,27	10	**1652,56**	6,26	24	10	1763
B-n31-k5	30	5	515,71	23,26	5	515,71	23,26	3	5	672
B-n34-k5	33	5	667,09	15,34	5	**667,17**	15,33	4	5	788
B-n35-k5	34	5	779,70	18,36	5	779,70	18,36	5	5	955
B-n38-k6	37	6	654,77	18,66	6	654,77	18,66	5	6	805
B-n39-k5	38	5	452,65	17,55	5	**452,74**	17,53	6	5	549
B-n41-k6	40	6	712,38	14,07	6	712,38	14,07	6	6	829
B-n43-k6	42	6	633,66	14,60	6	**633,68**	14,60	7	6	742
B-n44-k7	43	7	803,56	11,60	7	**803,59**	11,60	6	7	909

Table B.9: Valid lower bounds before and after tuning for (q, i)-route with 2-cycles pricing for instances A, B, P, E, M

Instance	n	Before tuning			After tuning			Time(s)	Best known	
		Veh.	Dist.	Gap(%)	Veh.	Dist.	Gap(%)		Veh.	Dist.
B-n45-k5	44	5	612,17	18,49	5	**612,25**	18,48	9	5	751
B-n45-k6	44	6	572,05	15,63	6	**572,06**	15,63	5	6	678
B-n50-k7	49	7	614,56	17,06	7	**614,63**	17,05	9	7	741
B-n50-k8	49	8	1176,67	10,31	8	**1176,71**	10,31	8	8	1312
B-n51-k7	50	7	898,59	12,93	7	**898,60**	12,93	9	7	1032
B-n52-k7	51	7	602,66	19,32	7	**602,73**	19,31	10	7	747
B-n56-k7	55	7	579,65	18,01	7	**579,68**	18,01	13	7	707
B-n57-k7	56	7	1040,17	9,79	7	**1040,20**	9,78	5	7	1153
B-n57-k9	56	9	1400,26	12,37	9	**1400,32**	12,37	10	9	1598
B-n63-k10	62	10	1374,41	8,13	10	**1374,68**	8,11	14	10	1496
B-n64-k9	63	9	746,70	13,28	9	**746,78**	13,27	13	9	861
B-n66-k9	65	9	1175,21	10,70	9	**1175,22**	10,70	16	9	1316
B-n67-k10	66	10	941,72	8,75	10	**941,96**	8,72	15	10	1032
B-n68-k9	67	9	1120,95	11,88	9	**1121,19**	11,86	17	9	1272
B-n78-k10	77	10	1090,68	10,67	10	**1090,80**	10,66	23	10	1221
E-n22-k4	21	4	343,91	8,29	4	343,91	8,29	6	4	375
E-n23-k3	22	3	492,88	13,38	3	**492,90**	13,37	7	3	569
E-n30-k3	29	3	445,01	16,66	3	**445,07**	16,65	9	3	534
E-n33-k4	32	4	776,62	6,99	4	**776,69**	6,98	19	4	835
E-n51-k5	50	5	494,52	5,08	5	**494,56**	5,07	12	5	521
E-n76-k7	75	7	629,95	7,63	7	**630,00**	7,62	30	7	682
E-n76-k8	75	8	688,61	6,31	8	**688,70**	6,30	25	8	735
E-n76-k10	75	10	784,83	5,44	10	**784,89**	5,43	20	10	830
E-n76-k14	75	14	974,27	4,58	14	**974,44**	4,56	12	14	1021
E-n101-k8	100	8	758,01	6,99	8	757,96	7,00	64	8	815
E-n101-k14	100	14	1023,00	4,12	14	**1023,16**	4,11	34	14	1067
M-n101-k10	100	10	760,29	7,28	10	**760,42**	7,27	46	10	820
M-n121-k7	120	7	977,05	5,51	7	976,88	5,52	110	7	1034
M-n151-k12	150	12	926,30	8,74	12	**926,38**	8,73	131	12	1015
M-n200-k16	199	16	1172,79	7,94	16	**1172,80**	7,94	72	16	1274
M-n200-k17	199	17	1172,68	8,03	17	**1172,70**	8,02	199	17	1275
P-n20-k2	19	2	199,50	7,64	2	**199,51**	7,63	2	2	216
P-n21-k2	20	2	195,86	7,18	2	**195,94**	7,14	2	2	211
P-n22-k2	21	2	199,95	7,43	2	**199,98**	7,42	2	2	216
P-n22-k8	21	8	584,87	3,01	8	**584,91**	3,00	3	8	603
P-n23-k8	22	8	522,20	1,29	8	522,20	1,29	1	8	529

Table B.9: Valid lower bounds before and after tuning for (q, i)-route with 2-cycles pricing for instances A, B, P, E, M

Instance	n	Before tuning			After tuning			Time(s)	Best known	
		Veh.	Dist.	Gap(%)	Veh.	Dist.	Gap(%)		Veh.	Dist.
P-n40-k5	39	5	430,32	6,04	5	**430,42**	6,02	6	5	458
P-n45-k5	44	5	478,52	6,17	5	**478,58**	6,16	8	5	510
P-n50-k7	49	7	527,34	4,81	7	**527,37**	4,81	8	7	554
P-n50-k8	49	8	596,52	5,46	8	596,38	5,49	2	8	631
P-n50-k10	49	10	665,22	4,42	10	**665,25**	4,42	6	10	696
P-n51-k10	50	10	715,29	3,47	10	**715,32**	3,47	5	10	741
P-n55-k7	54	7	532,79	6,20	7	**532,83**	6,19	11	7	568
P-n55-k10	54	10	657,50	5,26	10	**657,52**	5,26	8	10	694
P-n55-k15	54	15	929,20	6,05	15	928,65	6,10	2	15	989
P-n60-k10	59	10	702,43	5,59	10	**702,49**	5,58	11	10	744
P-n60-k15	59	15	939,66	2,93	15	**939,69**	2,92	7	15	968
P-n65-k10	64	10	754,96	4,68	10	**755,00**	4,67	13	10	792
P-n70-k10	69	10	782,36	5,40	10	**782,47**	5,38	15	10	827
P-n76-k4	75	4	547,20	7,72	4	**547,27**	7,71	62	4	593
P-n76-k5	75	5	576,33	8,08	5	**576,46**	8,06	47	5	627
P-n101-k4	100	4	630,16	7,47	4	628,14	7,76	130	4	681

Table B.10: Feasible solution values before and after tuning for *ng*-route pricing for instances A, B, P, E, M

Instance	n	Before tuning			After tuning			Best known	
		Veh.	Dist.	Gap(%)	Veh.	Dist.	Gap(%)	Veh.	Dist.
A-n32-k5	31	5	784	0,00	5	784	0,00	5	784
A-n33-k5	32	5	661	0,00	5	661	0,00	5	661
A-n33-k6	32	6	742	0,00	6	742	0,00	6	742
A-n34-k5	33	5	778	0,00	5	778	0,00	5	778
A-n36-k5	35	5	807	1,00	5	812	1,63	5	799
A-n37-k5	36	5	669	0,00	5	669	0,00	5	669
A-n37-k6	36	6	949	0,00	6	949	0,00	6	949
A-n38-k5	37	5	730	0,00	5	731	0,14	5	730
A-n39-k5	38	5	825	0,36	5	822	0,00	5	822
A-n39-k6	38	6	833	0,24	6	831	0,00	6	831
A-n44-k6	43	6	937	0,00	6	937	0,00	6	937
A-n45-k6	44	6	960	1,69	6	955	1,17	6	944
A-n45-k7	44	7	1146	0,00	7	1146	0,00	7	1146
A-n46-k7	45	7	914	0,00	7	914	0,00	7	914
A-n48-k7	47	7	1081	0,75	7	1086	1,21	7	1073
A-n53-k7	52	7	1017	0,69	7	1015	0,50	7	1010
A-n54-k7	53	7	1172	0,43	7	1167	0,00	7	1167
A-n55-k9	54	9	1073	0,00	9	1073	0,00	9	1073
A-n60-k9	59	9	1358	0,30	9	1359	0,37	9	1354
A-n61-k9	60	9	1044	0,97	9	1035	0,10	9	1034
A-n62-k8	61	8	1300	0,93	8	1299	0,85	8	1288
A-n63-k9	62	9	1636	1,24	9	1632	0,99	9	1616
A-n63-k10	62	10	1320	0,46	10	1318	0,30	10	1314
A-n64-k9	63	9	1425	1,71	9	1417	1,14	9	1401
A-n65-k9	64	9	1178	0,34	9	1178	0,34	9	1174
A-n69-k9	68	9	1164	0,43	9	1164	0,43	9	1159
A-n80-k10	79	10	1783	1,13	10	1786	1,30	10	1763
B-n31-k5	30	5	672	0,00	5	672	0,00	5	672
B-n34-k5	33	5	788	0,00	5	788	0,00	5	788
B-n35-k5	34	5	955	0,00	5	955	0,00	5	955
B-n38-k6	37	6	806	0,12	6	805	0,00	6	805
B-n39-k5	38	5	549	0,00	5	549	0,00	5	549
B-n41-k6	40	6	829	0,00	6	829	0,00	6	829
B-n43-k6	42	6	745	0,40	6	745	0,40	6	742

- datum not available.

Table B.10: Feasible solution values before and after tuning for *ng*-route pricing for instances A, B, P, E, M

Instance	n	Before tuning			After tuning			Best known	
		Veh.	Dist.	Gap(%)	Veh.	Dist.	Gap(%)	Veh.	Dist.
B-n44-k7	43	7	909	0,00	7	909	0,00	7	909
B-n45-k5	44	5	751	0,00	5	751	0,00	5	751
B-n45-k6	44	6	691	1,92	6	683	0,74	6	678
B-n50-k7	49	7	741	0,00	7	741	0,00	7	741
B-n50-k8	49	8	1321	0,69	8	1320	0,61	8	1312
B-n51-k7	50	7	1033	0,10	7	1032	0,00	7	1032
B-n52-k7	51	7	747	0,00	7	748	0,13	7	747
B-n56-k7	55	7	707	0,00	7	707	0,00	7	707
B-n57-k7	56	7	1230	6,68	7	1189	3,12	7	1153
B-n57-k9	56	9	1610	0,75	9	1602	0,25	9	1598
B-n63-k10	62	10	1507	0,74	10	1524	1,87	10	1496
B-n64-k9	63	9	868	0,81	9	868	0,81	9	861
B-n66-k9	65	9	1322	0,46	9	1327	0,84	9	1316
B-n67-k10	66	10	1041	0,87	10	1043	1,07	10	1032
B-n68-k9	67	9	1283	0,86	9	1283	0,86	9	1272
B-n78-k10	77	10	1244	1,88	10	1230	0,74	10	1221
E-n22-k4	21	4	375	0,00	4	375	0,00	4	375
E-n23-k3	22	3	569	0,00	3	569	0,00	3	569
E-n30-k3	29	3	534	0,00	3	534	0,00	3	534
E-n33-k4	32	4	835	0,00	4	835	0,00	4	835
E-n51-k5	50	5	521	0,00	5	521	0,00	5	521
E-n76-k7	75	7	689	1,03	7	684	0,29	7	682
E-n76-k8	75	8	739	0,54	8	737	0,27	8	735
E-n76-k10	75	10	845	1,81	10	851	2,53	10	830
E-n76-k14	75	14	1036	1,47	14	1042	2,06	14	1021
E-n101-k8	100	8	824	1,10	8	822	0,86	8	815
E-n101-k14	100	14	1086	1,78	14	1086	1,78	14	1067
M-n101-k10	100	10	820	0,00	10	820	0,00	10	820
M-n121-k7	120	7	1036	0,19	7	1036	0,19	7	1034
M-n151-k12	150	12	1042	2,66	12	1042	2,66	12	1015
M-n200-k16	199	16	1473	15,62	16	1481	16,25	16	1274
M-n200-k17	199	16	1602	25,65	16	1515	18,82	17	1275
P-n20-k2	19	2	216	0,00	2	216	0,00	2	216
P-n21-k2	20	2	211	0,00	2	211	0,00	2	211
P-n22-k2	21	2	216	0,00	2	216	0,00	2	216

- datum not available.

Table B.10: Feasible solution values before and after tuning for *ng*-route pricing for instances A, B, P, E, M

Instance	n	Before tuning			After tuning			Best known	
		Veh.	Dist.	Gap(%)	Veh.	Dist.	Gap(%)	Veh.	Dist.
P-n22-k8	21	8	603	0,00	8	603	0,00	8	603
P-n23-k8	22	8	529	0,00	8	529	0,00	8	529
P-n40-k5	39	5	458	0,00	5	458	0,00	5	458
P-n45-k5	44	5	510	0,00	5	510	0,00	5	510
P-n50-k7	49	7	554	0,00	7	554	0,00	7	554
P-n50-k8	49	8	646	2,38	8	650	3,01	8	631
P-n50-k10	49	10	697	0,14	10	697	0,14	10	696
P-n51-k10	50	10	743	0,27	10	741	0,00	10	741
P-n55-k7	54	7	572	0,70	7	572	0,70	7	568
P-n55-k10	54	10	698	0,58	10	700	0,86	10	694
P-n55-k15	54	-	-	-	-	-	-	15	989
P-n60-k10	59	10	747	0,40	10	744	0,00	10	744
P-n60-k15	59	15	968	0,00	15	974	0,62	15	968
P-n65-k10	64	10	803	1,39	10	801	1,14	10	792
P-n70-k10	69	10	843	1,93	10	843	1,93	10	827
P-n76-k4	75	4	600	1,18	4	599	1,01	4	593
P-n76-k5	75	5	632	0,80	5	632	0,80	5	627
P-n101-k4	100	4	684	0,44	4	681	0,00	4	681

- datum not available.

Table B.11: Feasible solution values before and after tuning for (q, i)-route pricing for instances A, B, P, E, M

Instance	n	Before tuning			After tuning			Best known	
		Veh.	Dist.	Gap(%)	Veh.	Dist.	Gap(%)	Veh.	Dist.
A-n32-k5	31	5	784	0,00	5	784	0,00	5	784
A-n33-k5	32	5	661	0,00	5	661	0,00	5	661
A-n33-k6	32	6	742	0,00	6	742	0,00	6	742
A-n34-k5	33	5	778	0,00	5	778	0,00	5	778
A-n36-k5	35	5	807	1,00	5	799	0,00	5	799
A-n37-k5	36	5	669	0,00	5	669	0,00	5	669
A-n37-k6	36	6	949	0,00	6	949	0,00	6	949
A-n38-k5	37	5	730	0,00	5	730	0,00	5	730
A-n39-k5	38	5	823	0,12	5	822	0,00	5	822
A-n39-k6	38	6	833	0,24	6	833	0,24	6	831
A-n44-k6	43	6	937	0,00	6	939	0,21	6	937
A-n45-k6	44	6	955	1,17	6	955	1,17	6	944
A-n45-k7	44	7	1146	0,00	7	1146	0,00	7	1146
A-n46-k7	45	7	914	0,00	7	914	0,00	7	914
A-n48-k7	47	7	1073	0,00	7	1073	0,00	7	1073
A-n53-k7	52	7	1010	0,00	7	1017	0,69	7	1010
A-n54-k7	53	7	1174	0,60	7	1174	0,60	7	1167
A-n55-k9	54	9	1073	0,00	9	1073	0,00	9	1073
A-n60-k9	59	9	1358	0,30	9	1358	0,30	9	1354
A-n61-k9	60	9	1041	0,68	9	1037	0,29	9	1034
A-n62-k8	61	8	1302	1,09	8	1310	1,71	8	1288
A-n63-k9	62	9	1636	1,24	9	1629	0,80	9	1616
A-n63-k10	62	10	1326	0,91	10	1319	0,38	10	1314
A-n64-k9	63	9	1416	1,07	9	1425	1,71	9	1401
A-n65-k9	64	9	1178	0,34	9	1184	0,85	9	1174
A-n69-k9	68	9	1167	0,69	9	1167	0,69	9	1159
A-n80-k10	79	10	1804	2,33	10	1801	2,16	10	1763
B-n31-k5	30	5	672	0,00	5	672	0,00	5	672
B-n34-k5	33	5	788	0,00	5	788	0,00	5	788
B-n35-k5	34	5	955	0,00	5	955	0,00	5	955
B-n38-k6	37	6	805	0,00	6	805	0,00	6	805
B-n39-k5	38	5	549	0,00	5	549	0,00	5	549
B-n41-k6	40	6	829	0,00	6	829	0,00	6	829
B-n43-k6	42	6	742	0,00	6	744	0,27	6	742

- datum not available.

Table B.11: Feasible solution values before and after tuning for (q, i)-route pricing for instances A, B, P, E, M

Instance	n	Before tuning			After tuning			Best known	
		Veh.	Dist.	Gap(%)	Veh.	Dist.	Gap(%)	Veh.	Dist.
B-n44-k7	43	7	909	0,00	7	909	0,00	7	909
B-n45-k5	44	5	752	0,13	5	756	0,67	5	751
B-n45-k6	44	6	692	2,06	6	694	2,36	6	678
B-n50-k7	49	7	741	0,00	7	741	0,00	7	741
B-n50-k8	49	8	1322	0,76	8	1324	0,91	8	1312
B-n51-k7	50	7	1035	0,29	7	1032	0,00	7	1032
B-n52-k7	51	7	748	0,13	7	748	0,13	7	747
B-n56-k7	55	7	707	0,00	7	707	0,00	7	707
B-n57-k7	56	7	1202	4,25	7	1210	4,94	7	1153
B-n57-k9	56	9	1606	0,50	9	1603	0,31	9	1598
B-n63-k10	62	10	1512	1,07	10	1510	0,94	10	1496
B-n64-k9	63	9	874	1,51	9	868	0,81	9	861
B-n66-k9	65	9	1320	0,30	9	1320	0,30	9	1316
B-n67-k10	66	10	1049	1,65	10	1058	2,52	10	1032
B-n68-k9	67	9	1285	1,02	9	1289	1,34	9	1272
B-n78-k10	77	10	1227	0,49	10	1227	0,49	10	1221
E-n22-k4	21	4	375	0,00	4	375	0,00	4	375
E-n23-k3	22	3	569	0,00	3	569	0,00	3	569
E-n30-k3	29	3	534	0,00	3	534	0,00	3	534
E-n33-k4	32	4	835	0,00	4	835	0,00	4	835
E-n51-k5	50	5	521	0,00	5	521	0,00	5	521
E-n76-k7	75	7	691	1,32	7	689	1,03	7	682
E-n76-k8	75	8	739	0,54	8	738	0,41	8	735
E-n76-k10	75	10	848	2,17	10	842	1,45	10	830
E-n76-k14	75	14	1043	2,15	14	1045	2,35	14	1021
E-n101-k8	100	8	823	0,98	8	826	1,35	8	815
E-n101-k14	100	14	1085	1,69	14	1086	1,78	14	1067
M-n101-k10	100	10	820	0,00	10	820	0,00	10	820
M-n121-k7	120	7	1043	0,87	7	1044	0,97	7	1034
M-n151-k12	150	12	1044	2,86	12	1046	3,05	12	1015
M-n200-k16	199	16	1533	20,33	16	1505	18,13	16	1274
M-n200-k17	199	16	1698	33,18	16	1571	23,22	17	1275
P-n20-k2	19	2	216	0,00	2	216	0,00	2	216
P-n21-k2	20	2	211	0,00	2	211	0,00	2	211
P-n22-k2	21	2	216	0,00	2	216	0,00	2	216

- datum not available.

Table B.11: Feasible solution values before and after tuning for (q, i)-route pricing for instances A, B, P, E, M

Instance	n	Before tuning			After tuning			Best known	
		Veh.	Dist.	Gap(%)	Veh.	Dist.	Gap(%)	Veh.	Dist.
P-n22-k8	21	8	603	0,00	8	603	0,00	8	603
P-n23-k8	22	8	529	0,00	8	529	0,00	8	529
P-n40-k5	39	5	458	0,00	5	458	0,00	5	458
P-n45-k5	44	5	510	0,00	5	510	0,00	5	510
P-n50-k7	49	7	554	0,00	7	554	0,00	7	554
P-n50-k8	49	8	646	2,38	8	647	2,54	8	631
P-n50-k10	49	10	700	0,57	10	702	0,86	10	696
P-n51-k10	50	10	745	0,54	10	741	0,00	10	741
P-n55-k7	54	7	575	1,23	7	571	0,53	7	568
P-n55-k10	54	10	698	0,58	10	698	0,58	10	694
P-n55-k15	54	-	-	-	-	-	-	15	989
P-n60-k10	59	10	747	0,40	10	745	0,13	10	744
P-n60-k15	59	15	973	0,52	15	974	0,62	15	968
P-n65-k10	64	10	796	0,51	10	801	1,14	10	792
P-n70-k10	69	10	838	1,33	10	843	1,93	10	827
P-n76-k4	75	4	595	0,34	4	599	1,01	4	593
P-n76-k5	75	5	636	1,44	5	629	0,32	5	627
P-n101-k4	100	4	682	0,15	4	684	0,44	4	681

- datum not available.

Table B.12: Feasible solution values before and after tuning for (q, i)-route with 2-cycles pricing for instances A, B, P, E, M

Instance	n	Before tuning			After tuning			Best known	
		Veh.	Dist.	Gap(%)	Veh.	Dist.	Gap(%)	Veh.	Dist.
A-n32-k5	31	5	784	0,00	5	784	0,00	5	784
A-n33-k5	32	5	661	0,00	5	661	0,00	5	661
A-n33-k6	32	6	742	0,00	6	742	0,00	6	742
A-n34-k5	33	5	778	0,00	5	778	0,00	5	778
A-n36-k5	35	5	807	1,00	5	799	0,00	5	799
A-n37-k5	36	5	669	0,00	5	669	0,00	5	669
A-n37-k6	36	6	949	0,00	6	952	0,32	6	949
A-n38-k5	37	5	730	0,00	5	730	0,00	5	730
A-n39-k5	38	5	823	0,12	5	826	0,49	5	822
A-n39-k6	38	6	833	0,24	6	833	0,24	6	831
A-n44-k6	43	6	942	0,53	6	942	0,53	6	937
A-n45-k6	44	6	999	5,83	6	977	3,50	6	944
A-n45-k7	44	7	1148	0,17	7	1146	0,00	7	1146
A-n46-k7	45	7	917	0,33	7	914	0,00	7	914
A-n48-k7	47	7	1084	1,03	7	1073	0,00	7	1073
A-n53-k7	52	7	1020	0,99	7	1017	0,69	7	1010
A-n54-k7	53	7	1172	0,43	7	1174	0,60	7	1167
A-n55-k9	54	9	1073	0,00	9	1074	0,09	9	1073
A-n60-k9	59	9	1359	0,37	9	1354	0,00	9	1354
A-n61-k9	60	9	1057	2,22	9	1063	2,80	9	1034
A-n62-k8	61	8	1314	2,02	8	1314	2,02	8	1288
A-n63-k9	62	9	1644	1,73	9	1631	0,93	9	1616
A-n63-k10	62	10	1323	0,68	10	1328	1,07	10	1314
A-n64-k9	63	9	1434	2,36	9	1428	1,93	9	1401
A-n65-k9	64	9	1186	1,02	9	1193	1,62	9	1174
A-n69-k9	68	9	1169	0,86	9	1167	0,69	9	1159
A-n80-k10	79	10	1813	2,84	10	1797	1,93	10	1763
B-n31-k5	30	5	672	0,00	5	672	0,00	5	672
B-n34-k5	33	5	788	0,00	5	788	0,00	5	788
B-n35-k5	34	5	955	0,00	5	955	0,00	5	955
B-n38-k6	37	6	805	0,00	6	806	0,12	6	805
B-n39-k5	38	5	549	0,00	5	549	0,00	5	549
B-n41-k6	40	6	832	0,36	6	830	0,12	6	829
B-n43-k6	42	6	742	0,00	6	742	0,00	6	742

- datum not available.

Table B.12: Feasible solution values before and after tuning for (q, i)-route with 2-cycles pricing for instances A, B, P, E, M

Instance	n	Before tuning			After tuning			Best known	
		Veh.	Dist.	Gap(%)	Veh.	Dist.	Gap(%)	Veh.	Dist.
B-n44-k7	43	7	909	0,00	7	909	0,00	7	909
B-n45-k5	44	5	751	0,00	5	756	0,67	5	751
B-n45-k6	44	6	691	1,92	6	691	1,92	6	678
B-n50-k7	49	7	741	0,00	7	741	0,00	7	741
B-n50-k8	49	8	1317	0,38	8	1325	0,99	8	1312
B-n51-k7	50	7	1034	0,19	7	1034	0,19	7	1032
B-n52-k7	51	7	749	0,27	7	748	0,13	7	747
B-n56-k7	55	7	711	0,57	7	709	0,28	7	707
B-n57-k7	56	7	1153	0,00	7	1232	6,85	7	1153
B-n57-k9	56	9	1601	0,19	9	1604	0,38	9	1598
B-n63-k10	62	10	1531	2,34	10	1514	1,20	10	1496
B-n64-k9	63	9	880	2,21	9	867	0,70	9	861
B-n66-k9	65	9	1334	1,37	9	1329	0,99	9	1316
B-n67-k10	66	10	1039	0,68	10	1054	2,13	10	1032
B-n68-k9	67	9	1287	1,18	9	1290	1,42	9	1272
B-n78-k10	77	10	1248	2,21	10	1237	1,31	10	1221
E-n22-k4	21	4	375	0,00	4	375	0,00	4	375
E-n23-k3	22	3	569	0,00	3	569	0,00	3	569
E-n30-k3	29	3	537	0,56	3	534	0,00	3	534
E-n33-k4	32	4	835	0,00	4	835	0,00	4	835
E-n51-k5	50	5	526	0,96	5	525	0,77	5	521
E-n76-k7	75	7	689	1,03	7	692	1,47	7	682
E-n76-k8	75	8	740	0,68	8	741	0,82	8	735
E-n76-k10	75	10	855	3,01	10	857	3,25	10	830
E-n76-k14	75	14	1042	2,06	14	1051	2,94	14	1021
E-n101-k8	100	8	824	1,10	8	826	1,35	8	815
E-n101-k14	100	14	1083	1,50	14	1096	2,72	14	1067
M-n101-k10	100	10	825	0,61	10	823	0,37	10	820
M-n121-k7	120	7	1108	7,16	7	1091	5,51	7	1034
M-n151-k12	150	12	1052	3,65	12	1057	4,14	12	1015
M-n200-k16	199	16	1596	25,27	16	1589	24,73	16	1274
M-n200-k17	199	16	1715	34,51	16	1705	33,73	17	1275
P-n20-k2	19	2	216	0,00	2	216	0,00	2	216
P-n21-k2	20	2	211	0,00	2	211	0,00	2	211
P-n22-k2	21	2	216	0,00	2	216	0,00	2	216

- datum not available.

Table B.12: Feasible solution values before and after tuning for (q, i)-route with 2-cycles pricing for instances A, B, P, E, M

Instance	n	Before tuning			After tuning			Best known	
		Veh.	Dist.	Gap(%)	Veh.	Dist.	Gap(%)	Veh.	Dist.
P-n22-k8	21	8	603	0,00	8	603	0,00	8	603
P-n23-k8	22	8	529	0,00	8	529	0,00	8	529
P-n40-k5	39	5	458	0,00	5	458	0,00	5	458
P-n45-k5	44	5	510	0,00	5	510	0,00	5	510
P-n50-k7	49	7	557	0,54	7	556	0,36	7	554
P-n50-k8	49	8	707	12,04	8	694	9,98	8	631
P-n50-k10	49	10	701	0,72	10	702	0,86	10	696
P-n51-k10	50	10	746	0,67	10	745	0,54	10	741
P-n55-k7	54	7	575	1,23	7	571	0,53	7	568
P-n55-k10	54	10	699	0,72	10	700	0,86	10	694
P-n55-k15	54	-	-	-	-	-	-	15	989
P-n60-k10	59	10	745	0,13	10	747	0,40	10	744
P-n60-k15	59	15	977	0,93	15	975	0,72	15	968
P-n65-k10	64	10	799	0,88	10	805	1,64	10	792
P-n70-k10	69	10	853	3,14	10	844	2,06	10	827
P-n76-k4	75	4	596	0,51	4	600	1,18	4	593
P-n76-k5	75	5	637	1,59	5	638	1,75	5	627
P-n101-k4	100	4	683	0,29	4	685	0,59	4	681

- datum not available.

Table B.13: Valid lower bounds before and after tuning for *ng*-route pricing for instances by Solomon (200)

Instance	n	Before tuning			After tuning			Time(s)	Best known	
		Veh.	Dist.	Gap(%)	Veh.	Dist.	Gap(%)		Veh.	Dist.
C101	100	10	828,94	0,00	10	828,94	0,00	13	10	828,94
C102	100	10	821,93	0,85	10	821,93	0,85	43	10	828,94
C103	100	10	815,44	1,52	10	815,44	1,52	147	10	828,06
C104	100	10	803,16	2,62	10	803,16	2,62	531	10	824,78
C105	100	10	822,84	0,74	10	822,84	0,74	18	10	828,94
C106	100	10	828,94	0,00	10	828,94	0,00	19	10	828,94
C107	100	10	820,61	1,00	10	820,61	1,00	24	10	828,94
C108	100	10	820,61	1,00	10	820,61	1,00	38	10	828,94
C109	100	10	820,13	1,06	10	820,13	1,06	123	10	828,94
C201	100	3	591,56	0,00	3	591,56	0,00	54	3	591,56
C202	100	3	591,56	0,00	3	591,56	0,00	241	3	591,56
C203	100	3	591,17	0,00	3	591,17	0,00	1041	3	591,17
C204	100	3	590,60	0,00	3	590,60	0,00	2155	3	590,60
C205	100	3	588,88	0,00	3	588,88	0,00	50	3	588,88
C206	100	3	588,49	0,00	3	586,93	0,27	76	3	588,49
C207	100	3	588,29	0,00	3	588,29	0,00	280	3	588,29
C208	100	3	588,32	0,00	3	588,32	0,00	289	3	588,32
R101	100	19	1609,46	2,50	19	**1609,58**	2,50	3	19	1650,80
R102	100	17	1439,10	3,16	17	1439,10	3,16	8	17	1486,12
R103	100	13	1244,51	3,73	13	**1253,20**	3,05	56	13	1292,68
R104	100	9	984,30	2,28	9	**984,60**	2,25	86	9	1007,31
R105	100	14	1354,69	1,63	14	1354,68	1,63	11	14	1377,11
R106	100	12	1235,97	1,28	12	**1236,01**	1,28	28	12	1252,03
R107	100	10	1077,12	2,49	10	1076,60	2,54	50	10	1104,66
R108	100	9	925,68	3,66	9	925,67	3,66	94	9	960,88
R109	100	11	1148,08	3,90	11	**1148,31**	3,89	17	11	1194,73
R110	100	10	1077,12	3,73	10	1077,04	3,74	37	10	1118,84
R111	100	10	1061,38	3,22	10	1061,24	3,24	41	10	1096,72
R112	100	9	937,42	4,55	9	937,37	4,56	65	9	982,14
R201	100	4	1205,49	3,74	4	**1210,78**	3,32	143	4	1252,37
R202	100	3	1087,46	8,75	3	1059,22	11,12	339	3	1191,70
R203	100	3	828,92	11,77	3	**837,63**	10,84	565	3	939,50
R204	100	2	701,40	15,04	2	**711,99**	13,75	1247	2	825,52
R205	100	3	913,04	8,18	3	**927,99**	6,68	467	3	994,42
R206	100	3	822,64	9,21	3	**825,73**	8,87	783	3	906,14

Table B.13: Valid lower bounds before and after tuning for *ng*-route pricing for instances by Solomon (200)

Instance	n	Before tuning			After tuning			Time(s)	Best known	
		Veh.	Dist.	Gap(%)	Veh.	Dist.	Gap(%)		Veh.	Dist.
R207	100	2	765,39	14,06	2	**776,28**	12,84	1851	2	890,61
R208	100	2	635,33	12,59	2	**659,62**	9,25	2351	2	726,82
R209	100	3	840,45	7,56	3	822,50	9,53	498	3	909,16
R210	100	3	839,14	10,67	3	**857,60**	8,70	774	3	939,37
R211	100	2	788,50	10,98	2	766,53	13,46	1172	2	885,71
RC101	100	14	1588,60	6,38	14	1588,45	6,39	7	14	1696,94
RC102	100	12	1418,00	8,80	12	**1418,20**	8,78	17	12	1554,75
RC103	100	11	1219,47	3,34	11	**1219,59**	3,34	43	11	1261,67
RC104	100	10	1076,62	5,18	10	**1076,65**	5,18	60	10	1135,48
RC105	100	13	1510,68	7,29	13	1508,88	7,40	16	13	1629,44
RC106	100	11	1347,65	5,41	11	**1348,00**	5,39	16	11	1424,73
RC107	100	11	1170,40	4,88	11	**1170,41**	4,88	29	11	1230,48
RC108	100	10	1059,65	7,03	10	**1059,69**	7,03	53	10	1139,82
RC201	100	4	1354,77	3,71	4	**1354,98**	3,69	104	4	1406,94
RC202	100	3	1237,00	9,42	3	1231,66	9,81	405	3	1365,65
RC203	100	3	909,78	13,32	3	**921,59**	12,20	1156	3	1049,62
RC204	100	3	762,17	4,54	3	752,20	5,79	1334	3	798,46
RC205	100	4	1217,35	6,19	4	**1218,23**	6,12	127	4	1297,65
RC206	100	3	1083,63	5,47	3	1083,61	5,47	148	3	1146,32
RC207	100	3	971,94	8,41	3	928,22	12,53	297	3	1061,14
RC208	100	3	776,68	6,21	3	775,96	6,30	420	3	828,14

Table B.14: Valid lower bounds before and after tuning for (t, i)-route pricing for instances by Solomon (200)

Instance	n	Before tuning			After tuning			Time(s)	Best known	
		Veh.	Dist.	Gap(%)	Veh.	Dist.	Gap(%)		Veh.	Dist.
C101	100	10	828,94	0,00	10	828,94	0,00	4	10	828,94
C102	100	10	821,93	0,85	10	821,93	0,85	8	10	828,94
C103	100	10	813,30	1,78	10	813,30	1,78	27	10	828,06
C104	100	10	797,54	3,30	10	797,53	3,30	46	10	824,78
C105	100	10	822,84	0,74	10	822,84	0,74	6	10	828,94
C106	100	10	828,94	0,00	10	828,94	0,00	5	10	828,94
C107	100	10	820,61	1,00	10	820,61	1,00	6	10	828,94
C108	100	10	820,61	1,00	10	820,61	1,00	5	10	828,94
C109	100	10	805,43	2,84	10	**805,50**	2,83	24	10	828,94
C201	100	3	591,56	0,00	3	591,56	0,00	18	3	591,56
C202	100	3	591,56	0,00	3	591,56	0,00	41	3	591,56
C203	100	3	591,17	0,00	3	591,17	0,00	116	3	591,17
C204	100	3	580,51	1,71	3	**590,54**	0,01	502	3	590,60
C205	100	3	585,90	0,51	3	575,69	2,24	96	3	588,88
C206	100	3	584,36	0,70	3	**587,70**	0,13	130	3	588,49
C207	100	3	582,97	0,90	3	575,29	2,21	176	3	588,29
C208	100	3	556,39	5,43	3	**583,61**	0,80	158	3	588,32
R101	100	19	1609,43	2,51	19	**1609,68**	2,49	4	19	1650,80
R102	100	17	1439,10	3,16	17	1439,10	3,16	5	17	1486,12
R103	100	13	1245,50	3,65	13	1240,64	4,03	8	13	1292,68
R104	100	9	982,43	2,47	9	982,04	2,51	11	9	1007,31
R105	100	14	1354,69	1,63	14	**1354,71**	1,63	3	14	1377,11
R106	100	12	1235,84	1,29	12	**1235,95**	1,28	6	12	1252,03
R107	100	10	1075,54	2,64	10	**1075,57**	2,63	8	10	1104,66
R108	100	9	919,07	4,35	9	**919,21**	4,34	12	9	960,88
R109	100	11	1146,15	4,07	11	**1146,64**	4,03	4	11	1194,73
R110	100	10	1067,94	4,55	10	**1068,27**	4,52	7	10	1118,84
R111	100	10	1060,20	3,33	10	**1060,53**	3,30	8	10	1096,72
R112	100	9	930,78	5,23	9	930,78	5,23	7	9	982,14
R201	100	4	1146,72	8,44	4	**1194,55**	4,62	22	4	1252,37
R202	100	3	1014,16	14,90	3	**1073,55**	9,91	67	3	1191,70
R203	100	3	801,75	14,66	3	**829,57**	11,70	316	3	939,50
R204	100	2	656,90	20,43	2	**695,71**	15,72	424	2	825,52
R205	100	3	892,25	10,27	3	**906,99**	8,79	138	3	994,42
R206	100	3	800,33	11,68	3	**814,82**	10,08	230	3	906,14

Table B.14: Valid lower bounds before and after tuning for (t, i)-route pricing for instances by Solomon (200)

Instance	n	Before tuning			After tuning			Time(s)	Best known	
		Veh.	Dist.	Gap(%)	Veh.	Dist.	Gap(%)		Veh.	Dist.
R207	100	2	670,70	24,69	2	**719,48**	19,21	415	2	890,61
R208	100	2	629,17	13,44	2	**639,04**	12,08	430	2	726,82
R209	100	3	791,48	12,94	3	**825,44**	9,21	184	3	909,16
R210	100	3	820,93	12,61	3	**835,63**	11,04	238	3	939,37
R211	100	2	675,77	23,70	2	**750,34**	15,28	268	2	885,71
RC101	100	14	1588,50	6,39	14	**1588,52**	6,39	6	14	1696,94
RC102	100	12	1417,47	8,83	12	**1417,68**	8,82	12	12	1554,75
RC103	100	11	1213,91	3,79	11	**1214,07**	3,77	17	11	1261,67
RC104	100	10	1071,29	5,65	10	**1071,42**	5,64	23	10	1135,48
RC105	100	13	1503,31	7,74	13	1499,28	7,99	9	13	1629,44
RC106	100	11	1332,15	6,50	11	1331,58	6,54	10	11	1424,73
RC107	100	11	1155,53	6,09	11	**1156,08**	6,05	14	11	1230,48
RC108	100	10	1049,15	7,95	10	**1049,28**	7,94	15	10	1139,82
RC201	100	4	1319,94	6,18	4	**1319,99**	6,18	55	4	1406,94
RC202	100	3	1086,45	20,44	3	1079,59	20,95	161	3	1365,65
RC203	100	3	794,78	24,28	3	**822,66**	21,62	240	3	1049,62
RC204	100	3	684,57	14,26	3	683,04	14,46	256	3	798,46
RC205	100	4	1112,18	14,29	4	1101,53	15,11	96	4	1297,65
RC206	100	3	975,86	14,87	3	966,46	15,69	113	3	1146,32
RC207	100	3	846,15	20,26	3	**878,17**	17,24	167	3	1061,14
RC208	100	3	711,34	14,10	3	710,53	14,20	136	3	828,14

Table B.15: Valid lower bounds before and after tuning for (t, i)-route with 2-cycles pricing for instances by Solomon (200)

Instance	n	Before tuning			After tuning			Time(s)	Best known	
		Veh.	Dist.	Gap(%)	Veh.	Dist.	Gap(%)		Veh.	Dist.
C101	100	10	828,94	0,00	10	828,94	0,00	12	10	828,94
C102	100	10	821,93	0,85	10	821,93	0,85	26	10	828,94
C103	100	10	811,87	1,96	10	811,87	1,96	72	10	828,06
C104	100	10	793,60	3,78	10	**793,70**	3,77	145	10	824,78
C105	100	10	822,84	0,74	10	822,84	0,74	16	10	828,94
C106	100	10	828,94	0,00	10	828,94	0,00	15	10	828,94
C107	100	10	820,61	1,00	10	820,61	1,00	19	10	828,94
C108	100	10	801,55	3,30	10	801,49	3,31	43	10	828,94
C109	100	10	778,77	6,05	10	778,77	6,05	57	10	828,94
C201	100	3	591,56	0,00	3	591,56	0,00	43	3	591,56
C202	100	3	591,56	0,00	3	591,56	0,00	107	3	591,56
C203	100	3	571,35	3,35	3	**590,75**	0,07	294	3	591,17
C204	100	3	586,11	0,76	3	581,42	1,55	568	3	590,60
C205	100	3	540,68	8,19	3	**566,97**	3,72	44	3	588,88
C206	100	3	564,81	4,02	3	**570,73**	3,02	58	3	588,49
C207	100	3	519,29	11,73	3	**551,80**	6,20	83	3	588,29
C208	100	3	527,24	10,38	3	**549,47**	6,60	102	3	588,32
R101	100	19	1609,43	2,51	19	**1609,64**	2,49	4	19	1650,80
R102	100	17	1439,10	3,16	17	1439,10	3,16	5	17	1486,12
R103	100	13	1236,91	4,31	13	1233,17	4,60	8	13	1292,68
R104	100	9	968,77	3,83	9	**968,80**	3,82	10	9	1007,31
R105	100	14	1349,53	2,00	14	**1349,84**	1,98	4	14	1377,11
R106	100	12	1222,14	2,39	12	**1222,20**	2,38	6	12	1252,03
R107	100	10	1051,96	4,77	10	1051,91	4,78	9	10	1104,66
R108	100	9	898,53	6,49	9	**898,69**	6,47	10	9	960,88
R109	100	11	1102,04	7,76	11	**1102,24**	7,74	5	11	1194,73
R110	100	10	1036,65	7,35	10	**1036,78**	7,33	7	10	1118,84
R111	100	10	1016,51	7,31	10	**1016,63**	7,30	7	10	1096,72
R112	100	9	891,59	9,22	9	**891,73**	9,21	8	9	982,14
R201	100	4	1061,25	15,26	4	**1112,25**	11,19	28	4	1252,37
R202	100	3	0,00	100,00	3	**940,04**	21,12	79	3	1191,70
R203	100	3	708,58	24,58	3	**742,20**	21,00	92	3	939,50
R204	100	2	596,86	27,70	2	**622,62**	24,58	283	2	825,52
R205	100	3	793,00	20,26	3	**817,73**	17,77	63	3	994,42
R206	100	3	735,65	18,81	3	723,37	20,17	80	3	906,14

Table B.15: Valid lower bounds before and after tuning for (t, i)-route with 2-cycles pricing for instances by Solomon (200)

Instance	n	Before tuning			After tuning			Time(s)	Best known	
		Veh.	Dist.	Gap(%)	Veh.	Dist.	Gap(%)		Veh.	Dist.
R207	100	2	630,31	29,23	2	**647,34**	27,31	130	2	890,61
R208	100	2	598,54	17,65	2	589,10	18,95	148	2	726,82
R209	100	3	712,44	21,64	3	**739,12**	18,70	64	3	909,16
R210	100	3	703,71	25,09	3	**730,65**	22,22	75	3	939,37
R211	100	2	604,34	31,77	2	**650,16**	26,59	110	2	885,71
RC101	100	14	1566,22	7,70	14	**1567,16**	7,65	3	14	1696,94
RC102	100	12	1388,67	10,68	12	**1388,86**	10,67	4	12	1554,75
RC103	100	11	1162,60	7,85	11	**1162,68**	7,85	8	11	1261,67
RC104	100	10	1022,21	9,98	10	**1022,23**	9,97	8	10	1135,48
RC105	100	13	1472,80	9,61	13	1469,98	9,79	4	13	1629,44
RC106	100	11	1247,92	12,41	11	**1248,38**	12,38	5	11	1424,73
RC107	100	11	1093,10	11,16	11	**1093,20**	11,16	5	11	1230,48
RC108	100	10	1013,49	11,08	10	**1013,67**	11,07	7	10	1139,82
RC201	100	4	1113,46	20,86	4	**1132,53**	19,50	27	4	1406,94
RC202	100	3	862,37	36,85	3	**911,05**	33,29	61	3	1365,65
RC203	100	3	660,11	37,11	3	**672,21**	35,96	94	3	1049,62
RC204	100	3	575,69	27,90	3	**585,18**	26,71	112	3	798,46
RC205	100	4	925,46	28,68	4	**947,59**	26,98	31	4	1297,65
RC206	100	3	813,58	29,03	3	**846,18**	26,18	54	3	1146,32
RC207	100	3	732,11	31,01	3	**767,59**	27,66	68	3	1061,14
RC208	100	3	597,04	27,91	3	**624,52**	24,59	86	3	828,14

Table B.16: Valid lower bounds before and after tuning for *ng*-route pricing for instances by Gehring & Homberger

Instance	n	Before tuning			After tuning			Time(s)	Best known	
		Veh.	Dist.	Gap(%)	Veh.	Dist.	Gap(%)		Veh.	Dist.
C1_2_1	200	20	2700,36	0,16	20	2700,36	0,16	27	20	2704,57
C1_2_2	200	18	2757,04	5,51	18	**2758,20**	5,47	336	18	2917,89
C1_2_3	200	18	2582,29	4,62	18	**2582,37**	4,62	1220	18	2707,35
C1_2_4	200	18	2424,44	8,28	18	**2424,49**	8,28	3667	18	2643,31
C1_2_5	200	20	2696,83	0,19	20	2696,83	0,19	45	20	2702,05
C1_2_6	200	20	2696,83	0,16	20	2696,83	0,16	62	20	2701,04
C1_2_7	200	20	2690,57	0,39	20	2690,57	0,39	102	20	2701,04
C1_2_8	200	19	2651,81	4,46	19	**2652,54**	4,43	212	19	2775,48
C1_2_9	200	18	2553,65	4,99	18	2553,64	4,99	503	18	2687,83
C1_2_10	200	18	2492,46	5,71	18	**2492,65**	5,71	1129	18	2643,51
C2_2_1	200	6	1928,79	0,14	6	1928,58	0,15	295	6	1931,44
C2_2_2	200	6	1853,29	0,53	6	**1853,39**	0,52	1893	6	1863,16
C2_2_3	200	6	1762,43	0,71	6	1762,41	0,71	15104	6	1775,08
C2_2_4	200	6	1539,96	9,60	6	**1555,98**	8,66	t.l.	6	1703,43
C2_2_5	200	6	1877,54	0,07	6	1877,54	0,07	501	6	1878,85
C2_2_6	200	6	1844,63	0,68	6	**1844,67**	0,68	1273	6	1857,35
C2_2_7	200	6	1845,63	0,21	6	1845,42	0,22	1506	6	1849,46
C2_2_8	200	6	1804,82	0,86	6	**1805,13**	0,85	1428	6	1820,53
C2_2_9	200	6	1806,31	1,30	6	**1806,40**	1,29	3154	6	1830,05
C2_2_10	200	6	1735,56	3,93	6	**1737,32**	3,83	4964	6	1806,58
R1_2_1	200	20	4719,89	1,34	20	4719,89	1,34	24	20	4784,11
R1_2_2	200	18	3914,25	3,11	18	3858,48	4,49	238	18	4039,86
R1_2_3	200	18	3184,55	5,84	18	3052,79	9,73	1291	18	3381,96
R1_2_4	200	18	2428,91	20,57	18	2384,92	22,01	3182	18	3057,81
R1_2_5	200	18	3957,01	3,67	18	3954,76	3,73	45	18	4107,86
R1_2_6	200	18	3339,20	6,81	18	3230,69	9,84	429	18	3583,14
R1_2_7	200	18	2792,20	11,36	18	2686,61	14,71	1454	18	3150,11
R1_2_8	200	18	2236,08	24,25	18	2230,53	24,44	3517	18	2951,99
R1_2_9	200	18	3550,45	5,59	18	**3552,54**	5,53	112	18	3760,58
R1_2_10	200	18	2854,14	13,54	18	2854,07	13,54	307	18	3301,18
R2_2_1	200	4	0,00	100,00	4	0,00	100,00	682	4	4483,16
R2_2_2	200	4	0,00	100,00	4	0,00	100,00	3880	4	3621,20
R2_2_3	200	4	0,00	100,00	4	0,00	100,00	18540	4	2880,62
R2_2_4	200	4	0,00	100,00	4	0,00	100,00	t.l.	4	1981,29

t.l. time limit exceeded

Table B.16: Valid lower bounds before and after tuning for *ng*-route pricing for instances by Gehring & Homberger

Instance	n	Before tuning			After tuning			Time(s)	Best known	
		Veh.	Dist.	Gap(%)	Veh.	Dist.	Gap(%)		Veh.	Dist.
R2_2_5	200	4	0,00	100,00	4	0,00	100,00	1555	4	3366,79
R2_2_6	200	4	0,00	100,00	4	0,00	100,00	5094	4	2913,03
R2_2_7	200	4	0,00	100,00	4	0,00	100,00	t.l.	4	2451,14
R2_2_8	200	4	0,00	100,00	4	0,00	100,00	t.l.	4	1849,87
R2_2_9	200	4	0,00	100,00	4	0,00	100,00	3311	4	3092,04
R2_2_10	200	4	0,00	100,00	4	0,00	100,00	3502	4	2654,97
RC1_2_1	200	18	3342,19	7,23	18	**3342,37**	7,23	35	18	3602,80
RC1_2_2	200	18	2975,61	8,42	18	2975,56	8,42	349	18	3249,05
RC1_2_3	200	18	2540,23	15,56	18	2517,19	16,33	1336	18	3008,33
RC1_2_4	200	18	1997,27	29,96	18	**2054,50**	27,95	3150	18	2851,68
RC1_2_5	200	18	3063,88	9,11	18	3063,84	9,11	110	18	3371,00
RC1_2_6	200	18	3031,34	8,83	18	**3031,38**	8,83	83	18	3324,80
RC1_2_7	200	18	2815,14	11,73	18	2804,55	12,06	371	18	3189,32
RC1_2_8	200	18	2541,76	17,58	18	**2541,89**	17,58	631	18	3083,93
RC1_2_9	200	18	2548,97	17,27	18	2547,52	17,32	670	18	3081,13
RC1_2_10	200	18	2356,75	21,45	18	2356,70	21,45	1102	18	3000,30
RC2_2_1	200	6	2823,30	8,91	6	**2944,04**	5,02	1088	6	3099,53
RC2_2_2	200	5	2259,17	20,04	5	0,00	100,00	6955	5	2825,24
RC2_2_3	200	4	0,00	100,00	4	0,00	100,00	21597	4	2601,87
RC2_2_4	200	4	0,00	100,00	4	0,00	100,00	t.l.	4	2038,56
RC2_2_5	200	4	0,00	100,00	4	**2513,15**	13,68	2860	4	2911,46
RC2_2_6	200	4	0,00	100,00	4	0,00	100,00	2873	4	2873,12
RC2_2_7	200	4	0,00	100,00	4	0,00	100,00	5280	4	2525,83
RC2_2_8	200	4	0,00	100,00	4	0,00	100,00	8203	4	2292,53
RC2_2_9	200	4	0,00	100,00	4	0,00	100,00	8764	4	2175,04
RC2_2_10	200	4	0,00	100,00	4	0,00	100,00	12812	4	2015,60

t.l. time limit exceeded

Table B.17: Valid lower bounds before and after tuning for (t, i)-route pricing for instances by Gehring & Homberger

Instance	n	Before tuning			After tuning			Time(s)	Best known	
		Veh.	Dist.	Gap(%)	Veh.	Dist.	Gap(%)		Veh.	Dist.
C1_2_1	200	20	2700,35	0,16	20	**2700,36**	0,16	21	20	2704,57
C1_2_2	200	18	2751,58	5,70	18	**2754,70**	5,59	92	18	2917,89
C1_2_3	200	18	2572,50	4,98	18	**2572,63**	4,98	249	18	2707,35
C1_2_4	200	18	2380,53	9,94	18	**2398,21**	9,27	1017	18	2643,31
C1_2_5	200	20	2696,83	0,19	20	2696,83	0,19	69	20	2702,05
C1_2_6	200	20	2696,83	0,16	20	2696,83	0,16	73	20	2701,04
C1_2_7	200	20	2690,57	0,39	20	2690,57	0,39	106	20	2701,04
C1_2_8	200	19	2651,90	4,45	19	**2652,34**	4,44	117	19	2775,48
C1_2_9	200	18	2488,57	7,41	18	**2488,66**	7,41	239	18	2687,83
C1_2_10	200	18	2413,74	8,69	18	2413,08	8,72	239	18	2643,51
C2_2_1	200	6	1928,79	0,14	6	**1929,33**	0,11	57	6	1931,44
C2_2_2	200	6	1852,64	0,56	6	1851,48	0,63	701	6	1863,16
C2_2_3	200	6	1742,23	1,85	6	1738,44	2,06	1689	6	1775,08
C2_2_4	200	6	1541,59	9,50	6	**1597,16**	6,24	2440	6	1703,43
C2_2_5	200	6	1843,64	1,87	6	1831,89	2,50	481	6	1878,85
C2_2_6	200	6	1762,72	5,09	6	1741,12	6,26	471	6	1857,35
C2_2_7	200	6	1807,53	2,27	6	1806,91	2,30	574	6	1849,46
C2_2_8	200	6	1718,14	5,62	6	1710,87	6,02	947	6	1820,53
C2_2_9	200	6	0,00	100,00	6	**1715,33**	6,27	861	6	1830,05
C2_2_10	200	6	1603,85	11,22	6	**1633,40**	9,59	1356	6	1806,58
R1_2_1	200	20	4719,87	1,34	20	**4719,89**	1,34	45	20	4784,11
R1_2_2	200	18	3888,42	3,75	18	3856,77	4,53	221	18	4039,86
R1_2_3	200	18	3117,78	7,81	18	3101,25	8,30	572	18	3381,96
R1_2_4	200	18	2415,29	21,01	18	2393,90	21,71	695	18	3057,81
R1_2_5	200	18	3951,29	3,81	18	3948,81	3,87	72	18	4107,86
R1_2_6	200	18	3186,15	11,08	18	**3235,85**	9,69	293	18	3583,14
R1_2_7	200	18	2657,55	15,64	18	**2728,47**	13,38	226	18	3150,11
R1_2_8	200	18	2220,51	24,78	18	2207,07	25,23	265	18	2951,99
R1_2_9	200	18	3547,45	5,67	18	3545,91	5,71	38	18	3760,58
R1_2_10	200	18	2803,32	15,08	18	2798,70	15,22	77	18	3301,18
R2_2_1	200	4	3921,44	12,53	4	**3990,68**	10,99	420	4	4483,16
R2_2_2	200	4	0,00	100,00	4	0,00	100,00	1351	4	3621,20
R2_2_3	200	4	0,00	100,00	4	0,00	100,00	1958	4	2880,62
R2_2_4	200	4	0,00	100,00	4	0,00	100,00	3017	4	1981,29
R2_2_5	200	4	0,00	100,00	4	0,00	100,00	388	4	3366,79

Table B.17: Valid lower bounds before and after tuning for (t, i)-route pricing for instances by Gehring & Homberger

Instance	n	Before tuning			After tuning			Time(s)	Best known	
		Veh.	Dist.	Gap(%)	Veh.	Dist.	Gap(%)		Veh.	Dist.
R2_2_6	200	4	0,00	100,00	4	0,00	100,00	1616	4	2913,03
R2_2_7	200	4	0,00	100,00	4	0,00	100,00	2047	4	2451,14
R2_2_8	200	4	0,00	100,00	4	**1466,00**	20,75	3034	4	1849,87
R2_2_9	200	4	0,00	100,00	4	0,00	100,00	1195	4	3092,04
R2_2_10	200	4	0,00	100,00	4	0,00	100,00	1018	4	2654,97
RC1_2_1	200	18	3342,14	7,23	18	**3342,22**	7,23	46	18	3602,80
RC1_2_2	200	18	2972,86	8,50	18	2972,51	8,51	192	18	3249,05
RC1_2_3	200	18	2526,95	16,00	18	2483,64	17,44	382	18	3008,33
RC1_2_4	200	18	1873,05	34,32	18	**2004,89**	29,69	390	18	2851,68
RC1_2_5	200	18	3039,17	9,84	18	3037,25	9,90	90	18	3371,00
RC1_2_6	200	18	3006,55	9,57	18	3005,94	9,59	85	18	3324,80
RC1_2_7	200	18	2770,45	13,13	18	**2778,12**	12,89	77	18	3189,32
RC1_2_8	200	18	2401,98	22,11	18	**2499,44**	18,95	69	18	3083,93
RC1_2_9	200	18	2495,45	19,01	18	2490,06	19,18	65	18	3081,13
RC1_2_10	200	18	2315,30	22,83	18	**2325,40**	22,49	199	18	3000,30
RC2_2_1	200	6	2616,17	15,59	6	2608,56	15,84	428	6	3099,53
RC2_2_2	200	5	0,00	100,00	5	0,00	100,00	1295	5	2825,24
RC2_2_3	200	4	0,00	100,00	4	0,00	100,00	2382	4	2601,87
RC2_2_4	200	4	0,00	100,00	4	0,00	100,00	3241	4	2038,56
RC2_2_5	200	4	0,00	100,00	4	0,00	100,00	883	4	2911,46
RC2_2_6	200	4	0,00	100,00	4	0,00	100,00	834	4	2873,12
RC2_2_7	200	4	0,00	100,00	4	0,00	100,00	764	4	2525,83
RC2_2_8	200	4	0,00	100,00	4	0,00	100,00	1492	4	2292,53
RC2_2_9	200	4	0,00	100,00	4	0,00	100,00	1489	4	2175,04
RC2_2_10	200	4	0,00	100,00	4	0,00	100,00	1955	4	2015,60

Table B.18: Valid lower bounds before and after tuning for (t, i)-route with 2-cycles pricing for instances by Gehring & Homberger

Instance	n	Before tuning			After tuning			Time(s)	Best known	
		Veh.	Dist.	Gap(%)	Veh.	Dist.	Gap(%)		Veh.	Dist.
C1_2_1	200	20	2700,36	0,16	20	2700,36	0,16	17	20	2704,57
C1_2_2	200	18	2753,35	5,64	18	2747,11	5,85	142	18	2917,89
C1_2_3	200	18	2566,93	5,19	18	**2566,94**	5,19	323	18	2707,35
C1_2_4	200	18	2377,68	10,05	18	2365,16	10,52	332	18	2643,31
C1_2_5	200	20	2696,83	0,19	20	2696,83	0,19	26	20	2702,05
C1_2_6	200	20	2696,83	0,16	20	2696,83	0,16	30	20	2701,04
C1_2_7	200	20	2690,57	0,39	20	2690,57	0,39	37	20	2701,04
C1_2_8	200	19	2564,41	7,60	19	2563,24	7,65	76	19	2775,48
C1_2_9	200	18	2438,03	9,29	18	2437,81	9,30	135	18	2687,83
C1_2_10	200	18	2341,25	11,43	18	**2362,25**	10,64	258	18	2643,51
C2_2_1	200	6	1928,79	0,14	6	1928,33	0,16	46	6	1931,44
C2_2_2	200	6	1849,73	0,72	6	1847,46	0,84	435	6	1863,16
C2_2_3	200	6	0,00	100,00	6	**1679,76**	5,37	1534	6	1775,08
C2_2_4	200	6	1492,67	12,37	6	1436,55	15,67	2072	6	1703,43
C2_2_5	200	6	0,00	100,00	6	**1716,19**	8,66	351	6	1878,85
C2_2_6	200	6	0,00	100,00	6	**1611,93**	13,21	412	6	1857,35
C2_2_7	200	6	0,00	100,00	6	0,00	100,00	554	6	1849,46
C2_2_8	200	6	0,00	100,00	6	**1564,73**	14,05	518	6	1820,53
C2_2_9	200	6	0,00	100,00	6	**1522,66**	16,80	770	6	1830,05
C2_2_10	200	6	0,00	100,00	6	0,00	100,00	796	6	1806,58
R1_2_1	200	20	4719,89	1,34	20	4719,89	1,34	16	20	4784,11
R1_2_2	200	18	3886,48	3,80	18	**3913,12**	3,14	86	18	4039,86
R1_2_3	200	18	2951,65	12,72	18	**3037,56**	10,18	227	18	3381,96
R1_2_4	200	18	2301,41	24,74	18	**2342,11**	23,41	328	18	3057,81
R1_2_5	200	18	3921,13	4,55	18	3920,25	4,57	26	18	4107,86
R1_2_6	200	18	3156,30	11,91	18	3114,83	13,07	107	18	3583,14
R1_2_7	200	18	2476,91	21,37	18	**2560,86**	18,71	197	18	3150,11
R1_2_8	200	18	0,00	100,00	18	**1963,96**	33,47	306	18	2951,99
R1_2_9	200	18	3390,72	9,84	18	**3404,80**	9,46	46	18	3760,58
R1_2_10	200	18	2518,01	23,72	18	**2558,21**	22,51	78	18	3301,18
R2_2_1	200	4	0,00	100,00	4	0,00	100,00	202	4	4483,16
R2_2_2	200	4	0,00	100,00	4	0,00	100,00	780	4	3621,20
R2_2_3	200	4	0,00	100,00	4	0,00	100,00	1350	4	2880,62
R2_2_4	200	4	0,00	100,00	4	0,00	100,00	2218	4	1981,29
R2_2_5	200	4	0,00	100,00	4	0,00	100,00	347	4	3366,79

Table B.18: Valid lower bounds before and after tuning for (t, i)-route with 2-cycles pricing for instances by Gehring & Homberger

Instance	n	Before tuning			After tuning			Time(s)	Best known	
		Veh.	Dist.	Gap(%)	Veh.	Dist.	Gap(%)		Veh.	Dist.
R2_2_6	200	4	0,00	100,00	4	0,00	100,00	896	4	2913,03
R2_2_7	200	4	0,00	100,00	4	**703,67**	71,29	1395	4	2451,14
R2_2_8	200	4	0,00	100,00	4	**488,79**	73,58	2116	4	1849,87
R2_2_9	200	4	0,00	100,00	4	0,00	100,00	693	4	3092,04
R2_2_10	200	4	0,00	100,00	4	0,00	100,00	613	4	2654,97
RC1_2_1	200	18	3286,44	8,78	18	3286,44	8,78	55	18	3602,80
RC1_2_2	200	18	2830,59	12,88	18	2778,89	14,47	209	18	3249,05
RC1_2_3	200	18	2267,75	24,62	18	**2340,95**	22,18	290	18	3008,33
RC1_2_4	200	18	1699,40	40,41	18	**1834,93**	35,65	339	18	2851,68
RC1_2_5	200	18	2949,48	12,50	18	**2952,68**	12,41	39	18	3371,00
RC1_2_6	200	18	2882,76	13,30	18	**2894,00**	12,96	98	18	3324,80
RC1_2_7	200	18	2651,16	16,87	18	2624,51	17,71	149	18	3189,32
RC1_2_8	200	18	2367,48	23,23	18	2354,85	23,64	134	18	3083,93
RC1_2_9	200	18	2328,00	24,44	18	**2330,07**	24,38	70	18	3081,13
RC1_2_10	200	18	2165,47	27,82	18	2131,43	28,96	76	18	3000,30
RC2_2_1	200	6	0,00	100,00	6	0,00	100,00	195	6	3099,53
RC2_2_2	200	5	0,00	100,00	5	0,00	100,00	758	5	2825,24
RC2_2_3	200	4	0,00	100,00	4	0,00	100,00	1944	4	2601,87
RC2_2_4	200	4	0,00	100,00	4	0,00	100,00	1818	4	2038,56
RC2_2_5	200	4	0,00	100,00	4	0,00	100,00	609	4	2911,46
RC2_2_6	200	4	0,00	100,00	4	0,00	100,00	345	4	2873,12
RC2_2_7	200	4	0,00	100,00	4	0,00	100,00	476	4	2525,83
RC2_2_8	200	4	0,00	100,00	4	0,00	100,00	856	4	2292,53
RC2_2_9	200	4	0,00	100,00	4	0,00	100,00	877	4	2175,04
RC2_2_10	200	4	0,00	100,00	4	0,00	100,00	984	4	2015,60

Table B.19: Feasible solution values before and after tuning for *ng*-route pricing for instances by Solomon (200)

Instance	n	Before tuning			After tuning			Best known	
		Veh.	Dist.	Gap(%)	Veh.	Dist.	Gap(%)	Veh.	Dist.
C101	100	10	828,94	0,00	10	828,94	0,00	10	828,94
C102	100	10	828,94	0,00	10	828,94	0,00	10	828,94
C103	100	10	828,06	0,00	10	828,06	0,00	10	828,06
C104	100	10	824,78	0,00	10	824,78	0,00	10	824,78
C105	100	10	828,94	0,00	10	828,94	0,00	10	828,94
C106	100	10	828,94	0,00	10	828,94	0,00	10	828,94
C107	100	10	828,94	0,00	10	828,94	0,00	10	828,94
C108	100	10	828,94	0,00	10	828,94	0,00	10	828,94
C109	100	10	828,94	0,00	10	828,94	0,00	10	828,94
C201	100	3	591,56	0,00	3	591,56	0,00	3	591,56
C202	100	3	591,56	0,00	3	591,56	0,00	3	591,56
C203	100	3	591,17	0,00	3	591,17	0,00	3	591,17
C204	100	3	590,60	0,00	3	590,60	0,00	3	590,60
C205	100	3	588,88	0,00	3	588,88	0,00	3	588,88
C206	100	3	588,49	0,00	3	588,49	0,00	3	588,49
C207	100	3	588,29	0,00	3	588,29	0,00	3	588,29
C208	100	3	588,32	0,00	3	588,32	0,00	3	588,32
R101	100	19	1673,31	1,36	19	1673,31	1,36	19	1650,80
R102	100	17	1528,76	2,87	17	1528,76	2,87	17	1486,12
R103	100	-	-	-	-	-	-	13	1292,68
R104	100	-	-	-	-	-	-	9	1007,31
R105	100	14	1445,47	4,96	14	1445,47	4,96	14	1377,11
R106	100	-	-	-	12	1273,91	1,75	12	1252,03
R107	100	-	-	-	-	-	-	10	1104,66
R108	100	-	-	-	-	-	-	9	960,88
R109	100	-	-	-	-	-	-	11	1194,73
R110	100	-	-	-	-	-	-	10	1118,84
R111	100	-	-	-	-	-	-	10	1096,72
R112	100	-	-	-	-	-	-	9	982,14
R201	100	4	1268,29	1,27	4	1266,69	1,14	4	1252,37
R202	100	-	-	-	-	-	-	3	1191,70
R203	100	3	960,26	2,21	3	982,12	4,54	3	939,50
R204	100	-	-	-	-	-	-	2	825,52
R205	100	3	1019,21	2,49	3	1038,72	4,45	3	994,42

- datum not available.

Table B.19: Feasible solution values before and after tuning for *ng*-route pricing for instances by Solomon (200)

Instance	n	Before tuning			After tuning			Best known	
		Veh.	Dist.	Gap(%)	Veh.	Dist.	Gap(%)	Veh.	Dist.
R206	100	3	925,58	2,15	3	926,33	2,23	3	906,14
R207	100	-	-	-	-	-	-	2	890,61
R208	100	2	732,19	0,74	2	932,79	28,34	2	726,82
R209	100	3	962,99	5,92	3	926,86	1,95	3	909,16
R210	100	3	994,63	5,88	3	998,41	6,29	3	939,37
R211	100	-	-	-	-	-	-	2	885,71
RC101	100	-	-	-	-	-	-	14	1696,94
RC102	100	-	-	-	-	-	-	12	1554,75
RC103	100	11	1434,11	13,67	11	1387,63	9,98	11	1261,67
RC104	100	-	-	-	10	1184,65	4,33	10	1135,48
RC105	100	-	-	-	-	-	-	13	1629,44
RC106	100	-	-	-	-	-	-	11	1424,73
RC107	100	11	1455,97	18,33	11	1455,97	18,33	11	1230,48
RC108	100	-	-	-	-	-	-	10	1139,82
RC201	100	4	1436,41	2,09	4	1477,29	5,00	4	1406,94
RC202	100	3	1727,05	26,46	3	1727,05	26,46	3	1365,65
RC203	100	3	1133,07	7,95	3	1146,63	9,24	3	1049,62
RC204	100	3	807,95	1,19	3	809,74	1,41	3	798,46
RC205	100	4	1370,11	5,58	4	1353,49	4,30	4	1297,65
RC206	100	3	1511,94	31,90	3	1453,14	26,77	3	1146,32
RC207	100	3	1398,19	31,76	3	1362,08	28,36	3	1061,14
RC208	100	3	845,80	2,13	3	850,70	2,72	3	828,14

- datum not available.

Table B.20: Feasible solution values before and after tuning for (t, i)-route pricing for instances by Solomon (200)

Instance	n	Before tuning			After tuning			Best known	
		Veh.	Dist.	Gap(%)	Veh.	Dist.	Gap(%)	Veh.	Dist.
C101	100	10	828,94	0,00	10	828,94	0,00	10	828,94
C102	100	10	828,94	0,00	10	828,94	0,00	10	828,94
C103	100	10	828,06	0,00	10	828,06	0,00	10	828,06
C104	100	10	824,78	0,00	10	824,78	0,00	10	824,78
C105	100	10	828,94	0,00	10	828,94	0,00	10	828,94
C106	100	10	828,94	0,00	10	828,94	0,00	10	828,94
C107	100	10	828,94	0,00	10	828,94	0,00	10	828,94
C108	100	10	828,94	0,00	10	828,94	0,00	10	828,94
C109	100	10	828,94	0,00	10	828,94	0,00	10	828,94
C201	100	3	591,56	0,00	3	591,56	0,00	3	591,56
C202	100	3	591,56	0,00	3	591,56	0,00	3	591,56
C203	100	3	591,17	0,00	3	591,17	0,00	3	591,17
C204	100	3	590,60	0,00	3	590,60	0,00	3	590,60
C205	100	3	588,88	0,00	3	588,88	0,00	3	588,88
C206	100	3	588,49	0,00	3	588,50	0,00	3	588,49
C207	100	3	588,29	0,00	3	588,29	0,00	3	588,29
C208	100	3	588,32	0,00	3	588,32	0,00	3	588,32
R101	100	19	1673,31	1,36	19	1673,31	1,36	19	1650,80
R102	100	17	1528,76	2,87	17	1528,76	2,87	17	1486,12
R103	100	-	-	-	-	-	-	13	1292,68
R104	100	-	-	-	-	-	-	9	1007,31
R105	100	14	1445,47	4,96	14	1445,47	4,96	14	1377,11
R106	100	12	1274,12	1,76	-	-	-	12	1252,03
R107	100	-	-	-	-	-	-	10	1104,66
R108	100	-	-	-	-	-	-	9	960,88
R109	100	-	-	-	-	-	-	11	1194,73
R110	100	-	-	-	-	-	-	10	1118,84
R111	100	-	-	-	-	-	-	10	1096,72
R112	100	-	-	-	-	-	-	9	982,14
R201	100	4	1278,94	2,12	4	1295,96	3,48	4	1252,37
R202	100	-	-	-	-	-	-	3	1191,70
R203	100	3	967,68	3,00	3	1031,03	9,74	3	939,50
R204	100	-	-	-	-	-	-	2	825,52
R205	100	3	1034,63	4,04	3	1086,55	9,26	3	994,42

- datum not available.

Table B.20: Feasible solution values before and after tuning for (t, i)-route pricing for instances by Solomon (200)

Instance	n	Before tuning			After tuning			Best known	
		Veh.	Dist.	Gap(%)	Veh.	Dist.	Gap(%)	Veh.	Dist.
R206	100	3	942,26	3,99	3	936,71	3,37	3	906,14
R207	100	-	-	-	-	-	-	2	890,61
R208	100	2	791,64	8,92	2	932,79	28,34	2	726,82
R209	100	3	974,22	7,16	3	920,07	1,20	3	909,16
R210	100	3	989,56	5,34	3	1008,48	7,36	3	939,37
R211	100	-	-	-	-	-	-	2	885,71
RC101	100	-	-	-	-	-	-	14	1696,94
RC102	100	-	-	-	-	-	-	12	1554,75
RC103	100	11	1434,11	13,67	11	1434,11	13,67	11	1261,67
RC104	100	-	-	-	-	-	-	10	1135,48
RC105	100	-	-	-	-	-	-	13	1629,44
RC106	100	-	-	-	-	-	-	11	1424,73
RC107	100	11	1455,97	18,33	11	1455,97	18,33	11	1230,48
RC108	100	-	-	-	-	-	-	10	1139,82
RC201	100	4	1569,18	11,53	4	1499,97	6,61	4	1406,94
RC202	100	3	1727,05	26,46	3	1727,05	26,46	3	1365,65
RC203	100	3	1223,47	16,56	3	1223,47	16,56	3	1049,62
RC204	100	3	815,01	2,07	3	806,21	0,97	3	798,46
RC205	100	4	1670,51	28,73	4	1374,37	5,91	4	1297,65
RC206	100	3	1511,94	31,90	3	1511,94	31,90	3	1146,32
RC207	100	3	1398,19	31,76	3	1398,19	31,76	3	1061,14
RC208	100	3	847,56	2,35	3	863,83	4,31	3	828,14

- datum not available.

Table B.21: Feasible solution values before and after tuning for (t, i)-route with 2-cycles pricing for instances by Solomon (200)

Instance	n	Before tuning			After tuning			Best known	
		Veh.	Dist.	Gap(%)	Veh.	Dist.	Gap(%)	Veh.	Dist.
C101	100	10	828,94	0,00	10	828,94	0,00	10	828,94
C102	100	10	828,94	0,00	10	828,94	0,00	10	828,94
C103	100	10	828,06	0,00	10	828,06	0,00	10	828,06
C104	100	10	824,78	0,00	10	826,37	0,19	10	824,78
C105	100	10	828,94	0,00	10	828,94	0,00	10	828,94
C106	100	10	828,94	0,00	10	828,94	0,00	10	828,94
C107	100	10	828,94	0,00	10	828,94	0,00	10	828,94
C108	100	10	828,94	0,00	10	828,94	0,00	10	828,94
C109	100	10	828,94	0,00	10	828,94	0,00	10	828,94
C201	100	3	591,56	0,00	3	591,56	0,00	3	591,56
C202	100	3	591,56	0,00	3	591,56	0,00	3	591,56
C203	100	3	591,17	0,00	3	591,17	0,00	3	591,17
C204	100	3	590,60	0,00	3	596,55	1,01	3	590,60
C205	100	3	588,88	0,00	3	588,88	0,00	3	588,88
C206	100	3	588,49	0,00	3	588,49	0,00	3	588,49
C207	100	3	588,29	0,00	3	588,29	0,00	3	588,29
C208	100	3	588,32	0,00	3	588,32	0,00	3	588,32
R101	100	19	1673,31	1,36	19	1673,31	1,36	19	1650,80
R102	100	17	1528,76	2,87	17	1528,76	2,87	17	1486,12
R103	100	-	-	-	-	-	-	13	1292,68
R104	100	-	-	-	-	-	-	9	1007,31
R105	100	14	1445,47	4,96	14	1445,47	4,96	14	1377,11
R106	100	12	1289,04	2,96	-	-	-	12	1252,03
R107	100	-	-	-	-	-	-	10	1104,66
R108	100	-	-	-	-	-	-	9	960,88
R109	100	-	-	-	-	-	-	11	1194,73
R110	100	-	-	-	-	-	-	10	1118,84
R111	100	-	-	-	-	-	-	10	1096,72
R112	100	-	-	-	-	-	-	9	982,14
R201	100	4	1303,97	4,12	4	1302,65	4,01	4	1252,37
R202	100	-	-	-	-	-	-	3	1191,70
R203	100	3	983,94	4,73	3	985,68	4,92	3	939,50
R204	100	-	-	-	-	-	-	2	825,52
R205	100	3	1133,52	13,99	3	1095,87	10,20	3	994,42

- datum not available.

Table B.21: Feasible solution values before and after tuning for (t, i)-route with 2-cycles pricing for instances by Solomon (200)

Instance	n	Before tuning			After tuning			Best known	
		Veh.	Dist.	Gap(%)	Veh.	Dist.	Gap(%)	Veh.	Dist.
R206	100	3	967,01	6,72	3	976,22	7,73	3	906,14
R207	100	-	-	-	-	-	-	2	890,61
R208	100	2	932,79	28,34	2	921,87	26,84	2	726,82
R209	100	3	1017,46	11,91	3	957,17	5,28	3	909,16
R210	100	3	1041,37	10,86	3	1047,66	11,53	3	939,37
R211	100	-	-	-	-	-	-	2	885,71
RC101	100	-	-	-	-	-	-	14	1696,94
RC102	100	-	-	-	-	-	-	12	1554,75
RC103	100	11	1434,11	13,67	11	1434,11	13,67	11	1261,67
RC104	100	-	-	-	-	-	-	10	1135,48
RC105	100	-	-	-	-	-	-	13	1629,44
RC106	100	-	-	-	-	-	-	11	1424,73
RC107	100	11	1455,97	18,33	11	1455,97	18,33	11	1230,48
RC108	100	-	-	-	-	-	-	10	1139,82
RC201	100	4	1596,02	13,44	4	1596,02	13,44	4	1406,94
RC202	100	3	1727,05	26,46	3	1663,57	21,82	3	1365,65
RC203	100	3	1223,47	16,56	3	1220,18	16,25	3	1049,62
RC204	100	3	840,45	5,26	3	824,68	3,28	3	798,46
RC205	100	4	1660,84	27,99	4	1660,84	27,99	4	1297,65
RC206	100	3	1511,94	31,90	3	1453,14	26,77	3	1146,32
RC207	100	3	1398,19	31,76	3	1362,08	28,36	3	1061,14
RC208	100	3	849,04	2,52	3	862,31	4,13	3	828,14

- datum not available.

Table B.22: Feasible solution values before and after tuning for *ng*-route pricing for instances by Gehring & Homberger

Instance	n	Before tuning			After tuning			Best known	
		Veh.	Dist.	Gap(%)	Veh.	Dist.	Gap(%)	Veh.	Dist.
C1_2_1	200	20	2704,57	0,00	20	2704,57	0,00	20	2704,57
C1_2_2	200	18	4047,23	38,70	18	4019,84	37,77	18	2917,89
C1_2_3	200	18	2908,57	7,43	18	2950,09	8,97	18	2707,35
C1_2_4	200	18	2759,76	4,41	18	2777,75	5,09	18	2643,31
C1_2_5	200	20	2987,55	10,57	20	2987,55	10,57	20	2702,05
C1_2_6	200	20	2704,36	0,12	20	2701,04	0,00	20	2701,04
C1_2_7	200	20	2701,35	0,01	20	3197,42	18,38	20	2701,04
C1_2_8	200	19	3117,98	12,34	19	3117,98	12,34	19	2775,48
C1_2_9	200	18	3392,24	26,21	18	3392,24	26,21	18	2687,83
C1_2_10	200	18	2939,84	11,21	18	2930,36	10,85	18	2643,51
C2_2_1	200	6	1931,44	0,00	6	1931,44	0,00	6	1931,44
C2_2_2	200	6	1863,16	0,00	6	1863,16	0,00	6	1863,16
C2_2_3	200	6	1785,11	0,57	6	1782,22	0,40	6	1775,08
C2_2_4	200	6	1776,38	4,28	6	1775,67	4,24	6	1703,43
C2_2_5	200	6	1878,85	0,00	6	1879,31	0,02	6	1878,85
C2_2_6	200	6	1857,35	0,00	6	1857,35	0,00	6	1857,35
C2_2_7	200	6	1849,46	0,00	6	1849,46	0,00	6	1849,46
C2_2_8	200	6	1834,35	0,76	6	1824,52	0,22	6	1820,53
C2_2_9	200	6	1839,96	0,54	6	1848,14	0,99	6	1830,05
C2_2_10	200	6	1816,29	0,54	6	1812,27	0,31	6	1806,58
R1_2_1	200	20	5361,99	12,08	20	5347,76	11,78	20	4784,11
R1_2_2	200	18	4988,98	23,49	18	4769,66	18,06	18	4039,86
R1_2_3	200	18	3824,39	13,08	18	3696,53	9,30	18	3381,96
R1_2_4	200	18	3319,79	8,57	18	3276,30	7,15	18	3057,81
R1_2_5	200	18	5024,61	22,32	18	4652,59	13,26	18	4107,86
R1_2_6	200	18	4209,84	17,49	18	4213,53	17,59	18	3583,14
R1_2_7	200	18	3444,44	9,34	18	3473,16	10,26	18	3150,11
R1_2_8	200	18	3202,46	8,48	18	3231,77	9,48	18	2951,99
R1_2_9	200	18	4273,37	13,64	18	4442,12	18,12	18	3760,58
R1_2_10	200	18	3715,72	12,56	18	3645,88	10,44	18	3301,18
R2_2_1	200	-	-	-	-	-	-	4	4483,16
R2_2_2	200	4	4304,09	18,86	4	4304,09	18,86	4	3621,20
R2_2_3	200	4	3153,21	9,46	4	3559,55	23,57	4	2880,62
R2_2_4	200	4	2106,00	6,29	4	2046,10	3,27	4	1981,29

- datum not available.

Table B.22: Feasible solution values before and after tuning for *ng*-route pricing for instances by Gehring & Homberger

Instance	n	Before tuning			After tuning			Best known	
		Veh.	Dist.	Gap(%)	Veh.	Dist.	Gap(%)	Veh.	Dist.
R2_2_5	200	4	3677,03	9,21	4	3811,66	13,21	4	3366,79
R2_2_6	200	4	3057,12	4,95	4	3129,14	7,42	4	2913,03
R2_2_7	200	4	2664,96	8,72	4	2589,16	5,63	4	2451,14
R2_2_8	200	4	1915,80	3,56	4	1954,10	5,63	4	1849,87
R2_2_9	200	4	3187,81	3,10	4	3256,57	5,32	4	3092,04
R2_2_10	200	4	2794,72	5,26	4	2794,50	5,26	4	2654,97
RC1_2_1	200	-	-	-	-	-	-	18	3602,80
RC1_2_2	200	18	4848,85	49,24	18	4824,03	48,48	18	3249,05
RC1_2_3	200	18	3656,75	21,55	18	3653,30	21,44	18	3008,33
RC1_2_4	200	18	3248,65	13,92	18	3323,82	16,56	18	2851,68
RC1_2_5	200	18	4547,39	34,90	18	4547,39	34,90	18	3371,00
RC1_2_6	200	18	4106,07	23,50	18	4106,07	23,50	18	3324,80
RC1_2_7	200	18	4085,62	28,10	18	4085,62	28,10	18	3189,32
RC1_2_8	200	18	3602,97	16,83	18	3764,30	22,06	18	3083,93
RC1_2_9	200	18	3812,66	23,74	18	3701,26	20,13	18	3081,13
RC1_2_10	200	18	3616,50	20,54	18	3456,24	15,20	18	3000,30
RC2_2_1	200	6	3576,69	15,39	6	3576,69	15,39	6	3099,53
RC2_2_2	200	5	3458,12	22,40	5	3458,12	22,40	5	2825,24
RC2_2_3	200	4	3093,49	18,89	4	3093,49	18,89	4	2601,87
RC2_2_4	200	4	2607,69	27,92	4	2607,69	27,92	4	2038,56
RC2_2_5	200	-	-	-	-	-	-	4	2911,46
RC2_2_6	200	-	-	-	-	-	-	4	2873,12
RC2_2_7	200	4	3339,19	32,20	4	3339,19	32,20	4	2525,83
RC2_2_8	200	4	3060,08	33,48	4	3060,08	33,48	4	2292,53
RC2_2_9	200	4	2292,15	5,38	4	2321,25	6,72	4	2175,04
RC2_2_10	200	4	2053,56	1,88	4	2082,11	3,30	4	2015,60

- datum not available.

Table B.23: Feasible solution values before and after tuning for (t, i)-route pricing for instances by Gehring & Homberger

Instance	n	Before tuning			After tuning			Best known	
		Veh.	Dist.	Gap(%)	Veh.	Dist.	Gap(%)	Veh.	Dist.
C1_2_1	200	20	2704,57	0,00	20	2704,57	0,00	20	2704,57
C1_2_2	200	18	4047,24	38,70	18	4019,84	37,77	18	2917,89
C1_2_3	200	18	2941,13	8,64	18	2911,43	7,54	18	2707,35
C1_2_4	200	18	2829,49	7,04	18	2763,55	4,55	18	2643,31
C1_2_5	200	20	2703,85	0,07	20	2987,55	10,57	20	2702,05
C1_2_6	200	20	2703,85	0,10	20	2701,04	0,00	20	2701,04
C1_2_7	200	20	2701,35	0,01	20	3197,42	18,38	20	2701,04
C1_2_8	200	19	3230,78	16,40	19	3230,78	16,40	19	2775,48
C1_2_9	200	18	3427,62	27,52	18	3427,62	27,52	18	2687,83
C1_2_10	200	18	3529,79	33,53	18	2999,01	13,45	18	2643,51
C2_2_1	200	6	1931,44	0,00	6	1931,44	0,00	6	1931,44
C2_2_2	200	6	1863,16	0,00	6	1863,16	0,00	6	1863,16
C2_2_3	200	6	1787,46	0,70	6	1786,56	0,65	6	1775,08
C2_2_4	200	6	1788,56	5,00	6	1772,27	4,04	6	1703,43
C2_2_5	200	6	1879,31	0,02	6	1879,31	0,02	6	1878,85
C2_2_6	200	6	1858,59	0,07	6	1869,27	0,64	6	1857,35
C2_2_7	200	6	1874,00	1,33	6	1864,42	0,81	6	1849,46
C2_2_8	200	6	1852,52	1,76	6	1835,36	0,81	6	1820,53
C2_2_9	200	6	2352,25	28,53	6	2352,25	28,53	6	1830,05
C2_2_10	200	6	1816,22	0,53	6	1821,55	0,83	6	1806,58
R1_2_1	200	20	5347,76	11,78	20	5361,99	12,08	20	4784,11
R1_2_2	200	18	4769,66	18,06	18	4988,97	23,49	18	4039,86
R1_2_3	200	18	3764,71	11,32	18	3827,17	13,16	18	3381,96
R1_2_4	200	18	3280,96	7,30	18	3302,58	8,00	18	3057,81
R1_2_5	200	18	5024,61	22,32	18	5024,61	22,32	18	4107,86
R1_2_6	200	18	4174,28	16,50	18	4347,24	21,32	18	3583,14
R1_2_7	200	18	3528,80	12,02	18	3447,71	9,45	18	3150,11
R1_2_8	200	18	3203,04	8,50	18	3196,39	8,28	18	2951,99
R1_2_9	200	18	4494,69	19,52	18	4442,12	18,12	18	3760,58
R1_2_10	200	18	3732,25	13,06	18	3676,47	11,37	18	3301,18
R2_2_1	200	-	-	-	-	-	-	4	4483,16
R2_2_2	200	4	4304,09	18,86	4	4304,09	18,86	4	3621,20
R2_2_3	200	4	3559,55	23,57	4	3559,55	23,57	4	2880,62
R2_2_4	200	4	2124,41	7,22	4	2183,38	10,20	4	1981,29

- datum not available.

Table B.23: Feasible solution values before and after tuning for (t, i)-route pricing for instances by Gehring & Homberger

Instance	n	Before tuning			After tuning			Best known	
		Veh.	Dist.	Gap(%)	Veh.	Dist.	Gap(%)	Veh.	Dist.
R2_2_5	200	4	3661,96	8,77	4	3624,07	7,64	4	3366,79
R2_2_6	200	4	3198,17	9,79	4	3026,45	3,89	4	2913,03
R2_2_7	200	4	2630,27	7,31	4	2668,86	8,88	4	2451,14
R2_2_8	200	4	1917,85	3,67	4	1905,11	2,99	4	1849,87
R2_2_9	200	4	3294,95	6,56	4	3301,49	6,77	4	3092,04
R2_2_10	200	4	2791,85	5,16	4	2809,83	5,83	4	2654,97
RC1_2_1	200	-	-	-	-	-	-	18	3602,80
RC1_2_2	200	18	3925,33	20,81	18	4848,85	49,24	18	3249,05
RC1_2_3	200	18	3563,87	18,47	18	3760,65	25,01	18	3008,33
RC1_2_4	200	18	3293,96	15,51	18	3230,11	13,27	18	2851,68
RC1_2_5	200	18	4581,05	35,90	18	4581,05	35,90	18	3371,00
RC1_2_6	200	18	4205,97	26,50	18	4205,97	26,50	18	3324,80
RC1_2_7	200	18	4093,12	28,34	18	4093,12	28,34	18	3189,32
RC1_2_8	200	18	3764,30	22,06	18	3759,31	21,90	18	3083,93
RC1_2_9	200	18	3822,35	24,06	18	3812,66	23,74	18	3081,13
RC1_2_10	200	18	3442,20	14,73	18	3627,06	20,89	18	3000,30
RC2_2_1	200	6	3576,69	15,39	6	3576,69	15,39	6	3099,53
RC2_2_2	200	5	3458,12	22,40	5	3458,12	22,40	5	2825,24
RC2_2_3	200	4	3093,49	18,89	4	3093,49	18,89	4	2601,87
RC2_2_4	200	4	2607,69	27,92	4	2607,69	27,92	4	2038,56
RC2_2_5	200	-	-	-	-	-	-	4	2911,46
RC2_2_6	200	-	-	-	-	-	-	4	2873,12
RC2_2_7	200	4	3339,19	32,20	4	3339,19	32,20	4	2525,83
RC2_2_8	200	4	2608,55	13,78	4	2634,11	14,90	4	2292,53
RC2_2_9	200	4	2414,66	11,02	4	2296,14	5,57	4	2175,04
RC2_2_10	200	4	2097,58	4,07	4	2107,47	4,56	4	2015,60

- datum not available.

Table B.24: Feasible solution values before and after tuning for (t, i)-route with 2-cycles pricing for instances by Gehring & Homberger

Instance	n	Before tuning			After tuning			Best known	
		Veh.	Dist.	Gap(%)	Veh.	Dist.	Gap(%)	Veh.	Dist.
C1_2_1	200	20	2704,57	0,00	20	2704,57	0,00	20	2704,57
C1_2_2	200	18	4047,24	38,70	18	4019,84	37,77	18	2917,89
C1_2_3	200	18	2891,20	6,79	18	2940,99	8,63	18	2707,35
C1_2_4	200	18	2787,27	5,45	18	2798,10	5,86	18	2643,31
C1_2_5	200	20	2703,85	0,07	20	2987,55	10,57	20	2702,05
C1_2_6	200	20	2701,04	0,00	20	2701,04	0,00	20	2701,04
C1_2_7	200	20	2701,35	0,01	20	2740,38	1,46	20	2701,04
C1_2_8	200	19	3230,78	16,40	19	3117,98	12,34	19	2775,48
C1_2_9	200	18	3427,62	27,52	18	3392,24	26,21	18	2687,83
C1_2_10	200	18	3529,79	33,53	18	3441,41	30,18	18	2643,51
C2_2_1	200	6	1931,44	0,00	6	1931,44	0,00	6	1931,44
C2_2_2	200	6	1863,16	0,00	6	1863,16	0,00	6	1863,16
C2_2_3	200	6	1832,00	3,21	6	1788,78	0,77	6	1775,08
C2_2_4	200	6	1796,35	5,45	6	1762,70	3,48	6	1703,43
C2_2_5	200	6	1894,56	0,84	6	2089,49	11,21	6	1878,85
C2_2_6	200	6	1972,56	6,20	6	1872,38	0,81	6	1857,35
C2_2_7	200	6	1904,63	2,98	6	1856,06	0,36	6	1849,46
C2_2_8	200	6	1969,42	8,18	6	1965,42	7,96	6	1820,53
C2_2_9	200	6	1930,72	5,50	6	1858,88	1,58	6	1830,05
C2_2_10	200	6	1830,22	1,31	6	1830,28	1,31	6	1806,58
R1_2_1	200	20	5361,99	12,08	20	5347,76	11,78	20	4784,11
R1_2_2	200	18	4988,98	23,49	18	4769,66	18,06	18	4039,86
R1_2_3	200	18	3897,42	15,24	18	3737,44	10,51	18	3381,96
R1_2_4	200	18	3286,83	7,49	18	3264,93	6,77	18	3057,81
R1_2_5	200	18	5024,61	22,32	18	4881,89	18,84	18	4107,86
R1_2_6	200	18	4347,24	21,32	18	4269,61	19,16	18	3583,14
R1_2_7	200	18	3479,22	10,45	18	3544,81	12,53	18	3150,11
R1_2_8	200	18	3185,71	7,92	18	3229,80	9,41	18	2951,99
R1_2_9	200	18	4442,12	18,12	18	4442,12	18,12	18	3760,58
R1_2_10	200	18	3781,51	14,55	18	3648,97	10,54	18	3301,18
R2_2_1	200	-	-	-	-	-	-	4	4483,16
R2_2_2	200	4	4304,09	18,86	4	4304,09	18,86	4	3621,20
R2_2_3	200	4	3559,55	23,57	4	3559,55	23,57	4	2880,62
R2_2_4	200	4	2231,94	12,65	4	2140,27	8,02	4	1981,29

- datum not available.

Table B.24: Feasible solution values before and after tuning for (t, i)-route with 2-cycles pricing for instances by Gehring & Homberger

Instance	n	Before tuning			After tuning			Best known	
		Veh.	Dist.	Gap(%)	Veh.	Dist.	Gap(%)	Veh.	Dist.
R2_2_5	200	4	3574,26	6,16	4	3865,64	14,82	4	3366,79
R2_2_6	200	4	3926,62	34,80	4	3926,62	34,80	4	2913,03
R2_2_7	200	4	2677,67	9,24	4	2694,46	9,93	4	2451,14
R2_2_8	200	4	1960,71	5,99	4	1983,05	7,20	4	1849,87
R2_2_9	200	4	3201,73	3,55	4	3311,11	7,08	4	3092,04
R2_2_10	200	4	2805,80	5,68	4	2832,31	6,68	4	2654,97
RC1_2_1	200	-	-	-	-	-	-	18	3602,80
RC1_2_2	200	18	4824,03	48,48	18	4848,85	49,24	18	3249,05
RC1_2_3	200	18	3559,72	18,33	18	3667,80	21,92	18	3008,33
RC1_2_4	200	18	3274,32	14,82	18	3254,28	14,12	18	2851,68
RC1_2_5	200	18	4547,39	34,90	18	4547,39	34,90	18	3371,00
RC1_2_6	200	18	4106,07	23,50	18	4106,07	23,50	18	3324,80
RC1_2_7	200	18	4085,62	28,10	18	4093,12	28,34	18	3189,32
RC1_2_8	200	18	3759,31	21,90	18	3764,30	22,06	18	3083,93
RC1_2_9	200	18	3813,68	23,78	18	3750,50	21,72	18	3081,13
RC1_2_10	200	18	3473,80	15,78	18	3615,58	20,51	18	3000,30
RC2_2_1	200	6	3576,69	15,39	6	3576,69	15,39	6	3099,53
RC2_2_2	200	5	3458,12	22,40	5	3458,12	22,40	5	2825,24
RC2_2_3	200	4	3093,49	18,89	4	3093,49	18,89	4	2601,87
RC2_2_4	200	4	2607,69	27,92	4	2607,69	27,92	4	2038,56
RC2_2_5	200	-	-	-	-	-	-	4	2911,46
RC2_2_6	200	-	-	-	-	-	-	4	2873,12
RC2_2_7	200	4	3339,19	32,20	4	3339,19	32,20	4	2525,83
RC2_2_8	200	4	3060,08	33,48	4	3060,08	33,48	4	2292,53
RC2_2_9	200	4	2407,24	10,68	4	2322,55	6,78	4	2175,04
RC2_2_10	200	4	2089,29	3,66	4	2125,08	5,43	4	2015,60

- datum not available.

Table B.25: Feasible solution values before and after tuning for instances by Taillard

Instances	n	Before tuning		After tuning		Best known
		Heur.	Gap(%).	Heur.	Gap(%)	
tai27e02	27	2850	0,00	2850	0,00	2850
tai27e04	27	2822	0,00	2822	0,00	2822
tai27e06	27	2814	0,00	2814	0,00	2814
tai27e08	27	2430	0,00	2430	0,00	2430
tai27e10	27	2994	0,00	2994	0,00	2994
tai27e12	27	3070	0,00	3070	0,00	3070
tai27e14	27	3568	0,00	3568	0,00	3568
tai27e16	27	3124	0,00	3124	0,00	3124
tai27e18	27	2862	3,77	**2758**	0,00	2758
tai27e20	27	2638	0,00	2638	0,00	2638
tai45e02	45	5734	0,00	5734	0,00	5734
tai45e04	45	7182	7,23	**6698**	0,00	6698
tai45e06	45	7112	7,56	**6612**	0,00	6612
tai45e08	45	6648	1,43	**6554**	0,00	6554
tai45e10	45	8286	0,00	8286	0,00	8286
tai45e12	45	7792	3,75	**7510**	0,00	7510
tai45e14	45	6854	0,00	6854	0,00	6854
tai45e16	45	6970	6,90	**6520**	0,00	6520
tai45e18	45	6906	0,00	6906	0,00	6906
tai45e20	45	6842	5,10	**6510**	0,00	6510
tai75e02	75	15796	9,36	**15760**	9,11	14444
tai75e04	75	16646	21,56	**14420**	5,30	13694
tai75e06	75	14978	19,50	**13876**	10,71	12534
tai75e08	75	15556	11,53	15824	13,45	13948
tai75e10	75	16326	15,04	**15970**	12,53	14192
tai75e12	75	14664	14,92	**14478**	13,46	12760
tai75e14	75	14634	16,11	**13570**	7,66	12604
tai75e16	75	14978	5,45	**14498**	2,07	14204
tai75e18	75	14712	8,98	15662	16,01	13500
tai75e20	75	15422	1,06	17352	13,71	15260
tai125e02	125	46462	26,34	**42940**	16,76	36776
tai125e04	125	41694	22,87	**40136**	18,28	33934
tai125e06	125	43774	23,15	**41602**	17,04	35546
tai125e08	125	45318	24,66	**40458**	11,29	36354
tai125e10	125	45926	31,60	**44256**	26,82	34898
tai125e12	125	38334	18,31	40198	24,06	32402

Table B.25: Feasible solution values before and after tuning for instances by Taillard

Instances	n	Before tuning		After tuning		Best known
		Heur.	Gap(%).	Heur.	Gap(%)	
tai125e14	125	39688	29,92	**36990**	21,09	30548
tai125e16	125	40568	19,32	**39316**	15,64	33998
tai125e18	125	45634	15,24	48008	21,23	39600
tai125e20	125	36822	15,08	**36702**	14,71	31996
tai175e02	175	69530	35,10	**67718**	31,58	51464
tai175e04	175	77866	20,71	**76768**	19,01	64506
tai175e06	175	79668	42,86	**71632**	28,45	55768
tai175e08	175	71682	25,03	71776	25,19	57334
tai175e10	175	69982	34,48	**61306**	17,81	52040
tai175e12	175	77654	30,06	**71600**	19,92	59704
tai175e14	175	76618	37,47	**67188**	20,55	55736
tai175e16	175	75666	32,13	**68910**	20,33	57266
tai175e18	175	68774	31,87	**59194**	13,50	52152
tai175e20	175	72690	27,50	**70434**	23,54	57014
tai343e02	343	185840	20,66	**184058**	19,50	154018
tai343e04	343	196150	21,01	**190310**	17,41	162092
tai343e06	343	184664	28,00	**175638**	21,74	144274
tai343e08	343	171360	28,10	**162628**	21,57	133770
tai343e10	343	188398	23,27	**180392**	18,04	152828
tai343e12	343	188482	15,67	190128	16,68	162954
tai343e14	343	182510	21,33	**178848**	18,89	150428
tai343e16	343	182430	18,26	182702	18,43	154264
tai343e18	343	172428	26,14	**170722**	24,89	136694
tai343e20	343	191670	26,47	**179040**	18,14	151552
tai729e01	729	543394	15,70	**513152**	9,26	469650
tai729e02	729	534864	12,56	547232	15,16	475184
tai729e03	729	521724	16,49	**503124**	12,34	447854
tai729e04	729	510992	12,24	**501728**	10,20	455268
tai729e05	729	545956	14,83	**514090**	8,12	475466
tai729e06	729	557476	19,39	**529800**	13,46	466946
tai729e07	729	521600	14,77	524504	15,41	454480
tai729e08	729	493738	-40,88	**488966**	-41,45	835098
tai729e09	729	479628	12,20	490250	14,68	427478
tai729e10	729	527824	15,39	536726	17,33	457434

Table B.26: Feasible solution values before and after tuning for instances by Pellegrini et al. (166)

Instances	n	Before tuning		After tuning		Best known
		Heur.	Gap(%).	Heur.	Gap(%)	
EuclideanStructured.537000.n100.sp72.00	100	126101996	0,92	**126053982**	0,88	124958594
EuclideanStructured.540000.n100.sp72.00	100	113177896	1,78	**111615464**	0,38	111193532
EuclideanStructured.555000.n100.sp72.00	100	107122462	1,09	**106945848**	0,92	105972320
EuclideanStructured.587000.n100.sp72.00	100	106533484	0,92	**106085166**	0,50	105560170
EuclideanStructured.588000.n60.sp72.00	60	32753338	1,81	**32318756**	0,45	32172576
EuclideanStructured.593000.n100.sp72.00	100	125186284	0,50	**125172160**	0,48	124568788
EuclideanStructured.595000.n100.sp72.00	100	112653820	1,18	**111905818**	0,51	111334982
EuclideanStructured.603000.n60.sp72.00	60	47490820	0,51	**47353132**	0,22	47247944
EuclideanStructured.615000.n100.sp72.00	100	109070870	0,28	109655416	0,82	108763200
EuclideanStructured.624000.n60.sp72.00	60	38960204	1,15	**38761498**	0,63	38517234
EuclideanStructured.626000.n60.sp72.00	60	30571916	0,11	30773456	0,77	30537816
EuclideanStructured.631000.n60.sp72.00	60	36154494	0,49	**36125564**	0,41	35979406
EuclideanStructured.634000.n60.sp72.00	60	35003536	0,72	35056588	0,87	34754410
EuclideanStructured.634000.n100.sp72.00	100	102116072	1,05	**101941254**	0,87	101057394
EuclideanStructured.635000.n60.sp72.00	60	41101442	0,06	41412324	0,82	41074780
EuclideanStructured.637000.n100.sp72.00	100	108099550	0,97	**107541876**	0,44	107066066
EuclideanStructured.639000.n60.sp72.00	60	28953916	0,25	29090568	0,72	28882396
EuclideanStructured.639000.n80.sp72.00	80	63220106	0,89	**63169530**	0,81	62663506
EuclideanStructured.643000.n60.sp72.00	60	37748692	2,06	**37157546**	0,46	36986278
EuclideanStructured.643000.n100.sp72.00	100	127466438	1,07	127846016	1,37	126115736
EuclideanStructured.644000.n100.sp72.00	100	108629678	1,17	**108359930**	0,92	107374218
EuclideanStructured.647000.n80.sp72.00	80	64693152	0,90	**64428204**	0,48	64117308
EuclideanStructured.650000.n100.sp72.00	100	117999756	1,36	**117261026**	0,72	116421904

Table B.26: Feasible solution values before and after tuning for instances by Pellegrini et al. (166)

Instances	n	Before tuning		After tuning		Best known
		Heur.	Gap(%).	Heur.	Gap(%)	
EuclideanStructured.652000.n100.sp72.00	100	121745630	1,61	**120809290**	0,82	119822342
EuclideanStructured.654000.n80.sp72.00	80	69464896	0,71	69518152	0,79	68976650
EuclideanStructured.662000.n60.sp72.00	60	40358550	0,07	40521384	0,47	40332044
EuclideanStructured.666000.n80.sp72.00	80	75612150	1,48	**74672118**	0,22	74510610
EuclideanStructured.667000.n100.sp72.00	100	109498992	0,61	**109477566**	0,59	108839840
EuclideanStructured.669000.n60.sp72.00	60	43776704	1,96	**43332014**	0,93	42933788
EuclideanStructured.669000.n80.sp72.00	80	69347668	0,72	69399256	0,80	68849172
EuclideanStructured.669000.n100.sp72.00	100	111667480	1,67	**110681660**	0,77	109837720
EuclideanStructured.670000.n80.sp72.00	80	59125324	0,58	**58877516**	0,16	58783702
EuclideanStructured.671000.n100.sp72.00	100	119310922	0,94	**119079614**	0,75	118196026
EuclideanStructured.676000.n80.sp72.00	80	62154872	0,22	62185912	0,27	62018414
EuclideanStructured.680000.n60.sp72.00	60	30641144	0,00	30675710	0,11	30641144
EuclideanStructured.680000.n80.sp72.00	80	70017842	0,85	**69776400**	0,50	69429308
EuclideanStructured.681000.n100.sp72.00	100	114600792	0,77	**114323690**	0,53	113719688
EuclideanStructured.687000.n80.sp72.00	80	78578808	1,18	**77990868**	0,42	77663210
EuclideanStructured.689000.n80.sp72.00	80	69910068	1,12	**69461804**	0,48	69133272
EuclideanStructured.693000.n80.sp72.00	80	67560804	0,11	67807408	0,48	67483376
EuclideanStructured.697000.n60.sp72.00	60	43080364	0,57	**43030930**	0,46	42835112
EuclideanStructured.699000.n80.sp72.00	80	66778258	0,35	66857812	0,47	66546888
EuclideanStructured.711000.n80.sp72.00	80	68007992	1,06	**67762410**	0,69	67297914
EuclideanStructured.715000.n60.sp72.00	60	36009162	0,95	**35781718**	0,31	35671124
EuclideanStructured.720000.n60.sp72.00	60	39810798	0,69	39831060	0,74	39538250
EuclideanStructured.726000.n100.sp72.00	100	115726316	0,37	115940326	0,55	115302874
EuclideanStructured.735000.n80.sp72.00	80	67309178	0,66	**67153448**	0,43	66868248
EuclideanStructured.735000.n100.sp72.00	100	118332872	1,19	**117405360**	0,40	116941494

Table B.26: Feasible solution values before and after tuning for instances by Pellegrini et al. (166)

Instances	n	Before tuning		After tuning		Best known
		Heur.	Gap(%).	Heur.	Gap(%)	
EuclideanStructured.736000.n80.sp72.00	80	62540626	0,36	62589060	0,44	62313738
EuclideanStructured.737000.n80.sp72.00	80	68748714	2,16	**67950346**	0,97	67298018
EuclideanStructured.739000.n60.sp72.00	60	35968118	1,51	**35705386**	0,77	35433530
EuclideanStructured.742000.n80.sp72.00	80	70309508	1,42	**69895334**	0,83	69322296
EuclideanStructured.746000.n60.sp72.00	60	35828738	0,99	**35680138**	0,57	35477438
EuclideanStructured.747000.n100.sp72.00	100	122640342	1,51	**121297418**	0,40	120819810
EuclideanStructured.751000.n80.sp72.00	80	70018174	0,56	**69736716**	0,16	69628302
EuclideanStructured.756000.n100.sp72.00	100	108687750	2,03	**106679456**	0,14	106529678
EuclideanStructured.761000.n100.sp72.00	100	114485876	0,68	**114376772**	0,58	113712662
EuclideanStructured.771000.n100.sp72.00	100	107584292	0,83	**107452776**	0,71	106697124
EuclideanStructured.775000.n80.sp72.00	80	72435146	0,48	**72322702**	0,33	72086996
EuclideanStructured.776000.n100.sp72.00	100	103593154	0,45	103670810	0,53	103126816
EuclideanStructured.778000.n60.sp72.00	60	40249296	0,48	**40206804**	0,38	40056360
EuclideanStructured.783000.n100.sp72.00	100	129979286	0,86	**129478756**	0,47	128869952
EuclideanStructured.788000.n80.sp72.00	80	79457928	0,59	**79218818**	0,29	78989256
EuclideanStructured.789000.n100.sp72.00	100	108323054	2,41	**107219742**	1,36	105777792
EuclideanStructured.804000.n80.sp72.00	80	70080390	0,59	70520312	1,22	69669752
EuclideanStructured.807000.n80.sp72.00	80	68760398	0,68	68864506	0,83	68295996
EuclideanStructured.812000.n100.sp72.00	100	109259930	1,66	**107616348**	0,14	107471096
EuclideanStructured.813000.n60.sp72.00	60	43512986	0,37	43760132	0,94	43352742
EuclideanStructured.819000.n80.sp72.00	80	70837686	1,30	**70425214**	0,71	69929564
EuclideanStructured.822000.n60.sp72.00	60	31819780	1,20	**31636398**	0,61	31443972
EuclideanStructured.825000.n60.sp72.00	60	35364636	0,81	**35119008**	0,11	35080218
EuclideanStructured.830000.n100.sp72.00	100	133853232	1,16	**132687362**	0,28	132320590
EuclideanStructured.835000.n80.sp72.00	80	68025776	0,81	**67735318**	0,38	67477546

Table B.26: Feasible solution values before and after tuning for instances by Pellegrini et al. (166)

Instances	n	Before tuning		After tuning		Best known
		Heur.	Gap(%).	Heur.	Gap(%)	
EuclideanStructured.844000.n60.sp72.00	60	36884964	0,57	**36783498**	0,29	36677312
EuclideanStructured.846000.n60.sp72.00	60	35021910	1,76	**34826330**	1,19	34417590
EuclideanStructured.850000.n60.sp72.00	60	39974644	0,18	40166512	0,66	39902508
EuclideanStructured.855000.n100.sp72.00	100	109292644	1,62	**108087986**	0,50	107552048
EuclideanStructured.857000.n100.sp72.00	100	111862108	1,97	**109989168**	0,27	109696520
EuclideanStructured.860000.n80.sp72.00	80	58616222	1,05	**58180258**	0,30	58005768
EuclideanStructured.864000.n100.sp72.00	100	113075778	1,43	**111934072**	0,41	111481954
EuclideanStructured.866000.n80.sp72.00	80	58309080	0,69	58416140	0,88	57907384
EuclideanStructured.867000.n60.sp72.00	60	39171654	1,30	**38997514**	0,85	38667874
EuclideanStructured.868000.n100.sp72.00	100	119723126	1,10	**118752686**	0,28	118415460
EuclideanStructured.872000.n60.sp72.00	60	37054136	0,50	**36954628**	0,23	36870068
EuclideanStructured.875000.n80.sp72.00	80	75163904	0,71	**75061670**	0,57	74636076
EuclideanStructured.879000.n60.sp72.00	60	43076966	0,23	43164374	0,43	42979374
EuclideanStructured.881000.n80.sp72.00	80	76060066	0,69	**75829002**	0,39	75535750
EuclideanStructured.883000.n60.sp72.00	60	41050756	0,83	41051262	0,83	40711496
EuclideanStructured.886000.n80.sp72.00	80	68542936	0,57	**68481590**	0,48	68155962
EuclideanStructured.893000.n100.sp72.00	100	108693198	0,28	109125394	0,68	108391024
EuclideanStructured.896000.n100.sp72.00	100	108884394	1,54	**107990798**	0,71	107231972
EuclideanStructured.909000.n80.sp72.00	80	68341434	0,99	**68176002**	0,75	67669876
EuclideanStructured.909000.n100.sp72.00	100	124703964	0,38	125043414	0,65	124231650
EuclideanStructured.911000.n80.sp72.00	80	69885442	0,96	**69375502**	0,22	69222340
EuclideanStructured.911000.n100.sp72.00	100	106029516	0,52	**105915974**	0,41	105478438
EuclideanStructured.915000.n60.sp72.00	60	36668538	0,94	**36478620**	0,42	36325406
EuclideanStructured.917000.n60.sp72.00	60	37613630	1,54	**37070096**	0,08	37041782
EuclideanStructured.929000.n80.sp72.00	80	66916400	1,04	**66750522**	0,79	66229198

Table B.26: Feasible solution values before and after tuning for instances by Pellegrini et al. (166)

Instances	n	Before tuning		After tuning		Best known
		Heur.	Gap(%).	Heur.	Gap(%)	
EuclideanStructured.930000.n60.sp72.00	60	37844896	0,47	**37825144**	0,41	37669320
EuclideanStructured.933000.n100.sp72.00	100	107994914	0,76	**107881640**	0,66	107177774
EuclideanStructured.935000.n100.sp72.00	100	107114922	1,25	**106314466**	0,49	105795498
EuclideanStructured.943000.n60.sp72.00	60	35524366	2,73	**34727306**	0,43	34578764
EuclideanStructured.949000.n60.sp72.00	60	30424336	0,54	**30422004**	0,53	30261308
EuclideanStructured.957000.n60.sp72.00	60	33849878	0,01	33876844	0,09	33845074
EuclideanStructured.959000.n100.sp72.00	100	100440456	1,05	**99798602**	0,40	99398590
EuclideanStructured.965000.n80.sp72.00	80	79329806	2,23	**78437542**	1,08	77597630
EuclideanStructured.967000.n60.sp72.00	60	40371244	0,00	40650524	0,69	40371244
EuclideanStructured.967000.n80.sp72.00	80	64599918	1,62	**63816742**	0,39	63570754
EuclideanStructured.969000.n80.sp72.00	80	69173908	2,02	**68405172**	0,89	67804576
EuclideanStructured.973000.n80.sp72.00	80	65801588	0,61	**65599136**	0,30	65404566
EuclideanStructured.973000.n100.sp72.00	100	105241482	1,09	**105157020**	1,01	104105856
EuclideanStructured.975000.n100.sp72.00	100	107537190	0,51	108296584	1,22	106996400
EuclideanStructured.978000.n80.sp72.00	80	71346416	0,42	71715750	0,94	71046250
EuclideanStructured.980000.n60.sp72.00	60	42819282	0,25	**42800772**	0,20	42713884
EuclideanStructured.983000.n60.sp72.00	60	40309178	1,14	**40037464**	0,46	39853938
EuclideanStructured.985000.n60.sp72.00	60	39723738	0,37	39784402	0,53	39576096
EuclideanStructured.986000.n80.sp72.00	80	71625386	1,29	**71342588**	0,89	70716600
EuclideanStructured.993000.n60.sp72.00	60	38541170	0,90	**38308468**	0,29	38198224
EuclideanStructured.997000.n60.sp72.00	60	35564488	1,58	**35438318**	1,22	35011292
EuclideanStructured.997000.n100.sp72.00	100	126227344	0,23	126514398	0,46	125938958
EuclideanStructured.998000.n80.sp72.00	80	72605726	1,02	**71917864**	0,06	71871968
EuclideanStructured.1004000.n100.sp72.00	100	105645950	0,80	**105364526**	0,53	104805358
EuclideanStructured.1007000.n100.sp72.00	100	107006620	0,19	107839132	0,97	106798368

Table B.26: Feasible solution values before and after tuning for instances by Pellegrini et al. (166)

Instances	n	Before tuning		After tuning		Best known
		Heur.	Gap(%).	Heur.	Gap(%)	
EuclideanStructured.1008000.n60.sp72.00	60	37347368	0,83	**37290976**	0,68	37039266
EuclideanStructured.1013000.n60.sp72.00	60	35068114	1,30	**34748834**	0,38	34616592
EuclideanStructured.1014000.n80.sp72.00	80	77185256	0,99	**76511412**	0,11	76427306
EuclideanStructured.1016000.n100.sp72.00	100	91677864	0,46	**91674316**	0,46	91255840
EuclideanStructured.1020000.n60.sp72.00	60	34656006	0,84	**34585986**	0,64	34365724
EuclideanStructured.1024000.n80.sp72.00	80	67573116	1,07	**67064944**	0,31	66860022
EuclideanStructured.1025000.n60.sp72.00	60	40248378	1,03	**39866462**	0,07	39838860
EuclideanStructured.1027000.n80.sp72.00	80	76264644	1,01	**76073382**	0,76	75500308
EuclideanStructured.1028000.n100.sp72.00	100	122424956	0,79	**122271470**	0,66	121469650
EuclideanStructured.1033000.n80.sp72.00	80	73861196	0,72	**73693142**	0,49	73333202
EuclideanStructured.1037000.n60.sp72.00	60	35430734	0,55	**35359380**	0,34	35238550
EuclideanStructured.1045000.n100.sp72.00	100	119872700	0,86	**119321714**	0,40	118850020
EuclideanStructured.1051000.n60.sp72.00	60	36862240	0,94	**36596722**	0,22	36517482
EuclideanStructured.1052000.n80.sp72.00	80	81979508	0,40	82310766	0,80	81655890
EuclideanStructured.1058000.n80.sp72.00	80	64179788	1,05	**63749886**	0,37	63513850
EuclideanStructured.1059000.n80.sp72.00	80	68856214	0,41	**68727802**	0,22	68576168
EuclideanStructured.1060000.n60.sp72.00	60	32900334	2,01	**32299808**	0,15	32252200
EuclideanStructured.1070000.n100.sp72.00	100	121796680	0,95	**121414144**	0,63	120651046
EuclideanStructured.1077000.n60.sp72.00	60	33339518	1,09	**33227680**	0,75	32981098
EuclideanStructured.1083000.n80.sp72.00	80	62485176	1,51	**61966678**	0,67	61556782
EuclideanStructured.1085000.n100.sp72.00	100	95596682	0,07	96067074	0,56	95532018
EuclideanStructured.1089000.n100.sp72.00	100	116565038	0,71	**116240904**	0,43	115739778
EuclideanStructured.1091000.n60.sp72.00	60	32896204	0,43	33061144	0,93	32756782
EuclideanStructured.1091000.n80.sp72.00	80	60287916	1,29	**59614158**	0,16	59518042
EuclideanStructured.1093000.n80.sp72.00	80	71403942	0,34	71549870	0,54	71165068

Table B.26: Feasible solution values before and after tuning for instances by Pellegrini et al. (166)

Instances	n	Before tuning		After tuning		Best known
		Heur.	Gap(%).	Heur.	Gap(%)	
EuclideanStructured.1096000.n60.sp72.00	60	45461940	1,07	45482160	1,11	44981832
EuclideanStructured.1100000.n80.sp72.00	80	82209692	1,23	**82053856**	1,04	81210906

B.2 Charts

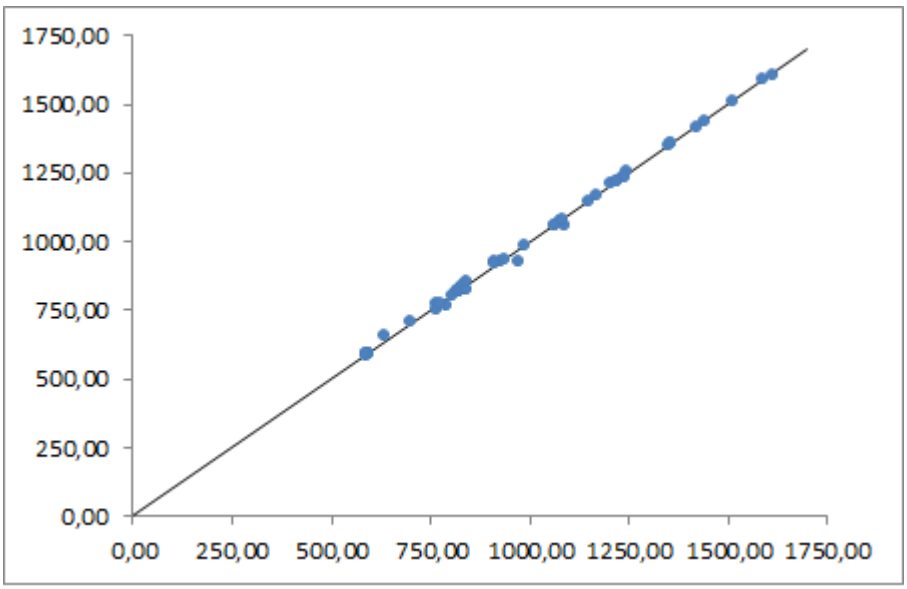

Figure B.1: Distribution of valid lower bounds for ng-route pricing before and after tuning for instances by Solomon (200)

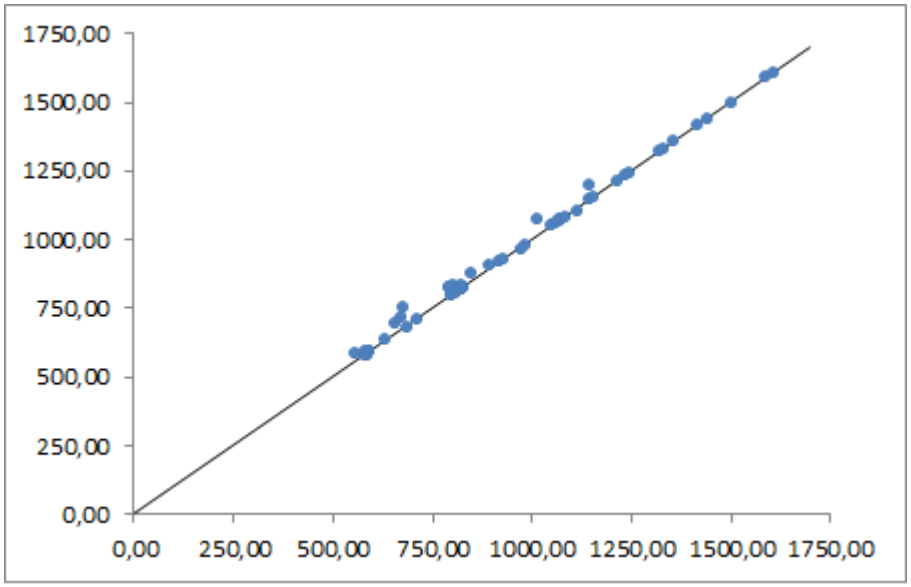

Figure B.2: Distribution of valid lower bounds for (t, i)-route pricing before and after tuning for instances by Solomon (200)

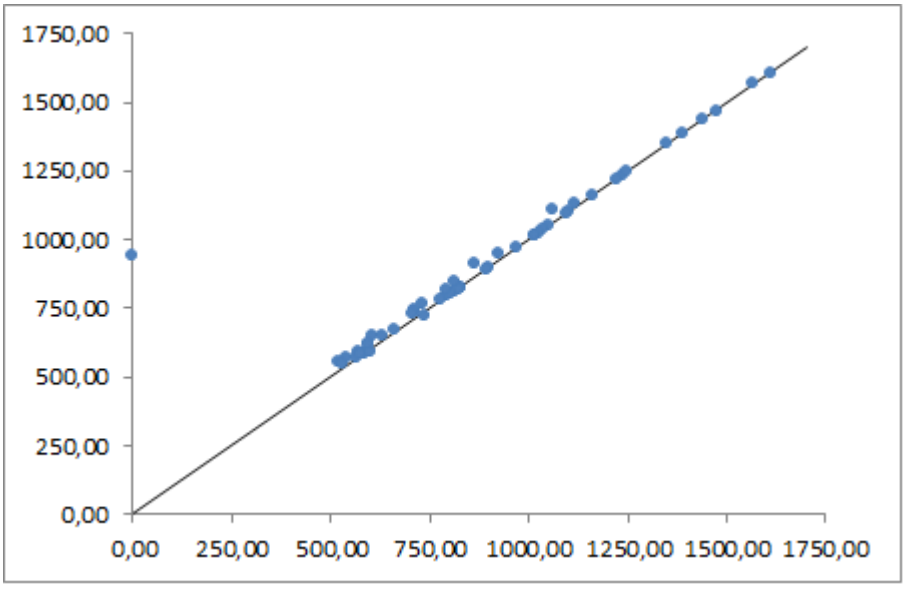

Figure B.3: Distribution of valid lower bounds for (t, i)-route with 2-cycles pricing before and after tuning for instances by Solomon (200)

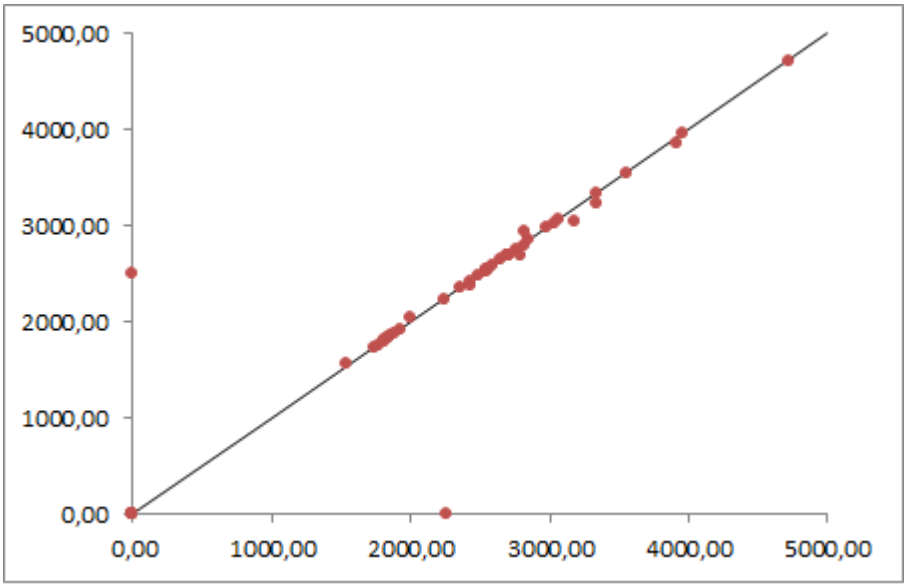

Figure B.4: Distribution of valid lower bounds for *ng*-route pricing before and after tuning for instances by Gehring & Homberger

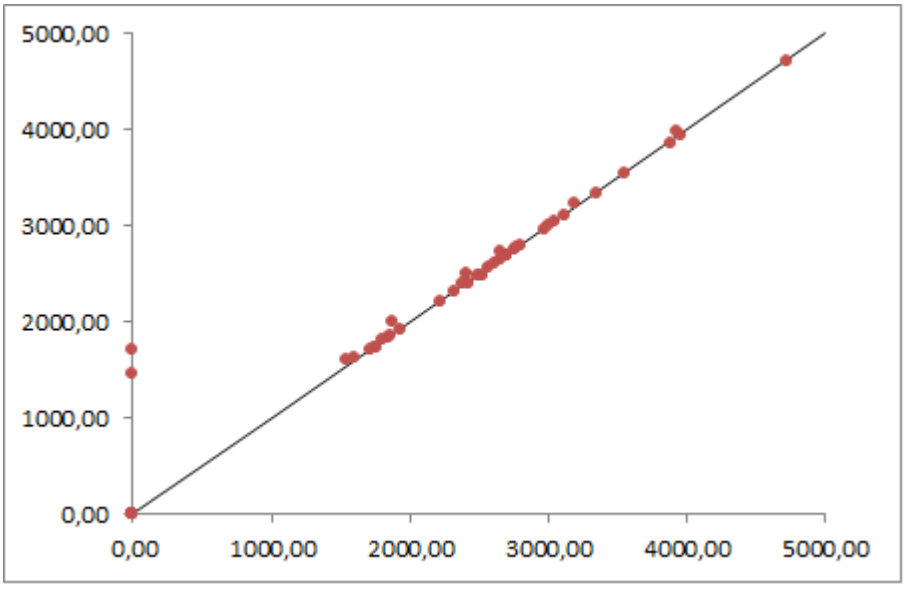

Figure B.5: Distribution of valid lower bounds for (t, i)-route pricing before and after tuning for instances by Gehring & Homberger

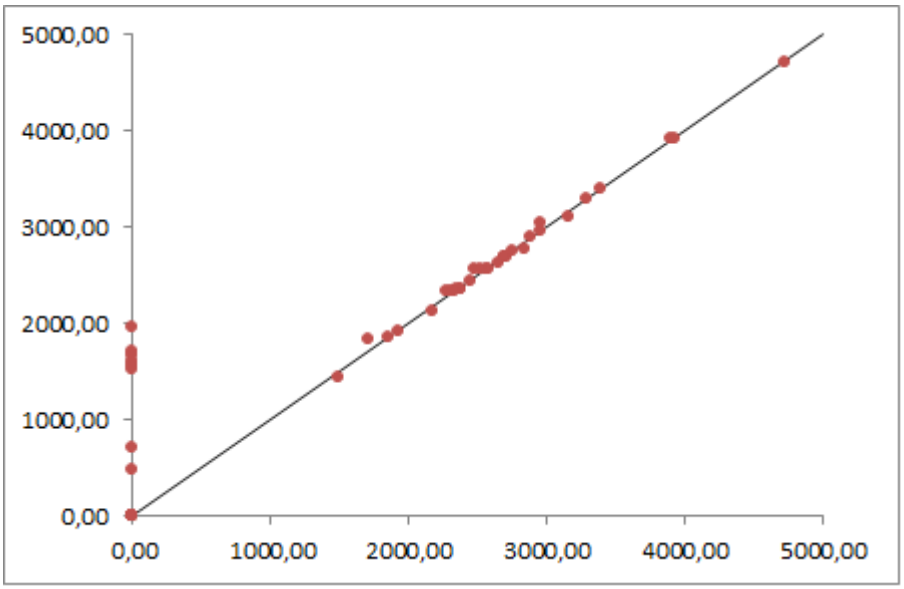

Figure B.6: Distribution of valid lower bounds for (t, i)-route with 2-cycles pricing before and after tuning for instances by Gehring & Homberger

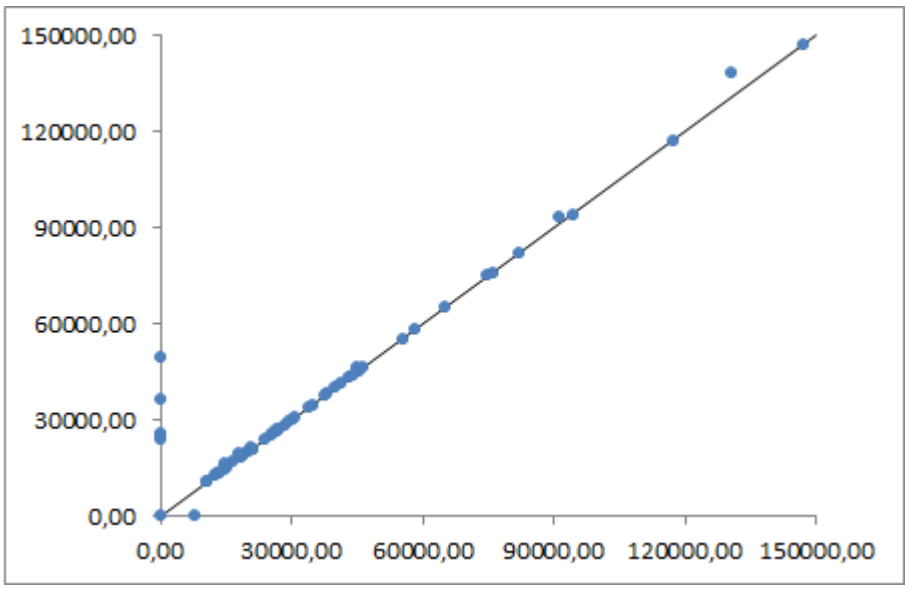

Figure B.7: Distribution of valid lower bounds for *ng*-route pricing before and after tuning for instances by Uchoa et al. (211)

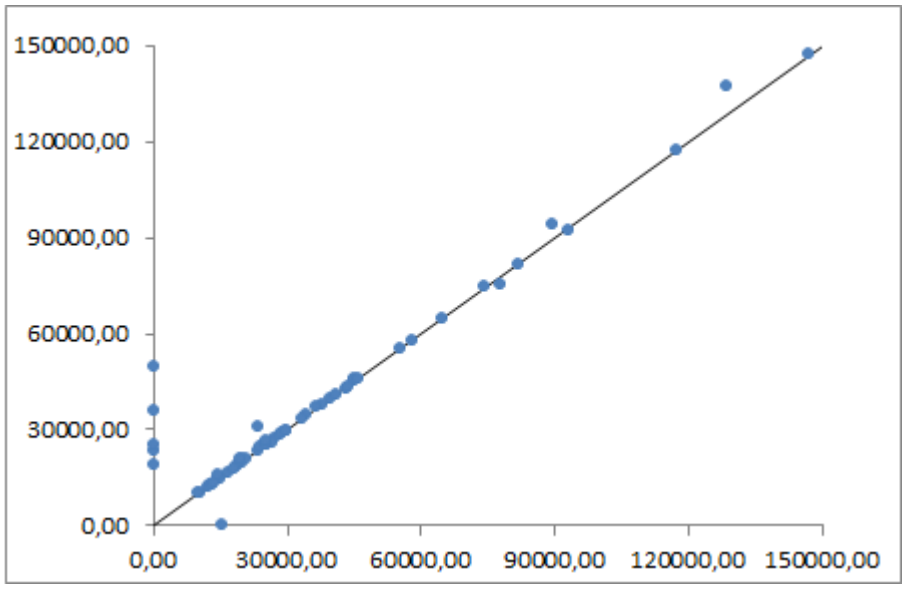

Figure B.8: Distribution of valid lower bounds for (q, i)-route pricing before and after tuning for instances by Uchoa et al. (211)

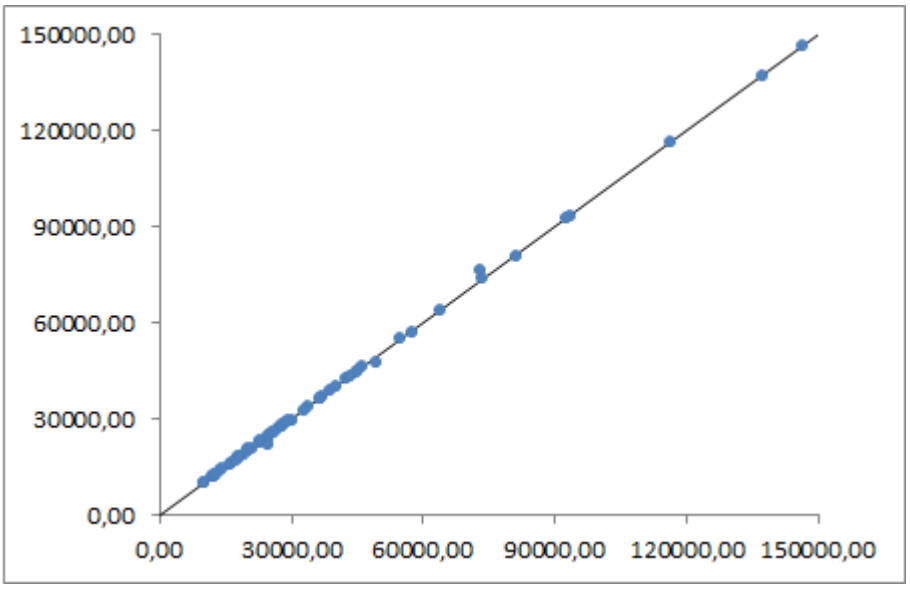

Figure B.9: Distribution of valid lower bounds for (q, i)-route with 2-cycles pricing before and after tuning for instances by Uchoa et al. (211)

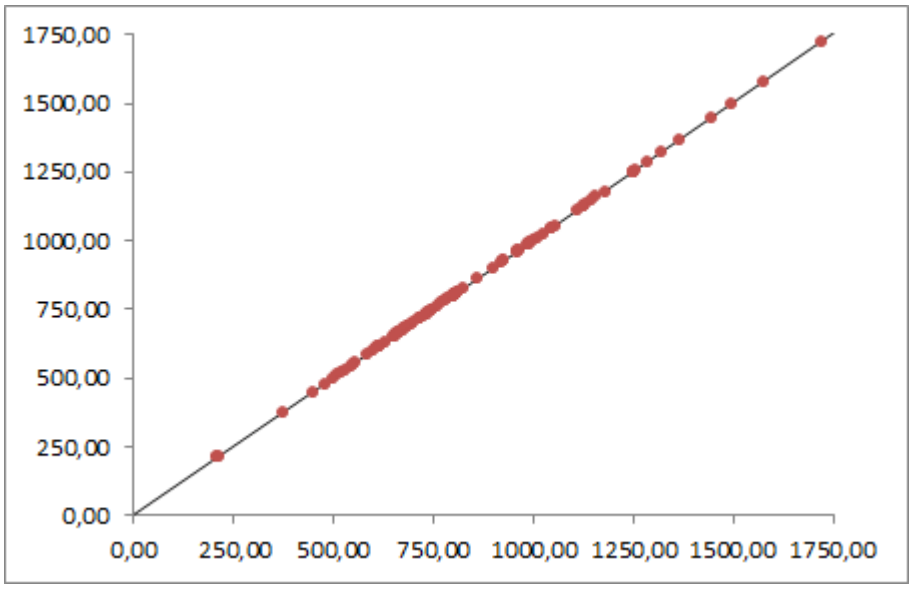

Figure B.10: Distribution of valid lower bounds for ng-route pricing before and after tuning for instances A, B, P, E, M

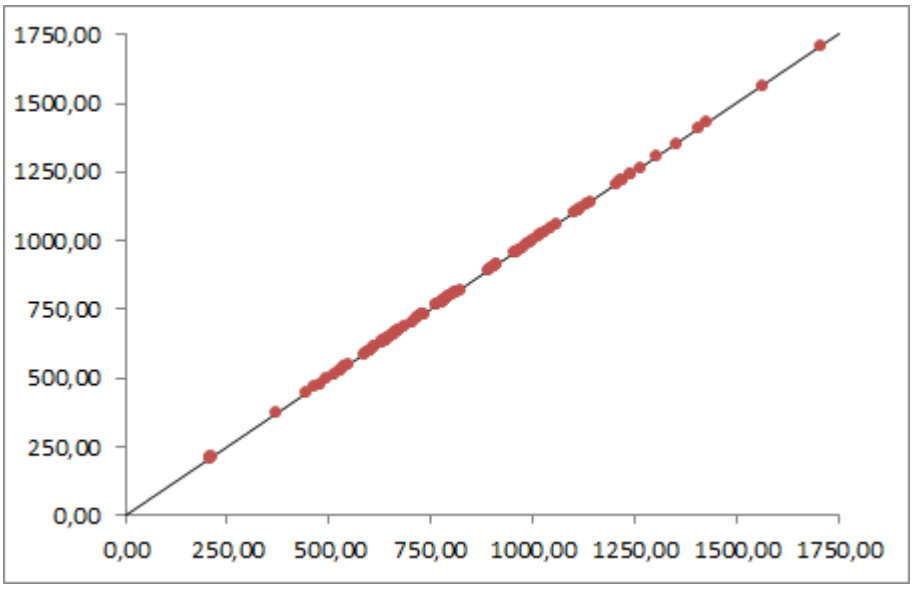

Figure B.11: Distribution of valid lower bounds for (q, i)-route pricing before and after tuning for instances A, B, P, E, M

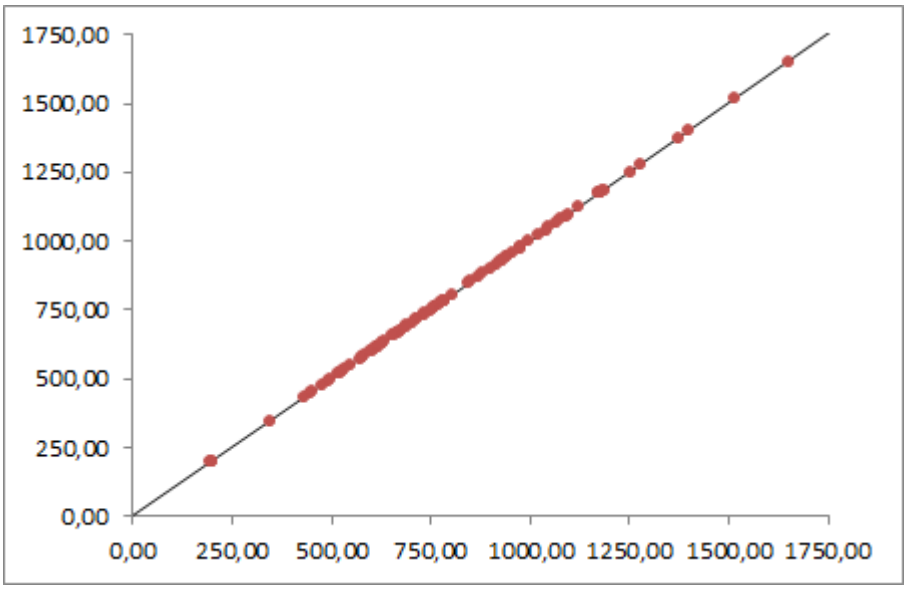

Figure B.12: Distribution of valid lower bounds for (q, i)-route with 2-cycles pricing before and after tuning for instances A, B, P, E, M

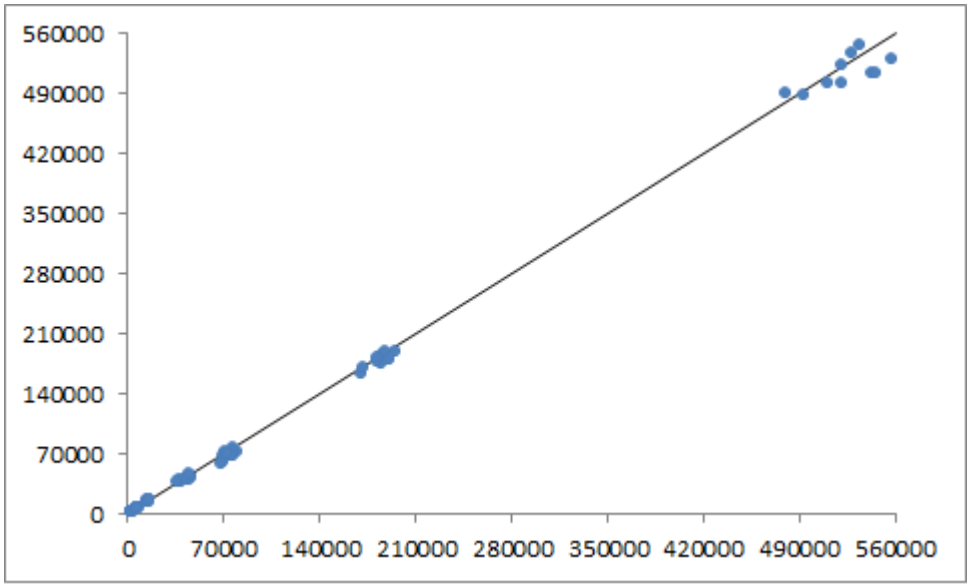

Figure B.13: Distribution of heuristic values before and after tuning for instances by Taillard

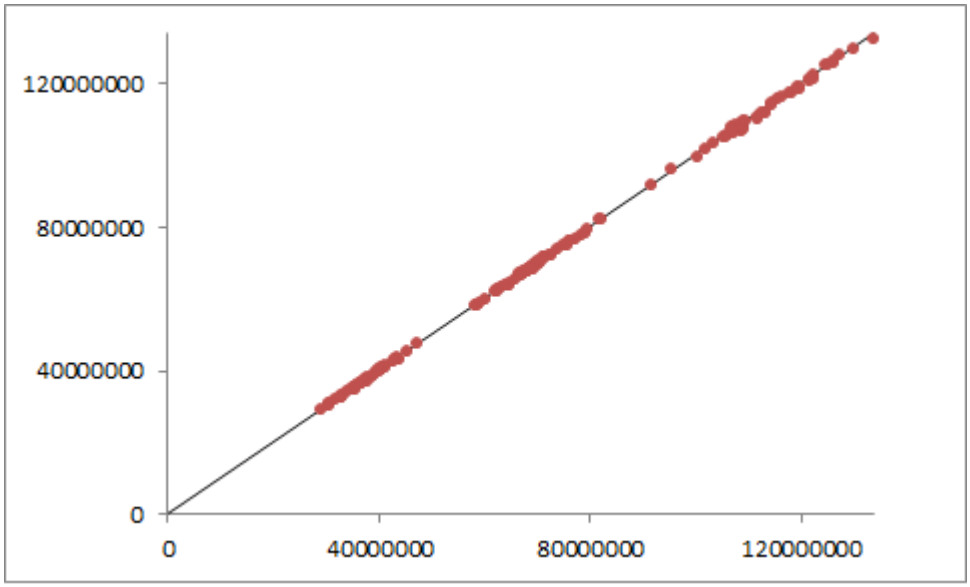

Figure B.14: Distribution of heuristic values before and after tuning for instances by Pellegrini et al. (166)