UNIVERSIDADE FEDERAL FLUMINENSE

EDUARDO VIEIRA QUEIROGA

Exact and heuristic approaches for relaxed correlation clustering and vehicle routing with backhauls

NITERÓI

2021

UNIVERSIDADE FEDERAL FLUMINENSE

EDUARDO VIEIRA QUEIROGA

Exact and heuristic approaches for relaxed correlation clustering and vehicle routing with backhauls

Thesis presented to the Computing Graduate Program of the Universidade Federal Fluminense in partial fulfillment of the requirements for the degree of Doctor of Science. Topic Area: Computer Science.

Advisor:
Yuri Abitbol De Menezes Frota

Co-Advisor:
Eduardo Uchoa Barboza

NITERÓI

2021

Ficha catalográfica automática - SDC/BEE
Gerada com informações fornecidas pelo autor

Q3e Queiroga, Eduardo Vieira
Exact and heuristic approaches for relaxed correlation clustering and vehicle routing with backhauls / Eduardo Vieira Queiroga ; Yuri Abitbol De Menezes Frota, orientador ; Eduardo Uchoa Barboza, coorientador. Niterói, 2021.
204 f. : il.

Tese (doutorado)-Universidade Federal Fluminense, Niterói, 2021.

DOI: http://dx.doi.org/10.22409/PGC.2021.d.09682216451

1. Roteamento de veículos com coleta na volta. 2. Roteamento de veículos capacitado. 3. Clusterização em grafos de sinais. 4. Produção intelectual. I. Frota, Yuri Abitbol De Menezes, orientador. II. Barboza, Eduardo Uchoa, coorientador. III. Universidade Federal Fluminense. Instituto de Computação. IV. Título.

CDD -

Bibliotecário responsável: Debora do Nascimento - CRB7/6368

EDUARDO VIEIRA QUEIROGA

Exact and heuristic approaches for relaxed correlation clustering and vehicle routing with backhauls

Thesis presented to the Computing Graduate Program of the Universidade Federal Fluminense in partial fulfillment of the requirements for the degree of Doctor of Science.

Topic Area: Computer Science.

Approved in May, 2021.

Yuri Abitbol De Menezes Frota

Prof. D.Sc. Yuri Abitbol De Menezes Frota / IC-UFF (President)

EDUARDO UCHOA BARBOZA eduardouchoa@id.uff.br:13768037800

Assinado de forma digital por EDUARDO UCHOA BARBOZA eduardouchoa@id.uff.br:13768037800 Dados: 2021.06.29 14:05:49 -03'00'

Prof. D.Sc. Eduardo Uchoa Barboza / TEP-UFF

Prof. D.Sc. Isabel Cristina Mello Rosseti / IC-UFF

LUIZ SATORU OCHI luizsatoru@id.uff.br:49197371734

Assinado de forma digital por LUIZ SATORU OCHI luizsatoru@id.uff.br:49197371734 Dados: 2021.07.02 13:32:36 -03'00'

Prof. D.Sc. Luiz Satoru Ochi / IC-UFF

Prof. D.Sc. Márcia Helena Costa Fampa / COPPE-UFRJ

Prof. D.Sc. Rosa Maria Videira de Figueiredo / UAPV, France

Niterói

2021

Para a minha família.

Agradecimentos

Gostaria de expressar a minha profunda gratidão a:

- Deus, pela vida.
- Minha família, em especial aos meus pais José Edu e Antonia, por terem cuidado de mim e da minha irmã Amanda com amor e dedicação desde o começo de nossas vidas.
- Meus orientadores Yuri Frota e Eduardo Uchoa, a quem admiro intelectual e pessoalmente. Obrigado pela amizade, suporte e pelos ensinamentos em valiosas conversas sobre temas diversos.
- Lucídio Cabral e Anand Subramanian (orientadores de mestrado na UFPB), pela amizade e por terem me introduzido ao universo da otimização, além de terem me ajudado de diversas formas na minha jornada acadêmica.
- Meus amigos e/ou colaboradores Teobaldo, João, Thaylon, Thiago Gouveia, Bruno, Gilmar, Markov, Artur, Thibaut, Renatha, Vitor Hugo, Paulo, Guillaume, Ruslan, Verônica, e Rian.
- Instituto de Computação-UFF, que proporciona uma estrutura de excelência para os seus alunos desenvolverem pesquisa de alto nível.
- CAPES, pelo financiamento da pesquisa realizada neste trabalho.

Resumo

Inicialmente, este trabalho aborda o *relaxed correlation clustering* (RCC), que é um problema NP-difícil de particionamento de vértices que busca minimizar o desequilíbrio relaxado em grafos de sinais, tendo aplicações em análise de redes sociais. Para resolvê-lo, são propostas duas novas formulações de programação linear inteira e uma meta-heurística baseada em busca local. Esta última usa estruturas de dados auxiliares para realizar, de forma eficiente, avaliações de movimentos durante o processo de busca. Experimentos computacionais em instâncias existentes e em novas instâncias geradas demonstram a superioridade das abordagens propostas quando comparadas com métodos da literatura. Enquanto as abordagens exatas obtiveram soluções ótimas para instâncias em aberto, a meta-heurística proposta foi capaz de encontrar soluções de alta qualidade em tempo razoável de CPU.

A segunda parte deste trabalho é sobre o *problema de roteamento de veículos com coleta na volta* (VRPB, do inglês *vehicle routing problem with backhauls*), que é uma variante clássica de roteamento de veículos que considera dois tipos de clientes: entrega e coleta. No VRPB, uma rota precisa visitar clientes de entrega antes de clientes de coleta. Para resolver o VRPB de forma exata, são propostos dois algoritmos do tipo *branch-cut-and-price* (BCP). O primeiro deles segue o modelo tradicional com apenas um subproblema de *princing*, enquanto o segundo explora o particionamento dos clientes em entrega e coleta e define dois subproblemas. Experimentos computacionais mostram que os algoritmos de BCP são capazes de obter soluções ótimas, muitas delas sendo inéditas, para todas as instâncias da literatura, que possuem até 200 clientes. Foi observado que a abordagem com dois subproblemas de *pricing* é mais eficiente que a tradicional. Além disso, novas instâncias com até 1000 clientes foram geradas para as quais bons limitantes foram encontrados. Também foram avaliadas três meta-heurísticas efetivas, onde duas delas exploram, em níveis diferentes, informações específicas do problema.

A última parte deste trabalho propõe um *Partial OPtimization Metaheuristic Under Special Intensification Conditions* (POPMUSIC) para o clássico problema de roteamento de veículos capacitado (CVRP). A abordagem proposta usa um algoritmo de BCP como uma heurística para resolver subproblemas com dimensões que geralmente variam entre 25 e 200 clientes. Experimentos foram realizados em instâncias tendo entre 302 e 1000 clientes. Partindo de soluções iniciais obtidas por algumas das melhores meta-heurísticas da literatura, o POPMUSIC foi capaz de obter consistentemente melhores soluções para execuções longas de até 32 horas. Ao começar da melhor solução disponível na *CVRP library*, o POPMUSIC foi capaz de encontrar novas melhores soluções para várias instâncias, incluindo algumas muito grandes. Em um experimento final, o POPMUSIC foi aplicado com sucesso para o VRP com frota heterogênea e o VRPB.

Palavras-chave: Correlação de Clusters Relaxado, Roteamento de Veículos com Coleta na Volta, Roteamento de Veículos Capacitado, POPMUSIC.

Abstract

At first, this work approaches the relaxed correlation clustering (RCC), which is a vertex partitioning NP-Hard problem that aims at minimizing the relaxed imbalance in signed graphs, having applications in social network analysis. To solve it, we propose two linear integer programming formulations and a local search-based heuristic. The latter relies on auxiliary data structures to efficiently perform move evaluations during the search process. Computational experiments on existing and newly proposed benchmark instances demonstrate the superior performance of the proposed approaches when compared to those available in the literature. While the exact approaches obtained optimal solutions for open problems, the proposed heuristic was capable of finding high quality solutions within a reasonable CPU time.

The second part of this work is about the vehicle routing problem (VRP) with backhauls (VRPB), which is a VRP with two types of customers: linehaul and backhaul ones. In the VRPB, a route must visit linehaul customers before backhaul customers. To solve VRPB exactly, we propose two branch-cut-and-price (BCP) algorithms. The first of them follows the traditional approach with one pricing subproblem, whereas the second one exploits the linehaul/backhaul customer partitioning and defines two pricing subproblems. Computational experiments show that the BCP algorithms are capable of obtaining optimal solutions for all existing benchmark instances with up to 200 customers, many of them for the first time. It is observed that the approach involving two pricing subproblems is more efficient computationally than the traditional one. Moreover, new instances with up to 1000 customers are also proposed for which were provided tight bounds. Three effective heuristics were also evaluated, where two of them take advantage, at different levels, of problem-specific information.

The last part of this work proposes a Partial OPtimization Metaheuristic Under Special Intensification Conditions (POPMUSIC) for the classical capacitated vehicle routing problem (CVRP). The proposed approach uses a BCP algorithm as a powerful heuristic to solve subproblems whose dimensions are typically between 25 and 200 customers. Computational experiments were carried out on instances having between 302 and 1000 customers. Using initial solutions generated by some of the best heuristics for the problem, POPMUSIC was able to obtain consistently better solutions for long runs of up to 32 hours. Starting from the best known solutions in CVRP library, POPMUSIC was able to find new best solutions for several instances, including some very large ones. In a final experiment, POPMUSIC was successfully applied to tackle the heterogeneous fleet VRP and the VRPB.

Keywords: Relaxed Correlation Clustering, Vehicle Routing With Backhauls, Capacitated Vehicle Routing Problem, POPMUSIC.

List of Figures

List of Tables

Contents

Chapter 1

Introduction

Graphs are mathematical structures composed by a set of vertices (a.k.a. nodes) and a set of edges (or arcs) which connect pairs of vertices. There is a class of problems called *graph clustering*, which is related to the task of finding groups (a.k.a. clusters) of vertices in a graph. Among these problems, there are those that consider signed graphs – graphs whose the edges (or arcs) have a positive (+) or a negative (-) label (such as the one illustrated in Figure 1.1). The first part of this thesis is dedicated to the *relaxed correlation clustering* (RCC) [48], which is a signed graph clustering problem with applications in social network analysis. In this work, the RCC—an NP-Hard combinatorial optimization problem—is studied from a methodological point of view through intensive experimentation, where different exact and heuristic approaches are investigated.

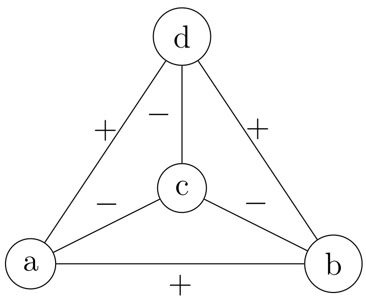


Figure 1.1: A signed graph.

The remainder of this thesis is related to another very important class of problems: vehicle routing problems (VRPs). VRPs are combinatorial optimization problems linked to a critical issue faced by the transportation industry: What are the best routes for a fleet of vehicles visiting a set of customers? Figure 1.2 illustrates a VRP solution via a directed graph, where a fleet of three vehicles leaves the depot (vertex 0) to visit the nine customers, and then returns to the same depot. This work approaches two VRP variants:

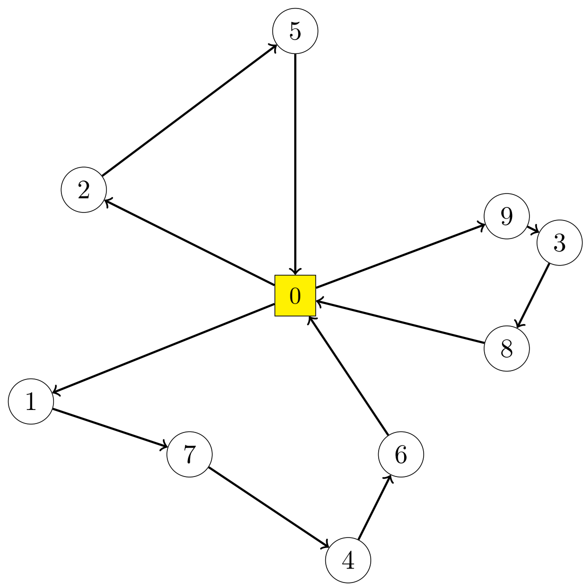


Figure 1.2: A VRP solution with three routes.

the VRP with backhauls (VRPB) and the capacitated VRP (CVRP). For the VRPB—a well-known VRP variant that considers two types of customers—, exact and heuristic algorithms are investigated, whereas a decomposition-based heuristic (that explores the re-optimization of parts of a solution) is proposed to tackle the CVRP—the most widely studied VRP variant—and its extensions.

In fact, this thesis is a compilation of three publications [97, 98, 105] and a submitted manuscript (available as a technical report in Queiroga, Sadykov, and Uchoa [96]), so the chapters are almost self-contained, and the reader would have no difficulty reading them independently.

1.1 Objectives

The objectives of this work are as follows.

- Develop new mathematical formulations and an effective local-search based heuristic for the RCC.
- Propose exact algorithms for the VRPB which are capable of solving all the literature instances.
- Generate a novel and more challenging benchmark instances for the VRPB.
- Study different heuristic strategies for the VRPB, and the benefits of considering problem-specific information during their executions.

- Develop a decomposition-based heuristic for the CVRP (and its extensions) for medium and long runs.

1.2 Thesis outline

The remainder of this work is organized as follows.

- Chapter 2 presents the new formulations and local-search based heuristic for the RCC.
- Chapter 3 presents the proposed exact and heuristic approaches for the VRPB.
- Chapter 4 presents the proposed decomposition-based heuristic for the CVRP and its extensions.
- Chapter 5 contains the concluding remarks of this thesis.

Chapter 2

Relaxed Correlation Clustering

2.1 Introduction

In graph theory, signed graphs are those where each arc (or edge) has a positive or negative sign [128, 129]. This type of graph has been extensively used for modeling problems in various fields including biology [35], economy [57, 118], chemistry [79], ecology [33], image segmentation [64], linguistics [116], but mainly in social network analysis [3, 5, 20, 39, 41, 44]. One of these problems is *correlation clustering* (CC) [11], which is a well-known unsupervised learning problem that aims at finding a vertex partitioning in a signed graph so as to minimize the disagreements, given by negative arcs (or edges) within a cluster and positive arcs (or edges) between clusters. Before formally defining the CC problem on a directed signed graph, we shall introduce some notation.

- Let $G = (V, A, s)$ be a signed digraph, where $V = \{1, 2, \ldots, n\}$ is the vertex set, $A \subseteq V \times V$ is the arc set, and $s : A \to \{+, -\}$ is a function that assigns a sign to each arc.
- An arc $a \in A$ is called *negative* if $s(a) = -$ and *positive* if $s(a) = +$.
- For each arc $a \in A$, let w_a be an associated non-negative weight. We will also use w_{ij} and w_{ji} to denote the weight of the arcs (i, j) and (j, i), respectively.
- The set of positive and negative arcs are denoted, respectively, as A^+ and A^-; thus $A = A^- \cup A^+$.
- A partition of V into l disjoint subsets $P = \{S_1, S_2, \ldots, S_l\}$ is called a l-partition of V.

- For $1 \leq p, q \leq l$, let $A[S_p : S_q] = \{(i,j) \in A \mid i \in S_p, j \in S_q\}$.
- $\Omega^+(S_p, S_q) = \sum_{a \in A^+ \cap A[S_p:S_q]} w_a$ and $\Omega^-(S_p, S_q) = \sum_{a \in A^- \cap A[S_p:S_q]} w_a$.

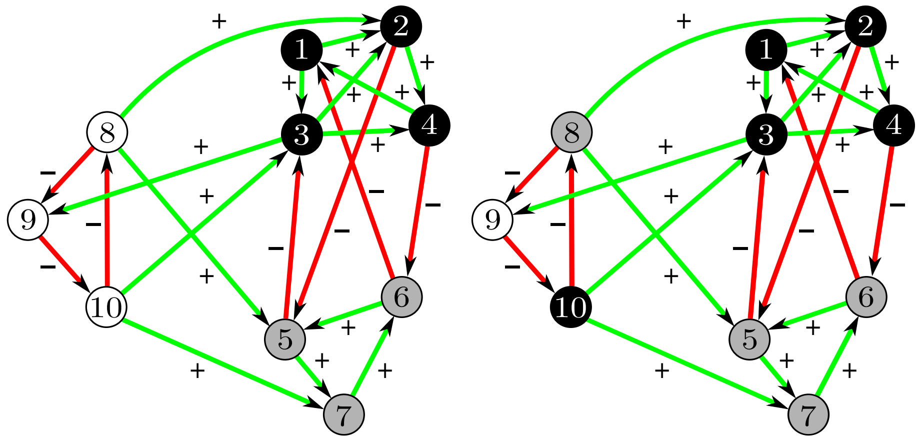
The *imbalance* $I(P)$ of a l-partition P is defined as

$$I(P) = \sum_{1 \leq p \leq l} \Omega^-(S_p, S_p) + \sum_{\substack{1 \leq p \leq l, \\ 1 \leq q \leq l, \\ p \neq q}} \Omega^+(S_p, S_q) \tag{2.1}$$

The CC problem consists of determining a partition P which minimizes $I(P)$. One of the applications of CC is related to the analysis of structural balance on social networks. In such networks, the arcs represent social relations between actors (i.e. the vertices of the network), whereas the sign represents feelings such as like/dislike and agreement/disagreement. According to the structural balance theory of Heider [26, 58], a network is *balanced* if there is a bipartition of the vertex set so that every positive arc joins actors in a same group and every negative arc joins actors in different groups. Later, Davis [36] generalized the structural balance to support a partition with more than two groups, which is fully compatible with the criterion optimized by the CC. Indeed, an algorithm for CC is a useful tool for evaluating how balanced a social network is. Note that a signed graph is balanced if there is a partition P such that $I(P) = 0$.

Although traditional structural balance works in several scenarios, Doreian and Mrvar [40] pointed out that this concept could not be appropriate for some networks. They argued that Equation (2.1) penalizes patterns in the partitions associated with relevant social psychological processes. For example, the network in Figure 2.1a represents a scenario with three groups, where one of them is a group of mutually hostile mediators (vertices 8, 9, and 10). Figure 2.1b illustrates why CC is not suitable in this case: it is not capable of detecting the group of mediators or any type of subgroup internal hostility. This was illustrated in practice by Levorato et al. [76], where positive/negative mediation was detected in networks describing the United Nations General Assembly Voting Data. The equivalent was also verified for the case of differential popularity (a process in which some actors receive more positive links than others in a group) detected in benchmark instances from the literature [42], and for internal hostility detected in networks describing voting activity of members of the European Parliament [5].

Still in [40], Doreian and Mrvar introduced the concept of generalized structural balance giving rise to a new definition for the imbalance of a vertex partition which



(a) Ideal partition (b) CC partition

Figure 2.1: Structural balance in a network with mutually hostile mediators.

corrects the partition patterns penalized in Equation (2.1). The *relaxed imbalance* of a l-partition partition P, denoted here $RI(P)$, is defined as

$$RI(P) = \sum_{1 \le p \le l} \min\{\Omega^+(S_p, S_p), \Omega^-(S_p, S_p)\} + \sum_{\substack{1 \le p \le l, \\ 1 \le q \le l, \\ p \ne q}} \min\{\Omega^+(S_p, S_q), \Omega^-(S_p, S_q)\}. \tag{2.2}$$

The main subject of this chapter is the RCC problem [40, 48], which is a relaxed version of the CC. In the RCC, given an integer parameter $1 \le k \le n$, one aims at finding a partition $P \in \cup_{l=1}^{k} \mathcal{P}^l$ that minimizes the *relaxed imbalance* given by Equation (2.2), such that $\mathcal{P}^l$ is the set of all l-partitions of V. The optimal value of the RCC problem determines how balanced a network is w.r.t. the relaxed structural balance introduced by Doreian and Mrvar[40]. For example, the partition in Figure 2.1a is an optimal solution for RCC because $RI(P) = 0$. Both CC and RCC problems were proven NP-hard by Bansal, Blum, and Chawla [11] and Figueiredo and Moura [48], respectively.

Doreian and Mrvar [40] tackled the RCC by applying a relocation algorithm to analyze four real data sets related to relaxed structural balance, with up to 20 vertices. The authors adapted a heuristic method proposed in Doreian and Mrvar [39] for the CC problem with a fixed number of clusters, which optimizes a generic and parameterized function called *criterion function*. Later, Brusco et al. [19] proposed a branch-and-bound algorithm for solving RCC to optimality. This algorithm was capable of solving instances with up to 29 vertices (hereafter referred to as small-sized instances) and k varying from 2 to 7. Moreover, an additional set of instances with up to 40 vertices was considered for experiments with $k = \{3, 5\}$. Figueiredo and Moura [48] developed an *integer linear programming* (ILP) formulation that was capable of solving some small-sized instances when $k = \{2, 3\}$ and for high values of k, concluding that the proposed ILP formulation and the existing branch-and-bound algorithm are somewhat complementary approaches. The authors also proposed a symmetrical version of RCC (SRCC). Although different heuristic procedures were proposed in the literature for CC (see [13, 39, 76, 123, 125], among others), to the best of our knowledge, only one metaheuristic based procedure has been applied to the RCC. Levorato et al. [76] adapted their iterated local search (ILS) algorithm, originally developed for CC, to solve SRCC. Thus, there are no heuristics specifically proposed for RCC problem.

The two main contributions can be summarized as follows:

- We present two novel integer programming formulations for the RCC problem and we investigate their empirical performance in comparison to an existing formulation. The results show that the new formulations appear to produce better results in practice.
- We propose a local search-based heuristic that relies on a series of auxiliary data structures to efficiently recalculate the relaxed imbalance after applying an operation that modifies a partition, as well as on a novel perturbation mechanism. The results obtained suggest that the developed algorithm is superior to an existing approach, producing, on average, high quality solutions in a limited amount of CPU time, not only for RCC instances but also for SRCC instances. Finally, we also demonstrate the practical benefits of implementing the move evaluation in an efficient way.

The remainder of this chapter is organized as follows. Section 2.2 presents a small instance for the RCC problem. Section 2.3 defines the symmetric version of the problem. Section 2.4 introduces two novel mathematical formulations for the RCC. Section 2.5 explains the proposed efficient local search-based heuristic including the efficient move evaluation schemes. Section 2.6 presents the results of the computational experiments. Finally, the conclusions are discussed in Section 2.7.

2.2 A small RCC example

A small RCC example involving 6 vertices is depicted in Figure 2.2. In Figure 2.2(a), the signed graph to be partitioned is illustrated, and we consider all weights equal to one. In

Figure 2.2(b), a feasible solution $P = \{S_1 = \{1,2\}, S_2 = \{3,4\}, S_3 = \{5,6\}\}$ is presented. This solution has relaxed imbalance $RI(P) = 4$ obtained by adding the following terms.

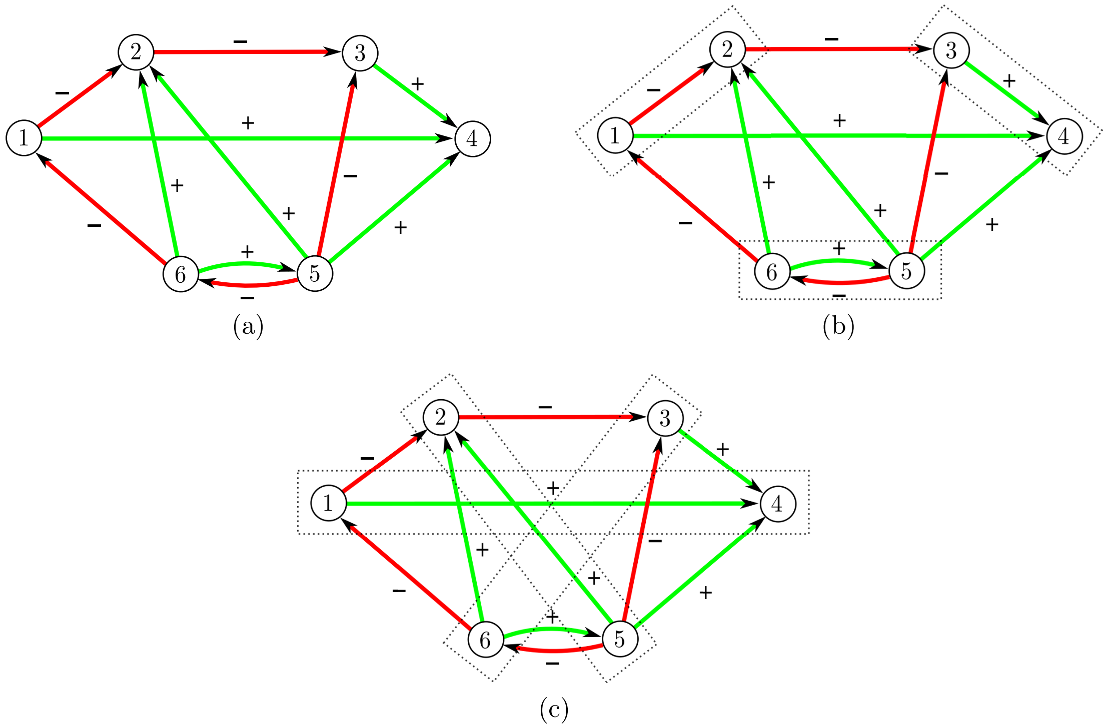


Figure 2.2: (a) A small RCC instance with unitary weights and $k = 3$. (b) A feasible solution $P = \{\{1,2\},\{3,4\},\{5,6\}\}$ with relaxed imbalance $RI(P) = 4$. (c) An optimal solution $P^* = \{\{1,4\},\{2,5\},\{3,6\}\}$ with relaxed imbalance $RI(P) = 1$.

- $\min\{\Omega^+(S_1,S_1), \Omega^-(S_1,S_1)\} = \min\{\emptyset, w_{12}\} = \min\{0,1\} = 0$
- $\min\{\Omega^+(S_2,S_2), \Omega^-(S_2,S_2)\} = \min\{w_{34}, \emptyset\} = \min\{1,0\} = 0$
- $\min\{\Omega^+(S_3,S_3), \Omega^-(S_3,S_3)\} = \min\{w_{65}, w_{56}\} = \min\{1,1\} = 1$
- $\min\{\Omega^+(S_1,S_2), \Omega^-(S_1,S_2)\} = \min\{w_{14}, w_{23}\} = \min\{1,1\} = 1$
- $\min\{\Omega^+(S_2,S_1), \Omega^-(S_2,S_1)\} = \min\{\emptyset, \emptyset\} = \min\{0,0\} = 0$
- $\min\{\Omega^+(S_1,S_3), \Omega^-(S_1,S_3)\} = \min\{\emptyset, \emptyset\} = \min\{0,0\} = 0$
- $\min\{\Omega^+(S_3,S_1), \Omega^-(S_3,S_1)\} = \min\{w_{52} + w_{62}, w_{61}\} = \min\{2,1\} = 1$
- $\min\{\Omega^+(S_2,S_3), \Omega^-(S_2,S_3)\} = \min\{\emptyset, \emptyset\} = \min\{0,0\} = 0$
- $\min\{\Omega^+(S_3,S_2), \Omega^-(S_3,S_2)\} = \min\{w_{53}, w_{54}\} = \min\{1,1\} = 1$

Figure 2.2(c) depicts an optimal solution $P^* = \{S_1 = \{1,4\}, S_2 = \{2,5\}, S_3 = \{3,6\}\}$ for the problem with $RI(P^*) = \min\{\Omega^+(S_3, S_1), \Omega^-(S_3, S_1)\} = \min\{w_{34}, w_{61}\} = \min\{1,1\} = 1$.

2.3 Symmetric RCC

In this work, we also consider the symmetric version of RCC (SRCC) introduced in Figueiredo and Moura [48]. The relaxed imbalance, as given by Equation (2.2), penalizes non-predominant relations (w.r.t. the signs) inside each cluster q and non-predominant relations *from* a cluster p *to* a cluster q. The difference in the symmetric relaxed imbalance defined in Figueiredo and Moura [48] is that it penalizes non-predominant relations *among* pairs of clusters, i.e., it considers simultaneously all positive (all negative) relations from S_p to S_q and from S_q to S_p. Thus, the SRCC can be defined on an undirected graph in which parallel edges with opposite signs are allowed.

Let $G' = (V, E, s')$ be an undirected signed graph with a positive weight w'_{ij} associated to each edge $\{i, j\} \in E$. Let us denote E^+ and E^-, respectively, the sets of positive and negative edges in E; thus $E = E^+ \cup E^-$. In this work, we transform the SRCC instance defined on $G' = (V, E, s')$ into a RCC instance defined on a directed signed graph $G_d = (V, A, s)$ in which: $A = \{(i, j), (j, i) : \{i, j\} \in E\}$; for each $(i, j) \in A$, $s((i, j)) = s((j, i)) = s'(\{i, j\})$ and the associated weight $w_{ij} = w_{ji} = \frac{w'_{ij}}{2}$. In other words, for each edge $\{i, j\} \in E$, one creates two arcs $(i, j), (j, i) \in A$ with the same signal and half of the weight.

Let $E[S_p : S_q]$ be the set of edges connecting the vertices in S_p and those in S_q. We denote $\Omega'^+(S_p, S_q) = \sum_{e \in E^+ \cap E[S_p:S_q]} w'_e$ and $\Omega'^-(S_p, S_q) = \sum_{e \in E^- \cap E[S_p:S_q]} w'_e$. The symmetric relaxed imbalance $SRI(P)$ of a l-partition P is defined as

$$SRI(P) = \sum_{1 \le p \le l} \min\{\Omega'^+(S_p, S_p), \Omega'^-(S_p, S_p)\} + \sum_{1 \le p < q \le l} \min\{\Omega'^+(S_p, S_q), \Omega'^-(S_p, S_q)\} \tag{2.3}$$

The following result relates Equations (2.2) and (2.3).

Proposition 1. *Consider an undirected signed graph G' and the directed signed graph G_d described above. Given any partitioning P, then $SRI(P) = RI(P)$.*

Proof. First, we will show the equivalence of the first terms in (2.2) and (2.3), i.e. the intracluster imbalance. Then we will do the same for the second terms, i.e. for the intercluster imbalance. Let S_p be any cluster in P. We have $\Omega'^+(S_p, S_p) = \Omega^+(S_p, S_p)$

since, for each $e = \{i, j\} \in E^+ \cap E[S_p : S_p]$, $(i, j), (j, i) \in A^+ \cap A[S_p : S_p]$ with $w'_{ij} = w_{ij} + w_{ji}$. The same can be argued for $\Omega'^-(S_p, S_p)$ and these two facts imply the equivalence of the first terms in (2.2) and (2.3). Now, let S_p and S_q be two clusters in P. By the definition of the directed graph G_d, we have $e = \{i, j\} \in E^+ \cap E[S_p : S_q]$ iff $(i, j) \in A^+ \cap A[S_p : S_q]$ and $(j, i) \in A^+ \cap A[S_q : S_p]$. Since, for each $e = \{i, j\} \in E$, $w'_{ij} = w_{ij} + w_{ji}$, we have that $\Omega^+(S_p, S_q) = \Omega^+(S_q, S_p) = \Omega'^+(S_p, S_q)/2$. The same holds for $\Omega^-(S_p, S_q)$ and, since the second term in (2.3) is written only for $p < q$, the equivalence of the second terms in (2.2) and (2.3) follows. $\square$

As a consequence of Proposition 1, solving the RCC over graph G_d is equivalent to solving SRCC over G'.

2.4 Mathematical formulations

Integer linear programming (ILP) problem formulations have been used to solve CC and other related problems defined on signed graphs [8, 24, 25, 47, 50]. For RCC, an ILP formulation was presented in [48]. When modeling vertex-clustering problems, if there is a need for keeping track the clusters used, two types of formulations can be adopted: cluster-indexed formulation [15, 21, 45] or representatives formulation [2, 8, 25, 50]. Indeed, the ILP formulation of Figueiredo and Moura [48] is a representatives one. Next, we introduce two new formulations for RCC, one of each type.

2.4.1 Formulation `F1`: a cluster-indexed formulation

Let $K = \{1, \ldots, k\}$ be the set of possible cluster indexes. For each vertex $i \in V$ and $p \in K$ we define,

$$x_i^p = \begin{cases} 1, & \text{if vertex } i \text{ belongs to cluster } S_p, \\ 0, & \text{otherwise.} \end{cases}$$

A set of binary variables is used to describe the set of arcs that will be penalized once the predominant relations are defined (according to Equation (2.2)). For each arc $(i, j) \in A$, we define,

$$t_{ij} = \begin{cases} 1, & \text{if arc } (i, j) \text{ is penalized,} \\ 0, & \text{otherwise.} \end{cases}$$

For example, if the predominant sign in a cluster S_p is $-$ (i.e. $\Omega^-(S_p, S_p) > \Omega^+(S_p, S_p)$), the variables associated to the arcs $(i, j) \in A^+ \cap A[S_p : S_p]$ should be penalized ($t_{ij} = 1$).

A set of binary variables is used to select if the imbalance from cluster S_p to cluster S_q, with $p, q \in K$, is given by negative arcs (predominant relations are positive) or positive arcs (predominant relations are negative). Notice that intracluster imbalance is defined whenever $p = q$. For each pair of cluster indexes $p, q \in K$, we define,

$$s_{pq} = \begin{cases} 1, & \text{if positive arcs from cluster } S_p \text{ to cluster } S_q \text{ are penalized,} \\ 0, & \text{if negative arcs from cluster } S_p \text{ to cluster } S_q \text{ are penalized.} \end{cases}$$

The formulation can be written as

$$\text{minimize} \sum_{(i,j)\in A} w_{ij} t_{ij} \tag{2.4}$$

$$\text{s.t:} \sum_{p\in K} x_i^p = 1, \qquad \forall i \in V, \tag{2.5}$$

$$t_{ij} \geq x_i^p + x_j^q - 2 + s_{pq}, \qquad \forall (i,j) \in A^+, \forall p, q \in K, \tag{2.6}$$

$$t_{ij} \geq x_i^p + x_j^q - 2 + (1 - s_{pq}), \qquad \forall (i,j) \in A^-, \forall p, q \in K, \tag{2.7}$$

$$x_i^p \in \{0, 1\}, \qquad \forall i \in V, \forall p \in K, \tag{2.8}$$

$$t_{ij} \in \{0, 1\}, \qquad \forall (i,j) \in A, \tag{2.9}$$

$$s_{pq} \in \{0, 1\}, \qquad \forall p, q \in K. \tag{2.10}$$

Objective function (2.4) minimizes the total relaxed imbalance (i.e. the sum of penalties). Constraints (2.5) ensure that each vertex is assigned exactly to one cluster. Constraints (2.6) and (2.7) define, respectively, if a positive or negative arc will be penalized due to assignment variables x_i^p, x_j^q and the penalizing variables s_{pq}. For example, if $p = q$ (intracluster case), $s_{pp} = 1$ (positive arcs are penalized), $x_i^p = 1$, and $x_j^q = 1$ such that $(i, j) \in A^+ \cap A[S_p : S_p]$, then the right hand side (RHS) of (2.6) will be $x_i^p + x_j^q - 2 + s_{pq} = 1 + 1 - 2 + 1 = 1$, which leads $t_{ij} = 1$. On the other hand, notice that constraints (2.7) will not force $t_{ij} = 1$ if $(i, j) \in A^- \cap A[S_p : S_p]$ because

$(1 - s_{pq}) = 1 - 1 = 0$ forbids the RHS to be greater than 0; hence $t_{ij} = 0$ because the objective function will not “allow" penalized arcs which are not forced by constraint. Finaly, constraints (2.8)–(2.10) define the domain of all variables.

Formulations that make use of cluster-indexed variables such as F1 are considered to be symmetric, as there are a substantial number of ways to represent the same partitioning with a different permutation of indices. In view of this, we add the following set of symmetry breaking inequalities introduced in Bulhões et al. [21] for the p-cluster editing problem:

$$\sum_{l=1}^{p} x_i^l \geq \left(\sum_{\substack{j \in V \\ j < i}} \sum_{l=1}^{p-1} x_j^l \right) - (i-2), \ \forall i \in V, \ \forall p \in K. \tag{2.11}$$

The inequality above forbids the cluster containing i to use a label superior to p whenever each vertex $j < i$ is assigned to a cluster of index strictly smaller than p. For example, if $i = 5$ and $x_1^2, x_2^1, x_3^1, x_4^2 = 1$ (vertices 1,2,3, and 4 use at most cluster index 2), then the cluster having the vertex 5 can use at most the index 3. The formulation in the next section adopts a similar strategy in order to break symmetry in the solution space.

2.4.2 Formulation F2: a representatives formulation

A representatives formulation for the RCC is presented as follows. The idea behind this kind of formulation [25] is the unique representation of a cluster by its vertex with the lowest label. Hence, for each pair of vertices $i, j \in V$ satisfying $i \leq j$, we define,

$$x_j^i = \begin{cases} 1, & \text{if the vertex } j \text{ is represented by vertex } i, \\ 0, & \text{otherwise.} \end{cases}$$

Note that when $i = j$, variable x_i^i indicates if i is a representative vertex.

Variables s_{ij}, with $i, j \in V$, used in F2 are equivalent to variables s_{pq}, with $p, q \in K$, used in F1. However now, vertices i and j are used to identify two clusters ($i \neq j$) or one cluster ($i = j$). Variables t_{ij}, with $(i, j) \in A$, are exactly the same as defined in F1. Hence, formulation F2 can be expressed as follows.

$$
\begin{aligned}
\text{minimize} \; & \sum_{(i,j)\in A} w_{ij} t_{ij} && && \\
\text{s.t:} \; & \sum_{i\in V:\; i\leq j} x_j^i = 1, && \forall j \in V, && (2.12)\\
& x_j^i \leq x_i^i, && \forall i,j \in V, i < j, && (2.13)\\
& \sum_{i\in V} x_i^i \leq k, && && (2.14)\\
& t_{ij} \geq x_i^u + x_j^v - 2 + s_{uv}, && \forall (i,j) \in A^+, \forall u,v \in V, && (2.15)\\
& && u \leq i, v \leq j, && \\
& t_{ij} \geq x_i^u + x_j^v - 2 + (1 - s_{uv}), && \forall (i,j) \in A^-, \forall u,v \in V, && (2.16)\\
& && u \leq i, v \leq j, && \\
& x_j^i \in \{0,1\}, && \forall i,j \in V, i \leq j, && (2.17)\\
& t_{ij} \in \{0,1\}, && \forall (i,j) \in A, && (2.18)\\
& s_{ij} \in \{0,1\}, && \forall i,j \in V. && (2.19)
\end{aligned}
$$

Constraints (2.12) impose that each vertex must be represented by exactly one vertex: either by itself or by another one with a smaller index. Constraints (2.13) enforce vertex i to be a representative one whenever a vertex j is represented by i. Constraint (2.14) imposes k as an upper bound on the number of representative vertices, i.e., on the number of clusters in the partition. Constraints (2.15) and (2.16) are, respectively, equivalent to constraints (2.6) and (2.7) of formulation `F1`. Finally, constraints (2.17)–(2.19) are the binary constraints.

The number of variables and constraints of formulations `F1` and `F2` are illustrated in Table 2.1. Note that since the number of vertices of a graph is usually much greater than the number of clusters, formulation `F1` is more compact than `F2`. On the other hand, formulation `F2` succeeds in eliminating cluster indices from the representation which breaks symmetry from formulation [25].

Table 2.1: Number of variables and constraints of formulations F1 and F2

	#Variables	#Constraints
F1	$\mathcal{O}(nk + \lvert A\rvert + k^2)$	$\mathcal{O}(n + k^2\lvert A^+\rvert + k^2\lvert A^-\rvert)$
F2	$\mathcal{O}(2n^2 + \lvert A\rvert)$	$\mathcal{O}(n + n^2 + n^2\lvert A^+\rvert + n^2\lvert A^-\rvert)$

2.5 Proposed local search-based heuristic

In this section, we propose a heuristic algorithm for the RCC based on the iterated local search (ILS) method [77]. The key concept of ILS is to combine local search strategies and perturbation mechanisms to escape from local optima. The heuristic introduced in Levorato et al. [76] for CC and adapted for the solution of SRCC instances is also an ILS method. Different from Levorato et al. [76], in this work, we propose the use of several complex neighborhoods and the use of advanced data structures which make the search process more efficient.

Algorithm 1 presents a general framework of a multi-start ILS, hereafter referred to as ILS_{RCC}, which has the following input parameters:

a) I_R is the number of restarts of the heuristic;

b) I_{ILS} is the maximum number of ILS iterations without improvements;

c) I_P is the maximum number of moves performed by a perturbation mechanism.

For each restart, an initial solution is randomly generated (line 5) and such solution is possibly improved by alternately applying local search (line 9) and perturbation (line 13) strategies until the maximum number of iterations (I_{ILS}) without improvement is achieved. Finally, the best solution found among all restarts is returned (line 18).

The local search procedure is based on variable neighborhood descent (VND) [83], which is a technique that systematically explores a sequence of neighborhood structures (see Section 2.5.2), searching for better solutions. A neighborhood structure (or simply neighborhood) defines a set of neighbor solutions from a current solution by applying a so-called move. When VND finds an improving move using a particular neighborhood, the solution is updated and the procedure restarts from the improved solution. The procedure terminates when all neighborhoods fail to improve the current solution. The *best improvement* strategy was adopted, i.e., a neighborhood is fully enumerated and the best improving move (if there is any) is applied. In addition, the neighborhood ordering is defined in a random fashion, which results in a strategy known as Randomized VND

Algorithm 1: ILS_{RCC}

```
Procedure ILS_RCC(I_R, I_ILS, I_P)
    RI* = ∞
    P* = ∅
    for iter = 1 ... I_R do
        P = ConstructiveProcedure()
        P' = P
        iter_ILS = 0
        while iter_ILS < I_ILS do
            P = LocalSearch(P)
            if RI(P) < RI(P') then
                P' = P
                iter_ILS = 0
            P = Perturb(P', I_P)
            iter_ILS = iter_ILS + 1
        if RI(P') < RI* then
            P* = P'
            RI* = RI(P')
    return P*
```

(RVND). The combination of ILS and RVND led to state-of-the-art methods for several important combinatorial optimization problems, such as: split-delivery vehicle routing problem [102], minimum latency problem [103] and minimizing weighted tardiness in single machine scheduling with sequence-dependent setup times [104].

The perturbation procedure randomly chooses one of the implemented mechanisms (see Section 2.5.3) in order to modify the local optimal solution P'. The selected mechanism then applies I_P random consecutive moves over P' in order to generate a solution to continue the search.

In what follows, we provide a detailed description of the auxiliary data structures used for performing efficient move evaluation, as well as on the neighborhood structures and perturbations mechanisms.

2.5.1 Auxiliary data structures

Assuming that an adjacency matrix is used to access the signed digraph G, and a feasible solution is represented by a set of subsets of indices (*e.g.* $P = \{S_1, S_2, S_3\}$, such that $S_1 = \{1, 2\}, S_2 = \{3, 4\}, S_3 = \{5, 6\}$ for a graph with 6 vertices; see Figure 2.2), the value of its associated objective function can be straightforwardly computed in $\mathcal{O}(l^2 n^2)$ operations, where $l = |P|$. Note that this is due to the complexity of determining the intercluster imbalance. Consequently, performing the move evaluation of a neighbor solution from scratch every time during the local search may turn out to be computationally expensive, especially for large size instances. However, this can be done in a more efficient manner by precomputing and storing information in auxiliary data structures (ADSs).

We thus propose to implement two classes of ADSs: $\texttt{SumIntra}[S_p]$, which is a set of ADSs that stores the sum of the weights for different subsets of $A[S_p]$ (a.k.a. intracluster arcs of S_p); and $\texttt{SumInter}[S_p][S_q]$, which is a set of ADSs that stores the sum of the weights for different subsets of $A[S_p : S_q]$ (a.k.a. known as intercluster arcs from S_p to S_q). The ADSs are divided according to the sign and arc direction as described in Table 2.2.

Table 2.2: Description of the proposed ADSs

ADS	Description
$\texttt{SumIntra}^+[S_p] = \sum_{a \in A^+ \cap A[S_p]} w_a$	Sum of positive weights within S_p
$\texttt{SumIntra}^-[S_p] = \sum_{a \in A^- \cap A[S_p]} w_a$	Sum of negative weights within S_p
$\texttt{SumIntra}^+[S_p][i][\leftarrow] = \sum_{ji \in A^+, j \in S_p \setminus i} w_{ji}$	Sum of positive weights from $S_p \setminus i$ to $i \in S_p$
$\texttt{SumIntra}^+[S_p][i][\rightarrow] = \sum_{ij \in A^+, j \in S_p \setminus i} w_{ij}$	Sum of positive weights from $i \in S_p$ to $S_p \setminus i$
$\texttt{SumIntra}^-[S_p][i][\leftarrow] = \sum_{ji \in A^-, j \in S_p \setminus i} w_{ji}$	Sum of negative weights from $S_p \setminus i$ to $i \in S_p$
$\texttt{SumIntra}^-[S_p][i][\rightarrow] = \sum_{ij \in A^-, j \in S_p \setminus i} w_{ij}$	Sum of negative weights from $i \in S_p$ to $S_p \setminus i$
$\texttt{SumInter}^+[S_p][S_q] = \sum_{a \in A^+ \cap A[S_p:S_q]} w_a$	Sum of positive weights from S_p to S_q
$\texttt{SumInter}^-[S_p][S_q] = \sum_{a \in A^- \cap A[S_p:S_q]} w_a$	Sum of negative weights from S_p to S_q
$\texttt{SumInter}^+[S_p][i][S_q][\rightarrow] = \sum_{ij \in A^+, j \in S_q} w_{ij}$	Sum of positive weights from $i \in S_p$ to S_q
$\texttt{SumInter}^+[S_p][i][S_q][\leftarrow] = \sum_{ji \in A^+, j \in S_q} w_{ji}$	Sum of positive weights from S_q to $i \in S_p$
$\texttt{SumInter}^-[S_p][i][S_q][\rightarrow] = \sum_{ij \in A^-, j \in S_q} w_{ij}$	Sum of negative weights from $i \in S_p$ to S_q
$\texttt{SumInter}^-[S_p][i][S_q][\leftarrow] = \sum_{ji \in A^-, j \in S_q} w_{ji}$	Sum of negative weights from S_q to $i \in S_p$

Given a feasible solution, the `SumIntra` and `SumInter` ADSs can be initially built in $\mathcal{O}(l^2 n^2)$ operations as described in Algorithm 2.

2.5.2 Neighborhood structures

ILS_{RCC} uses three neighborhood structures in the local search, namely: *Insertion*, *Swap* and *Split*. In the following, each of them is described in detail.

Insertion The *Insertion* neighborhood moves a vertex from a cluster to another one, thus yielding $\mathcal{O}(l^2 n)$ possible neighbor solutions to be evaluated.

Algorithm 3 describes how an *Insertion* move is evaluated using the ADSs. This algorithm receives as input the solution P along with its associated cost (relaxed imbalance) RI_P, and the information regarding the move, i.e. S_p, $i \in S_p$ and S_q. At first,

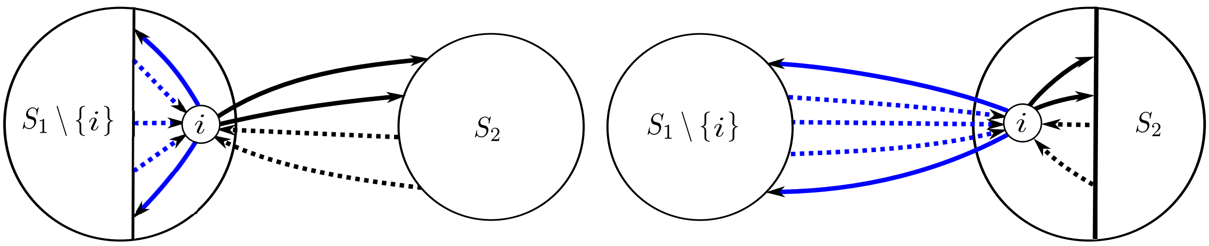
Algorithm 2: Computing the auxiliary data structures

```
Algorithm ComputeADSs(P)
    All ADSs are initialized with 0.0

    ▷ Computing the SumIntra ADSs
    for ∀p ∈ {1, 2, ..., l} do
        for ∀i, j ∈ S_p, i < j do
            if (i, j) ∈ A^+ then
                SumIntra^+[S_p] = SumIntra^+[S_p] + w_ij
                SumIntra^+[S_p][i][→] = SumIntra^+[S_p][i][→] + w_ij
                SumIntra^+[S_p][j][←] = SumIntra^+[S_p][j][←] + w_ij
            else if (i, j) ∈ A^- then
                SumIntra^-[S_p] = SumIntra^-[S_p] + w_ij
                SumIntra^-[S_p][i][→] = SumIntra^-[S_p][i][→] + w_ij
                SumIntra^-[S_p][j][←] = SumIntra^-[S_p][j][←] + w_ij
            if (j, i) ∈ A^+ then
                SumIntra^+[S_p] = SumIntra^+[S_p] + w_ji
                SumIntra^+[S_p][j][→] = SumIntra^+[S_p][j][→] + w_ji
                SumIntra^+[S_p][i][←] = SumIntra^+[S_p][i][←] + w_ji
            else if (j, i) ∈ A^- then
                SumIntra^-[S_p] = SumIntra^-[S_p] + w_ji
                SumIntra^-[S_p][j][→] = SumIntra^-[S_p][j][→] + w_ji
                SumIntra^-[S_p][i][←] = SumIntra^-[S_p][i][←] + w_ji

    ▷ Computing the SumInter ADSs
    for ∀p, q ∈ {1, 2, ..., l}, p ≠ q do
        for ∀i ∈ S_p, ∀j ∈ S_q do
            if (i, j) ∈ A^+ then
                SumInter^+[S_p][S_q] = SumInter^+[S_p][S_q] + w_ij
                SumInter^+[S_p][i][S_q][→] = SumInter^+[S_p][i][S_q][→] + w_ij
                SumInter^+[S_q][j][S_p][←] = SumInter^+[S_q][j][S_p][←] + w_ij
            else if (i, j) ∈ A^- then
                SumInter^-[S_p][S_q] = SumInter^-[S_p][S_q] + w_ij
                SumInter^-[S_p][i][S_q][→] = SumInter^-[S_p][i][S_q][→] + w_ij
                SumInter^-[S_q][j][S_p][←] = SumInter^-[S_q][j][S_p][←] + w_ij
            if (j, i) ∈ A^+ then
                SumInter^+[S_q][S_p] = SumInter^+[S_q][S_p] + w_ji
                SumInter^+[S_q][j][S_p][→] = SumInter^+[S_q][j][S_p][→] + w_ji
                SumInter^+[S_p][i][S_q][←] = SumInter^+[S_p][i][S_q][←] + w_ji
            else if (j, i) ∈ A^- then
                SumInter^-[S_q][S_p] = SumInter^-[S_q][S_p] + w_ji
                SumInter^-[S_q][j][S_p][→] = SumInter^-[S_q][j][S_p][→] + w_ji
                SumInter^-[S_p][i][S_q][←] = SumInter^-[S_p][i][S_q][←] + w_ji
```

auxiliary variables $sum^+_{S_p}$, $sum^-_{S_p}$, $sum^+_{S_p,S_q}$ and $sum^-_{S_p,S_q}$ temporarily store, in $\mathcal{O}(1)$ steps, the sum of the weights associated to the move (lines 2–13). Next, the value of the objective function of the neighbor solution under evaluation, denoted in the algorithm as *cost*, is partially obtained (lines 14–17) by recomputing the penalty decisions using function `UpdateCost` (see lines 31–36). In the loop from lines 18 to 30, a similar procedure is performed for the intercluster cases involving the other clusters and the clusters S_p and S_q. Finally, *cost* is returned and the move yields an improvement if $cost < RI_P$.

Because Algorithm 3 performs $\mathcal{O}(l)$ steps (due to the loop), finding the best improving move requires $\mathcal{O}(l^3n)$ operations. Moreover, when P is modified, the ADSs must be updated. However, instead of recomputing the ADSs from scratch in $\mathcal{O}(l^2n^2)$ operations, one only needs to update the ADSs affected by the vertex that was involved in the move and this can be performed in $\mathcal{O}(n)$ steps, as detailed in the Appendix A.

Figure 2.3 illustrates an example of an *Insertion* move. Note that the separation of the weights for the adjacent arcs of vertex i in `SumIntra` clearly facilitates the evaluation of the intercluster sums from S_1 to S_2 and from S_2 to S_1. Otherwise, it would be necessary to perform $\mathcal{O}(n)$ operations to compute the weights separately.



(a) Clusters before *Insertion* (b) Clusters after *Insertion*

Figure 2.3: Example of an *Insertion* move. The incoming arcs of i are in dashed lines to illustrate the separation of the weights in `SumIntra` and `SumInter`. For the sake of simplicity, the signs were omitted and all arcs have unitary weight.

Swap

The *Swap* neighborhood exchanges two vertices between two different clusters, which leads to $\mathcal{O}(l^2n^2)$ neighbor solutions if one intends to enumerate all possibilities. The pseudocodes presented in the Appendix B describe how a *Swap* move can be evaluated in $\mathcal{O}(l)$ steps using a similar rationale employed in the *Insertion* neighborhood. Finding the best improving *Swap* move thus require $\mathcal{O}(l^3n^2)$ operations and the ADSs can be updated in $\mathcal{O}(n)$ steps as also described in the supplementary material.

Figure 2.4 depicts an example of a swap move, highlighting the arcs connecting

Algorithm 3: Using the ADSs to evaluate an insertion move

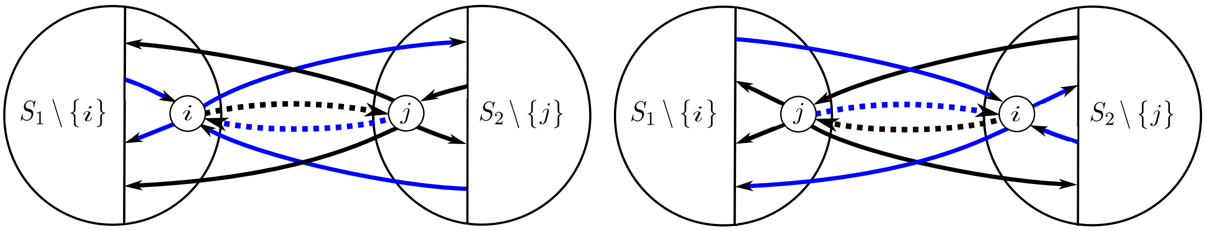
Algorithm $\texttt{CompCostInsert}(\boldsymbol{P, RI_P, S_p, i, S_q})$

    ▷ update the sum of the intracluster weights of S_p
    $sum^+_{S_p} = \texttt{SumIntra}^+[S_p] - \texttt{SumIntra}^+[S_p][i][\leftarrow] - \texttt{SumIntra}^+[S_p][i][\rightarrow]$
    $sum^-_{S_p} = \texttt{SumIntra}^-[S_p] - \texttt{SumIntra}^-[S_p][i][\leftarrow] - \texttt{SumIntra}^-[S_p][i][\rightarrow]$
    ▷ update the sum of the intracluster weights of S_q
    $sum^+_{S_q} = \texttt{SumIntra}^+[S_q] + \texttt{SumInter}^+[S_p][i][S_q][\leftarrow] + \texttt{SumInter}^+[S_p][i][S_q][\rightarrow]$
    $sum^-_{S_q} = \texttt{SumIntra}^-[S_q] + \texttt{SumInter}^-[S_p][i][S_q][\leftarrow] + \texttt{SumInter}^-[S_p][i][S_q][\rightarrow]$
    ▷ update the sum of the intercluster weights from S_p to S_q
    $sum^+_{S_p,S_q} = \texttt{SumInter}^+[S_p][S_q] - \texttt{SumInter}^+[S_p][i][S_q][\rightarrow]$
    $sum^-_{S_p,S_q} = \texttt{SumInter}^-[S_p][S_q] - \texttt{SumInter}^-[S_p][i][S_q][\rightarrow]$
    $sum^+_{S_p,S_q} = sum^+_{S_p,S_q} + \texttt{SumIntra}^+[S_p][i][\leftarrow]$
    $sum^-_{S_p,S_q} = sum^-_{S_p,S_q} + \texttt{SumIntra}^-[S_p][i][\leftarrow]$
    ▷ update the sum of the intercluster weights from S_q to S_p
    $sum^+_{S_q,S_p} = \texttt{SumInter}^+[S_q][S_p] - \texttt{SumInter}^+[S_p][i][S_q][\leftarrow]$
    $sum^-_{S_q,S_p} = \texttt{SumInter}^-[S_q][S_p] - \texttt{SumInter}^-[S_p][i][S_q][\leftarrow]$
    $sum^+_{S_q,S_p} = sum^+_{S_q,S_p} + \texttt{SumIntra}^+[S_p][i][\rightarrow]$
    $sum^-_{S_q,S_p} = sum^-_{S_q,S_p} + \texttt{SumIntra}^-[S_p][i][\rightarrow]$
    ▷ Recompute the penalty decisions and updates $RI_{P'}$
    $cost = \texttt{UpdateCost}(RI_P, S_p, S_p, sum^+_{S_p}, sum^-_{S_p})$
    $cost = \texttt{UpdateCost}(cost, S_q, S_q, sum^+_{S_q}, sum^-_{S_q})$
    $cost = \texttt{UpdateCost}(cost, S_p, S_q, sum^+_{S_p,S_q}, sum^-_{S_p,S_q})$
    $cost = \texttt{UpdateCost}(cost, S_q, S_p, sum^+_{S_q,S_p}, sum^-_{S_q,S_p})$
    ▷ update the sum of the intercluster weights involving others clusters
    for $S_r \in P \setminus \{S_p, S_q\}$ **do**
        $sum^+_{S_r,S_p} = \texttt{SumInter}^+[S_r][S_p] - \texttt{SumInter}^+[S_p][i][S_r][\leftarrow]$
        $sum^-_{S_r,S_p} = \texttt{SumInter}^-[S_r][S_p] - \texttt{SumInter}^-[S_p][i][S_r][\leftarrow]$
        $sum^+_{S_r,S_q} = \texttt{SumInter}^+[S_r][S_q] + \texttt{SumInter}^+[S_p][i][S_r][\leftarrow]$
        $sum^-_{S_r,S_q} = \texttt{SumInter}^-[S_r][S_q] + \texttt{SumInter}^-[S_p][i][S_r][\leftarrow]$
        $sum^+_{S_p,S_r} = \texttt{SumInter}^+[S_p][S_r] - \texttt{SumInter}^+[S_p][i][S_r][\rightarrow]$
        $sum^-_{S_p,S_r} = \texttt{SumInter}^-[S_p][S_r] - \texttt{SumInter}^-[S_p][i][S_r][\rightarrow]$
        $sum^+_{S_q,S_r} = \texttt{SumInter}^+[S_q][S_r] + \texttt{SumInter}^+[S_p][i][S_r][\rightarrow]$
        $sum^-_{S_q,S_r} = \texttt{SumInter}^-[S_q][S_r] + \texttt{SumInter}^-[S_p][i][S_r][\rightarrow]$
        $cost = \texttt{UpdateCost}(cost, S_r, S_p, sum^+_{S_r,S_p}, sum^-_{S_r,S_p})$
        $cost = \texttt{UpdateCost}(cost, S_r, S_q, sum^+_{S_r,S_q}, sum^-_{S_r,S_q})$
        $cost = \texttt{UpdateCost}(cost, S_p, S_r, sum^+_{S_p,S_r}, sum^-_{S_p,S_r})$
        $cost = \texttt{UpdateCost}(cost, S_q, S_r, sum^+_{S_q,S_r}, sum^-_{S_q,S_r})$
    return $cost$

Procedure $\texttt{UpdateCost}(\boldsymbol{cost, S_p, S_q, sum^+, sum^-})$

    if $S_p = S_q$ **then**
        return $cost - (\min\{\texttt{SumIntra}^+[S_p], \texttt{SumIntra}^-[S_p]\} - \min\{sum^+, sum^-\})$
    else
        return $cost - (\min\{\texttt{SumInter}^+[S_p][S_q], \texttt{SumInter}^-[S_p][S_q]\} - \min\{sum^+, sum^-\})$

exchanged vertices, as they must be treated separately with respect to some ADSs.

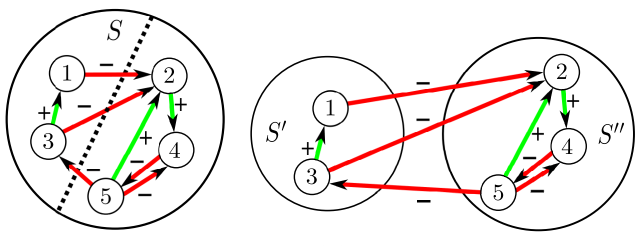


(a) Clusters before *Swap* (b) Clusters after *Swap*

Figure 2.4: Example of a *Swap* move. Arcs (i, j) and (j, i) are in dashed lines to illustrate their importance w.r.t. the ADSs. For the sake of simplicity, the signs were omitted and all arcs have unitary weight.

Split

The *Split* neighborhood splits a cluster into two, resulting in a total of $\mathcal{O}(ln)$ neighbor solutions to be examined. Formally, given a cluster $S = \{v_1, v_2, \ldots, v_{|S|}\} \in P$ and an index $c < |S|$, the clusters $S' = \{v_1, v_2, \ldots, v_c\}$ and $S'' = \{v_{c+1}, \ldots, v_{|S|}\}$ are produced to replace S in P. Clearly, a *Split* move can only be applied when $l < k$. The Pseudocodes presented in the Appendix C describes how a *Split* move can use previous evaluations to speedup the next ones. The overall complexity of determining the best improvement is $\mathcal{O}(ln^2)$, as also described in the supplementary material. Because of the considerable changes produced by the split move, all ADSs must be updated from scratch using Algorithm 2. It is worth mentioning that implementing a specific procedure to update the ADSs did not pay off the gains in CPU time. Moreover, note that a split move never worsens a solution, since the imbalance decreases monotonically as k increases [40]. Figure 2.5 shows an example of a split move considering a cluster with 5 vertices that is split into two with 2 and 3 vertices, respectively.



(a) Cluster before *Split* (b) Cluster after *Split*

Figure 2.5: Example of a *Split* move. For the sake of simplicity, all arcs have unitary weight.

Complexity summary

A summary on the complexity of the neighborhoods is provided in Table 2.3. For each neighborhood, we present its size, as well as the complexity of performing the move evaluation and the overall one using both the efficient best improvement (EBI) and the naive best improvement (NBI) strategies. In EBI, the search for the best improvement move is carried out as described in Section 2.5.2, whereas in NBI the objective function must be computed from scratch (with no support of ADSs) after each move. We also report the complexity of updating the ADSs in the case of EBI.

Table 2.3: Complexity summary of the neighborhoods considering both EBI and NBI strategies

Neighborhood	Size	EBI			NBI	
		Move eval.	Overall	Update	Move eval.	Overall
Insertion	$\mathcal{O}(l^2n)$	$\mathcal{O}(l)$	$\mathcal{O}(l^3n)$	$\mathcal{O}(n)$	$\mathcal{O}(l^2n^2)$	$\mathcal{O}(l^4n^3)$
Swap	$\mathcal{O}(l^2n^2)$	$\mathcal{O}(l)$	$\mathcal{O}(l^3n^2)$	$\mathcal{O}(n)$	$\mathcal{O}(l^2n^2)$	$\mathcal{O}(l^4n^4)$
Split	$\mathcal{O}(ln)$	$\mathcal{O}(n)$	$\mathcal{O}(ln^2)$	$\mathcal{O}(ln^2)$	$\mathcal{O}(l^2n^2)$	$\mathcal{O}(l^3n^3)$

2.5.3 Perturbation mechanisms

ILS$_{\text{RCC}}$ employs three diversification mechanisms to perturb local optimal solutions, namely: *Insertion*, *Merge* and *Sign inversion*. The first one simply performs random insertion moves. In the second, given two clusters S_1 and S_2 chosen at random, one merges them to form a new cluster S_3, that is, $S_3 = S_1 \cup S_2$. The latter perturbation is a novel procedure that considers some RCC specific features, as described in the following.

The proposed *Sign inversion* mechanism enforces the penalized sign in one of the decisions to be changed. More precisely, it modifies the solution in such a way that one of the intracluster or intercluster imbalances becomes defined by the opposite sign. The procedure randomly selects which case (i.e., intracluster or intercluster) is going to be considered. Basically, this is achieved by removing vertices that contribute with the non-penalized sign until the inversion happens. In what follows, we will explain the procedure used to invert an intracluster decision.

Let $+$ be the non-penalized sign for the intracluster imbalance of S_p (this also applies for sign $-$). Formally, the contribution of a vertex $i \in S_p$ is given by Equation (2.20).

$$\Delta^+(i) = \Omega^+(\{i\}, S_p) + \Omega^+(S_p, \{i\}) - \Omega^-(\{i\}, S_p) - \Omega^-(S_p, \{i\}) \tag{2.20}$$

At first, the vertices for which all incident arcs (indegree and outdegree arcs) are positive are removed in non-increasing order of Δ^+. The value of Δ^+ must be updated after each removal. If this does not suffice, the remaining vertices with $\Delta^+ > 0$ are removed using the same sorting criterion. Removals are performed while (i) $\Omega^+(S_p, S_p) \geq \Omega^-(S_p, S_p)$, (ii) there are vertices with $\Delta^+ > 0$ and (iii) $|S_p| > 2$. The removed vertices are randomly added to the other clusters. After applying the perturbation, if $\Omega^+(S_p, S_p) \geq \Omega^-(S_p, S_p)$ (i.e. the sign was not inverted), then the removals are undone and the solution returns to the initial state. Figure 2.6 illustrates an example involving the application of the *sign inversion* mechanism.

The intercluster sign invertion, e.g., from S_p to S_q, may be easily derived by considering only the arcs $A[S_p : S_q]$ that determine the vertices to be removed from S_p and by changing the condition (iii) to $|S_p| > 1$. Note that no vertices are removed from S_q but it may receive vertices from S_p.

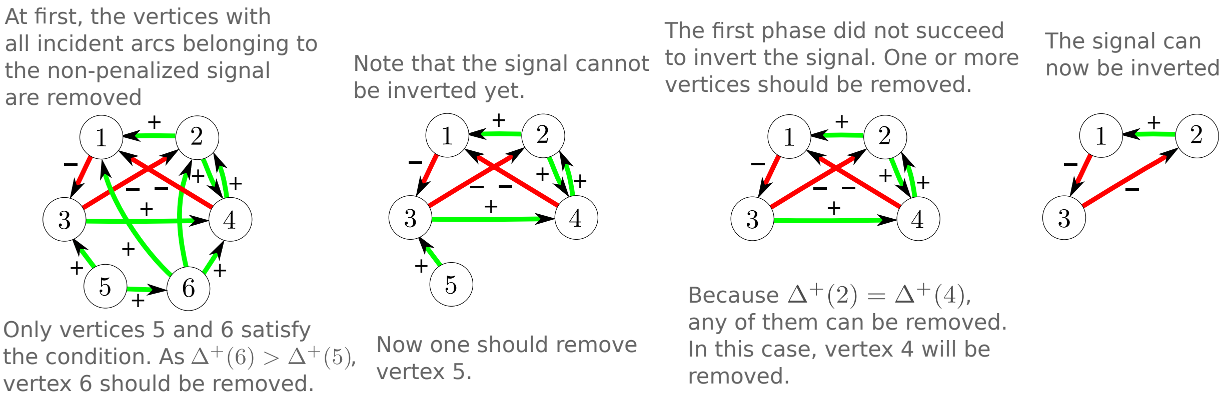


Figure 2.6: An example of sign inversion for an intracluster imbalance. For the sake of simplicity, all arcs have unitary weight.

This perturbation allows for exploring some particular regions of the search space that is difficult to be achieved by only using the other mechanisms, including the randomized construction procedure, mainly when sign distribution on arcs is unbalanced and small changes are not likely to invert the sign.

Each time the function `Perturb` is called, a perturbation mechanism is randomly chosen. The selected perturbation then applies from two up to *maxPert* moves. The number of moves is also chosen at random. If $l = 2$, *Merge* is not an eligible perturbation. Therefore, when *Sign inversion* is chosen and no change could be performed, one of the other two remaining mechanisms (or *Insertion* if $l = 2$) is randomly selected.

2.5.4 Differences between ILS$_{\text{RCC}}$ and ILS [76]

Table 2.4 presents the main differences between ILS$_{\text{RCC}}$ and the ILS by Levorato et al. [76] which was developed for the SRCC.

Table 2.4: Differences between ILS_{RCC} and ILS [76]

	ILS_{RCC}	ILS [76]
Initial solution	Random	Greedy randomized procedure
Local search	*Insertion*, *Swap* and *Split* Best improvement strategy	*Insertion* First improvement strategy
Perturbation	*Insertion*, *Merge* and *Sign Inversion*	*Insertion*

It is worth mentioning that we tried to incorporate the constructive procedure implemented in Levorato et al. [76] into our algorithm, but the experiments reported in Section 2.6.3 indicated that its inclusion did not seem to significantly affect the overall performance of ILS_{RCC} both in terms of solution quality and CPU time.

2.6 Computational results

All algorithms have been implemented in C++ and executed using a single thread on a PC Intel Core i7-2600 with 3.40 GHz and 16 GB of RAM running Ubuntu 16.04 LTS (64 bits). For results based on ILP formulations, CPLEX 12.7 is used as a MIP solver (single thread) with all other parameters set to their default values.

2.6.1 Benchmark instances

Regarding the benchmark instances used in our testing, we first present the small-size instances from the literature. Next, we introduce the newly proposed RCC instances and, finally, we describe the existing SRCC instances.

Small-size instances from the literature

The small-size instances considered here were proposed in different works and together they compose a set of nine signed digraphs described as follows.

- *House instances* — In 1952, Lemann and Solomon [72] carried out a sociometric study with students living in three different dormitories (denoted as House A, House B and House C) and obtained four relationship networks per dormitory considering the following information: date, friend, roommate and weekend. Doreian [38] later summed the arc weights of the signed networks associated with each dormitory so as to generate another three networks: House A Sum, House B Sum and House C Sum. We have considered these last three in our experiments.

- *Monastery instances* — In 1868, Sampson [101] studied, in different periods of time, groups of young or novice postulants of a monastery, cataloging data for four types of relationships: affect, esteem, influence, and sanction. From this data, networks were generated for each period of time and type of relationship. Among them, we considered those associated with the relationship affect for different periods of time, namely MonkT2, MonkT3, and MonkT4. In addition, we considered the network Mont4 Sum, generated in Doreian [38] by summing up the arc weights of the four types of relationships in period T4.

- *McKinney instance* — This signed digraph was built by Brusco et al. [19] from the data collected by McKinney [81] in a study about the relationship between children in a classroom. In such study, children were submitted to a test in which they had to choose between the "willing to serve with other children" (labeled as +1), "not being willing to serve" (labeled as -1) and "indifferent" (labeled as 0), defining the class relationship digraph.

- *NewComb instance* — In 1961, Newcomb [84] conducted a well-known sociometric study with University students. A signed digraph was generated in Doreian and Mrvar [40] by slightly modifying the data from this study.

The main characteristics of the aforementioned instances are described in Table 2.5, where d and d^- indicate the digraph density (given by $d = |A|/(|V|^2 - |V|)$) and the percentage of negative arcs (given by $d^- = |A^-|/|A|$), respectively. For the sake of convenience, we have specified an alias (in parentheses) for each instance.

Table 2.5: Small-size instance attributes

Name	$\|V\|$	d	d^-	Author(s)
House A Sum (HAS)	21	0.50	0.56	Doreian [38] and Lemann and Solomon [72]
House B Sum (HBS)	17	0.59	0.52	Doreian [38] and Lemann and Solomon [72]
House C Sum (HCS)	20	0.52	0.53	Doreian [38] and Lemann and Solomon [72]
MonkT2 (MT2)	18	0.34	0.47	Sampson [101]
MonkT3 (MT3)	18	0.34	0.46	Sampson [101]
MonkT4 (MT4)	18	0.34	0.46	Sampson [101]
MonkT4 Sum (MT4S)	18	0.50	0.49	Doreian [38] and Sampson [101]
McKinney (MK)	18	0.34	0.10	Brusco et al. [19] and McKinney [81]
NewComb (NC)	17	0.44	0.43	Doreian and Mrvar [40] and Newcomb [84]

Random instances

In order to test the ILS implementations on larger instances, we have generated 48 new signed digraphs with different values of $|V|$, d and d^-. Let $\mathcal{V}_{|V|} = \{100, 200, 400, 600\}$, $\mathcal{V}_d = \{0.1, 0.2, 0.5, 0.8\}$ and $\mathcal{V}_{d^-} = \{0.1, 0.3, 0.5\}$ be the set of values associated with $|V|$, d and d^-, respectively. For each setting obtained by the Cartesian product $\mathcal{V}_{|V|} \times \mathcal{V}_d \times \mathcal{V}_{d^-}$ (represented by a 3-tuple), we have randomly built a signed digraph. Note that larger values of d^- are not used because they are equivalent with respect to the desired sign distribution (e.g., if $d^- = 0.7$, then the percentage of positive arcs will be 0.3). A RCC instance consists of a digraph and a value for the parameter k (maximum number of clusters). For each generated digraph, we consider one instance for each value of k in $\{3, 5, 7, 9\}$. Therefore, this benchmark is composed of 192 instances.

The newly generated digraphs are available at `http://www.ic.uff.br/~yuri/files/rcc_random.zip`.

Symmetric RCC instances

We also considered three sets of symmetric RCC benchmark instances, namely:

- *UNGA instances* — Generated by Levorato et al. [75] and composed of 63 undirected graphs that were built from the voting data of the United Nations General Assembly (UNGA) annual meetings between 1946 and 2008. These networks are weighted versions of UNGA signed digraphs created by Figueiredo and Frota [46].

- *Slashdot instances* — Created by Levorato [73] from subgraphs of the social network Slashdot Zoo containing 200 to 10000 vertices. Such subgraphs were transformed into undirected graphs. Levorato [73] performed experiments with a parallel heuristic for the SRCC. Since we are specifically interested in comparing the performance of sequential implementations, it was thought advisable to consider the instances with up to 2000 vertices.

- *BR Congress instances* — Set of undirected graphs generated by Levorato and Frota [74] from voting sessions of the lower house of Brazilian National Congress. They created two graphs per year between 2011 and 2016, resulting in a total of 12 instances.

The reader is referred to Figueiredo and Frota [46], Levorato [73], Levorato and Frota [74], and Levorato et al. [75] for a more detailed description.

2.6.2 Results for the ILP formulations

Tables 2.6, 2.7 and 2.8 present the results obtained by the formulation proposed in Figueiredo and Moura [48], as well as those determined by `F1` and `F2`. Column ***z*** represents the relaxed imbalance, given by $RI(P^*)$, where P^* is the solution (optimal or not) found by the corresponding formulation, **gap** informs percentage gaps calculated between best integer solutions found and final lower bounds (LB) as described in Equation (2.21), ***t*** indicates the CPU time in seconds ("-" means the instance was not solved in the time limit set), and **nodes** is the number of nodes that were solved during the search. Regarding the ILP formulation proposed by Figueiredo and Moura [48], we report the original results which were found using XPRESS 21.01.00 and also those determined by CPLEX 12.7 in order to perform a fair comparison. Since Figueiredo and Moura [48] carried out their experiments using a different machine (Intel Core 2 Duo 2.10 GHz), we scaled the XPRESS CPU times by a factor of 0.48 according to the single-thread ratings reported in `https://www.cpubenchmark.net/compare/Intel-Core2-Duo-T6500-vs-Intel-i7-2600/995vs1`. A time limit of 3600 seconds was imposed for each run.

$$gap = 100 \times (BestInteger - LB)/BestInteger \tag{2.21}$$

We followed the same procedure adopted in Figueiredo and Moura [48] in our testing. For each digraph, we start the experiments with $k = 2$. If the instance is solved to optimality, we then increase the value of k by one unit and attempt to solve the problem again (*forward phase*). If an optimal solution with relaxed imbalance 0 is found, we then interrupt the experiments for that particular digraph since this solution is also optimal for instances with larger values of k. When an instance is not solved to optimality, a similar procedure is carried out to solve instances from $k = n - 1$, where the value of k is decreased by one unit after each successful optimization (*backward phase*). In the backward phase, we use the value of the optimal solution found in the previous run (i.e., for $k + 1$) as a lower bound for the instance with k. The backward phase is finished when an instance is not solved to optimality or when the current instance was solved during the forward phase.

Table 2.6: Results obtained for instances House A Sum, House B Sum and House C Sum.

Instance	k	Literature ILP formulation								F1				F2			
		XPRESS				CPLEX											
		z	gap	t	nodes	z	gap	t	nodes	z	gap	t	nodes	z	gap	t	nodes
HAS	2	96	0	28	1579	96	57.3	–	14874	96	0	1	688	96	0	22	1362
	3	57	78.9	–	31737					**50**	0	19	10921	**50**	0	2911	103801
	4									**31**	0	92	31468	**31**	0	3115	118591
	5									**27**	0	824	116453	27	78.2	–	94578
	6									**21**	0	1931	185902				
	7									18	29.7	–	270282				
	10									6	33.3	–	99450	6	33.3	–	50536
	11									**4**	0	1415	34689	**4**	0	501	8484
	12									**1**	0	190	5535	**1**	0	73	1544
	13	12	83.3	–	20945					**0**	0	109	2396	**0**	0	70	1240
	14	2^a	0	746	16703					0	0	61	1380	0	0	37	730
	15	0	0	1721	30208					0	0	60	831	0	0	7	100
	16	0	0	558	7358					0	0	18	320	0	0	9	210
	17	0	0	288	2319	6	100	–	2499	0	0	91	1300	0	0	19	592
	18	0	0	288	2634	0	0	921	1455	0	0	19	234	0	0	10	248
	19	0	0	11	1	0	0	618	1180	0	0	1	1	0	0	10	260
	20	0	0	< 1	1	0	0	408	1139	0	0	82	671	0	0	10	260
HBS	2	84	0	11	1115	84	69.6	–	28015	84	0	< 1	747	84	0	10	1381
	3	75	47.5	–	9064					**69**	0	56	61150	**69**	0	867	99990
	4									**56**	0	461	203765	56	39.3	–	286043
	5									**43**	0	2161	514692				
	6									**33**	0	2595	383348				
	7									29	44.1	–	409051				
	9									15	26.7	–	138796	15	26.7	–	102548
	10									**11**	0	2536	97956	**11**	0	2186	55691
	11									**8**	0	592	17562	**8**	0	642	20574
	12	5	60	–	80375					**5**	0	297	8618	**5**	0	185	6802
	13	2	0	343	13538					2	0	67	2056	2	0	11	883
	14	1	0	134	3761					1	0	14	631	1	0	9	690
	15	0	0	41	584					0	0	28	1026	0	0	2	1
	16	0	0	< 1	1					0	0	18	942	0	0	2	1

Table 2.6 – *Continued from previous page*

Instance	k	Literature ILP formulation								F1				F2			
		XPRESS				CPLEX											
		z	gap	t	nodes	z	gap	t	nodes	z	gap	t	nodes	z	gap	t	nodes
	2	64	0	12	615	64	0	1348	7266	64	0	1	363	64	0	8	555
	3	60	83	–	42981	96	100	–	1323	**53**	0	27	19422	**53**	0	417	25068
	4									**43**	0	341	137992	44	49.1	–	112782
	5									**35**	0	2348	396810				
	6									31	44.2	–	344290				
	12									8	37.5	–	73495	8	37.5	–	81225
HCS	13									**5**	0	2340	37753	**5**	0	1961	40391
	14									**3**	0	1301	18368	**3**	0	476	9800
	15	3	66.7	–	35179					**2**	0	369	4734	**2**	0	84	1900
	16	1	0	1007	12517	5	100	–	2556	1	0	103	2270	1	0	16	669
	17	0	0	57	383	0	0	304	500	0	0	34	557	0	0	2	1
	18	0	0	46	153	0	0	255	500	0	0	23	433	0	0	2	1
	19	0	0	< 1	1	0	0	261	500	0	0	59	1024	0	0	2	1

[a] Possible typo

Table 2.7: Results obtained for instances MonkT2, MonkT3, MonkT4, and MonkT4 Sum.

Instance	k	Literature ILP formulation								F1				F2			
		XPRESS				CPLEX											
		z	gap	t	nodes	z	gap	t	nodes	z	gap	t	nodes	z	gap	t	nodes
	2	43	0	6	733	43	0	37	1343	43	0	< 1	818	43	0	2	534
	3	25	0	1074	70771	27	92.6	–	11388	25	0	5	7049	25	0	42	6271
	4	20	85	–	121561					**13**	0	9	6998	**13**	0	212	26725
	5									**8**	0	15	7077	**8**	0	116	11601
	6									**4**	0	11	3765	**4**	0	76	8493
	7					14	92.9	–	12097	**2**	0	4	1177	**2**	0	6	718
MT2	8					**1**	0	509	2497	**1**	0	2	236	**1**	0	4	488
	9					**0**	0	666	3835	**0**	0	1	252	**0**	0	4	720
	10					0	0	128	1484								
	11	2	100	–	176063	0	0	169	1781								
	12	0	0	1137	102937	0	0	80	1578								
	13	0	0	107	8881	0	0	19	278								
	14	0	0	23	593	0	0	54	1438								
	15	0	0	7	69	0	0	62	1543								
	16	0	0	3	1	0	0	61	1597								
	17	0	0	< 1	1	0	0	40	1082								

Table 2.7 – *Continued from previous page*

Instance	k	Literature ILP formulation								F1				F2			
		XPRESS				CPLEX											
		z	gap	t	nodes	z	gap	t	nodes	z	gap	t	nodes	z	gap	t	nodes
MT3	2	32	0	3	243	32	0	30	1404	32	0	<1	126	32	0	2	109
	3	21	0	93	4765	28	92.9	–	8924	21	0	1	876	21	0	7	1964
	4	13	0	1089	54227					13	0	2	1835	13	0	13	2690
	5	8	0	1596	85056					8	0	7	3460	8	0	10	2013
	6	7	71.4	–	100613					**6**	0	4	1840	**6**	0	34	3979
	7	5	60	–	140597	4	50	–	21250	**4**	0	4	1551	**4**	0	9	1749
	8	2	0	1362	6725	2	0	675	6642	2	0	2	517	2	0	5	739
	9	1	0	210	11577	1	0	56	1610	1	0	1	171	1	0	3	290
	10	0	0	567	44984	0	0	50	1090	0	0	<1	1	0	0	2	402
	11	0	0	120	8439	0	0	32	989								
	12	0	0	249	21543	0	0	7	150								
	13	0	0	49	5335	0	0	7	143								
	14	0	0	24	1120	0	0	8	174								
	15	0	0	14	455	0	0	8	197								
	16	0	0	9	119	0	0	8	197								
	17	0	0	<1	1	0	0	8	197								
MT4	2	25	0	2	149	25	0	11	206	25	0	<1	194	25	0	1	99
	3	21	0	57	3381	21	75.2	–	12524	21	0	1	923	21	0	6	1459
	4	10	0	270	13945					10	0	1	706	10	0	9	1864
	5	6	0	702	42782					6	0	3	1283	6	0	64	6038
	6	4	0	1148	68659	5	80	–	7519	4	0	2	719	4	0	47	4389
	7	1	0	328	19452	1	0	141	1370	1	0	1	308	1	0	3	476
	8	0	0	199	12469	0	0	31	691	0	0	1	61	0	0	2	331
	9					0	0	66	537								
	10					0	0	35	756								
	11					0	0	4	1								
	12					0	0	11	1								
	13					0	0	11	1								
	14					0	0	9	1								
	15					0	0	8	1								
	16					0	0	9	1								
	17					0	0	8	1								

Table 2.7 – *Continued from previous page*

Instance	k	Literature ILP formulation								F1				F2			
		XPRESS				CPLEX											
		z	gap	t	nodes	z	gap	t	nodes	z	gap	t	nodes	z	gap	t	nodes
MT4S	2	86	0	7	347	86	0	66	518	86	0	< 1	217	86	0	2	229
	3	54	0	739	25379	85	98.8	–	2941	54	0	1	988	54	0	80	5595
	4	43	72.2	–	7483					**36**	0	8	3340	**36**	0	298	21034
	5									**25**	0	23	6128	**25**	0	300	17832
	6									**16**	0	42	7039	**16**	0	225	11659
	7									**12**	0	41	5561	**12**	0	131	5068
	8					9	66.7	–	9104	**8**	0	18	2281	**8**	0	58	2232
	9	6	100	–	61861	**3**	0	3407	5188	**3**	0	7	590	**3**	0	8	525
	10	2	0	1239	51491	2	0	3498	5701	2	0	3	172	2	0	6	372
	11	0	0	835	25094	0	0	79	470	0	0	1	1	0	0	8	602
	12	0	0	444	11838	0	0	126	935								
	13	0	0	28	240	0	0	108	935								
	14	0	0	144	3389	0	0	18	1								
	15	0	0	3	1	0	0	19	1								
	16	0	0	< 1	1	0	0	19	1								
	17	0	0	< 1	1	0	0	18	1								

Table 2.8: Results obtained for instances McKinney and NewComb.

Instance	k	Literature ILP formulation								F1				F2			
		XPRESS				CPLEX											
		z	gap	t	nodes	z	gap	t	nodes	z	gap	t	nodes	z	gap	t	nodes
MK	2	8	0	57	6531	8	100	–	7559	8	0	< 1	28	8	0	112	2802
	3	6	100	–	43762					**2**	0	< 1	197	**2**	0	227	7766
	4									**0**	0	< 1	1	**0**	0	50	1567
	13					2	100	–	1432								
	14					0	0	57	1								
	15					0	0	59	1								
	16	2	100	–	33562	0	0	53	1								
	17	0	0	39	169	0	0	53	1								
	18	0	0	1	1	0	0	52	1								
	19	0	0	9	1	0	0	53	1								
	20	0	0	< 1	1	0	0	55	1								
	21	0	0	8	1	0	0	58	1								
	22	0	0	1	1	0	0	55	1								
	23	0	0	2	1	0	0	60	1								
	24	0	0	3	1	0	0	58	1								
	25	0	0	< 1	1	0	0	53	1								
	26	0	0	46	49	0	0	57	1								
	27	0	0	1	1	0	0	59	1								
	28	0	0	< 1	1	0	0	54	1								
NC	2	10	0	2	167	10	0	34	468	10	0	< 1	66	10	0	1	101
	3	7	0	228	9869	8	100	–	4988	7	0	1	1193	7	0	43	6188
	4	5	34.6	–	90604					**5**	0	7	5570	**5**	0	83	10538
	5					6	83.3	–	9349	**3**	0	11	4690	**3**	0	80	8054
	6					**1**	0	1299	6295	**1**	0	1	331	**1**	0	35	3561
	7					**0**	0	2646	7542	**0**	0	3	1093	**0**	0	12	1177
	8	1	100	–	146619	0	0	846	2861								
	9	0	0	83	9807	0	0	594	2624								
	10	0	0	59	5969	0	0	500	2276								
	11	0	0	13	405	0	0	46	1005								
	12	0	0	18	162	0	0	142	1815								
	13	0	0	4	1	0	0	294	2075								
	14	0	0	< 1	1	0	0	116	1172								
	15	0	0	< 1	1	0	0	85	1505								
	16	0	0	< 1	1	0	0	64	1505								

The results obtained show that `F1` outperforms the other formulations with respect to the number of optimal solutions achieved. When a formulation obtained an optimal solution with relaxed imbalance 0 for a given value of k, then we assume that all optima were found for executions with larger values of k (e.g., `F1` and `F2` solved instances from MT2 to optimality). While this formulation found 38, 64, 27, 15 optimal solutions for each group (House, Monastery, McKinney, NewComb), respectively, `F2` found 29, 64, 27, 15, respectively, and the formulation by Figueiredo and Moura [48] obtained 18, 48, 13, 10 using XPRESS, and 7, 44, 15, 12 using CPLEX, respectively. Overall, a total of 40 new optimal solutions (counting only once the optimal solutions with relaxed imbalance 0 for each digraph) were found and all instances of 6 of the 9 groups were solved to optimality.

In addition, it can be observed that `F1` is generally faster, but it is outrun by `F2` on instances HAS, HBS and HCS for larger values of k. Note that for the latter two, `F2` solves some instances at the root node. The formulation by Figueiredo and Moura [48] is clearly slower than the proposed ones. In general, more nodes are solved when running such formulation, but in some cases, `F2` is the one to solve more nodes.

The results also demonstrate that RCC has the expected behavior for k-partition problems, where problems with k close to 2 and $n-1$ are easier than those problems with k close to $n/2$. This can be explained by the number of possible partitions, which is given by the Stirling number of the second type [18] as described in Equation (2.22).

$$S(n,k) = \frac{1}{k!}\sum_{j=0}^{k}(-1)^{k-j}\binom{k}{j}j^n \tag{2.22}$$

Finally, we report in Table 2.9 a comparison between the optimal solutions for RCC and CC. In addition to comparing $I(P)$ with $RI(P)$ and the correction $\delta = I(P) - RI(P)$ obtained with RCC (recall that RCC was proposed in order to correct *wrong penalties* in CC), we also decompose the total imbalance into minimum, average, and maximum penalties for intracluster and intercluster cases. With the exception of one instance (MT4 with $k=3$), a positive correction is obtained by RCC. It is worth mentioning that most of the imbalance (and hence the correction) occurs in intercluster cases.

2.6.3 ILS implementations

We used the same values adopted in Subramanian and Farias [104] for the main parameters of ILS$_{\text{RCC}}$, that is, $I_R = 20$ and $I_{ILS} = \min\{100, 4 \times n\}$. The only difference is that we imposed a minimum value of 100 for the latter as in Silva et al. [103]. Moreover, we set $maxPert = 6$ after conducting some experiments (see Section 2.6.3). The algorithms were executed 10 times on each instance in all experiments. Hereafter, the percentage gap of a solution P' is computed as $gap = 100 \times (f(P') - f(P_{best}))/f(P')$, such that f is

Table 2.9: Comparison of optimal solutions for RCC and CC in small instances.

Name	k	$I(P)$							$RI(P)$							δ
		total	intra			inter			total	intra			inter			
			min	avg	max	min	avg	max		min	avg	max	min	avg	max	
HAS	4	64	0	4.8	16	0	7.5	19	31	0	2.3	5	0	1.8	5	33
HBS	4	81	0	3.5	14	1	11.2	23	56	0	5.8	11	0	2.8	10	25
HCS	3	59	0	5.3	10	4	14.3	27	53	3	5.7	8	3	6.0	10	6
MT2	3	35	0	0.7	1	3	11.0	16	25	0	0.3	1	1	4.0	7	10
MT3	3	22	0	0.3	1	4	7.0	11	21	0	1.0	3	0	3.0	7	1
MT4	3	21	0	1.7	5	3	5.3	9	21	0	1.7	5	0	2.7	6	0
MT4S	3	62	1	3.0	7	12	17.7	25	54	1	3.0	7	2	7.5	15	8
MK	2	12	0	3.0	6	6	6.0	6	8	0	2.0	4	2	2.0	2	4
NC	4	20	0	1.3	5	0	2.5	7	5	0	0.5	1	0	0.3	1	15

the objective function (i.e., $f(P) = RI(P)$ for RCC and $f(P) = SRI(P)$ for SRCC) and P_{best} is the best solution among all solutions found by the ILS$_{\text{RCC}}$ and the ILS of Levorato et al. [76].

Impact of the different components of the algorithm

We evaluate the ILS$_{\text{RCC}}$ concerning the impact of: (i) using the greedy constructive algorithm by Levorato et al. [76]; (ii) the parameter $maxPert$; (iii) the neighborhood structures; (iv) perturbation mechanisms. To this end, we conducted experiments involving all 100-vertex random instances.

At first, we assess the impact of replacing the completely random construction (line 5 in Algorithm 1) with the greedy construction of Levorato et al. [76]. Table 2.10 shows the average gaps and CPU times obtained by the two versions of ILS$_{\text{RCC}}$ (using two different constructive procedures) for different values of k. It can be seen that none of the two versions are significantly superior than the other w.r.t. both criteria. The only exception occurs when $k = 9$, where using the completely random construction produces a superior average gap and CPU time. In general, using the greedy construction in ILS$_{\text{RCC}}$ leads to an improvement of only 0.01% in terms of average gap, and an increase of around 3% on the average CPU time. Therefore, it is reasonable to conclude that the effort to implement a more sophisticated constructive procedure may not be worthwhile for the RCC when the algorithm contains an effective local search.

The impact of varying the parameter $maxPert$ is illustrated in Figure 2.7, where values between 4 and 8 are considered to compare different versions of ILS$_{\text{RCC}}$. The results show that the values 5 and 7 are dominated by the remaining ones; the value 4 produces

Table 2.10: Comparison of two constructive heuristics in ILS_{RCC} for different values of k.

Construction	k	Avg. gap (%)	Avg. time (s)
Greedy [76]	3	**0.14**	7.09
	5	**0.92**	13.76
	7	**1.77**	19.78
	9	2.84	24.20
Mean		**1.42**	16.21
Random	3	0.17	**6.83**
	5	1.03	**13.36**
	7	1.96	**19.20**
	9	**2.56**	**23.31**
Mean		1.43	**15.68**

the faster execution but with a poor average gap, whereas the value 8 achieves the best average gap but the worst CPU time. Therefore, we decided to use $maxPert = 6$ because it yields a good balance between solution quality and CPU time.

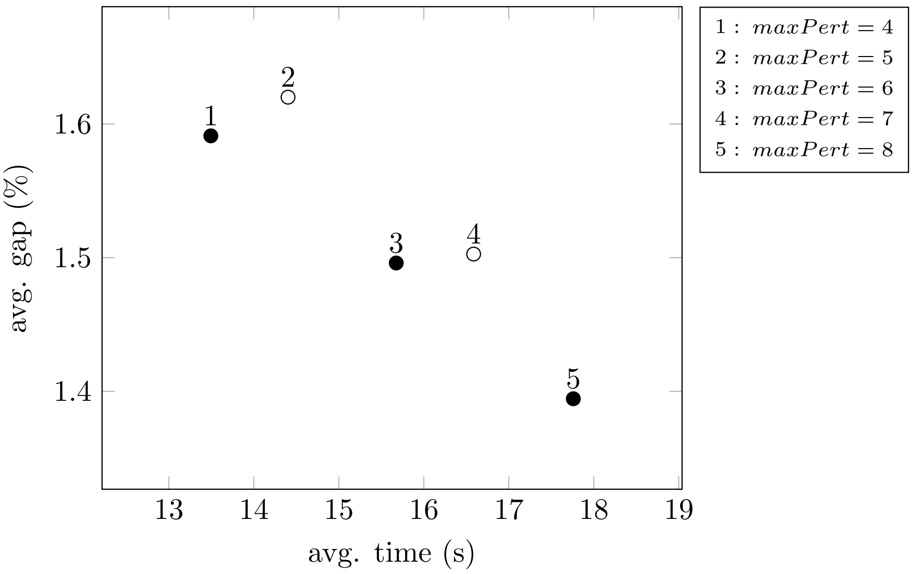


Figure 2.7: Impact of the parameter $maxPert$. Each point represents a configuration and points with no fill represent those dominated by one or more settings.

We assess the impact of the neighborhood operations by considering 7 different configurations. In this case, we consider $I_R = 20$ and $I_{ILS} = 1$ (i.e. only one iteration of the RVND procedure is executed) and we measure the percentage improvement over the initial solution. The average results are shown in Figure 2.8. We can observe that the configurations that yields the most promising results are those 5 and 7. The difference between both settings is that the latter includes the neighborhood *Swap*, which led to a slight improvement despite the additional CPU time.

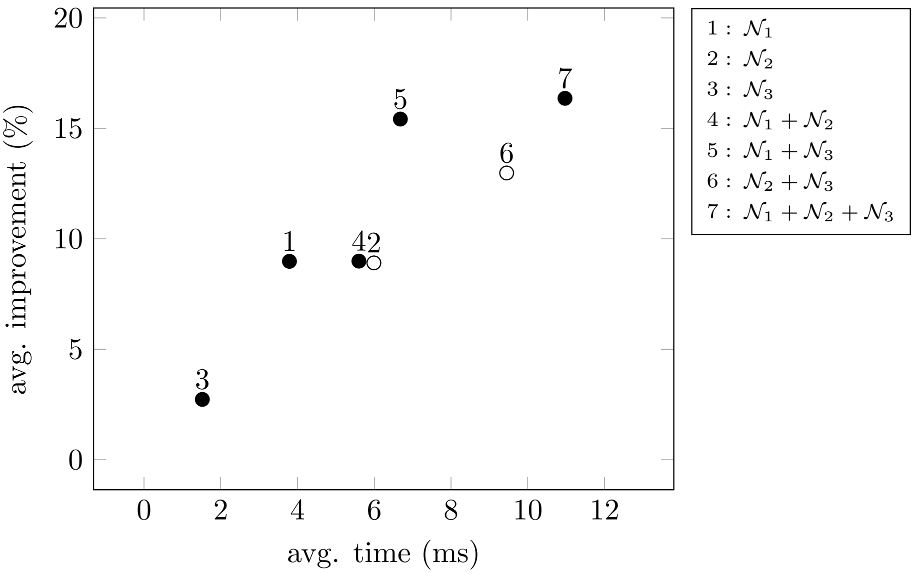


Figure 2.8: Impact of the neighborhood operations for all 100-vertex random instances. Each point represents a configuration and points with no fill represent those dominated by one or more settings. $\mathcal{N}_1$, $\mathcal{N}_2$ and $\mathcal{N}_3$ denote neighborhoods *Insert*, *Swap* and *Split*, respectively.

In order to measure the impact of the perturbation mechanisms, we run ILS_{RCC} using the default values of the parameters and store the percentage improvement over the best solution found in the previous testing for each instance. To choose an interesting configuration, we perform experiments considering two scenarios: with and without the neighborhood *swap*. The average results obtained are depicted in Figure 2.9. The best results are obtained by settings 6 and 7 for both scenarios. The scenario that considered *Swap* achieves better improvements at the expense of CPU time. Although not reported in Figure 2.9, the settings tested in the second scenario (i.e., the one including *Swap*) systematically found more best solutions than their corresponding counterpart in the first scenario. We, therefore, decided to select configuration 7 of the second scenario because it appears to offer an interesting compromise between solution quality and CPU time.

Comparison with the literature

The implementation by Levorato et al. [76] was originally devised for the symmetric version of the problem. Therefore, we had to slightly modify the source code, which was provided by the authors, to cope with the asymmetric case. We will refer to this method as $\text{ILS}_{\text{adapt}}$.

Concerning the small-size instances considered in Section 2.6.1, it was observed that ILS_{RCC} and $\text{ILS}_{\text{adapt}}$ are capable of consistently finding the optimal solutions in a

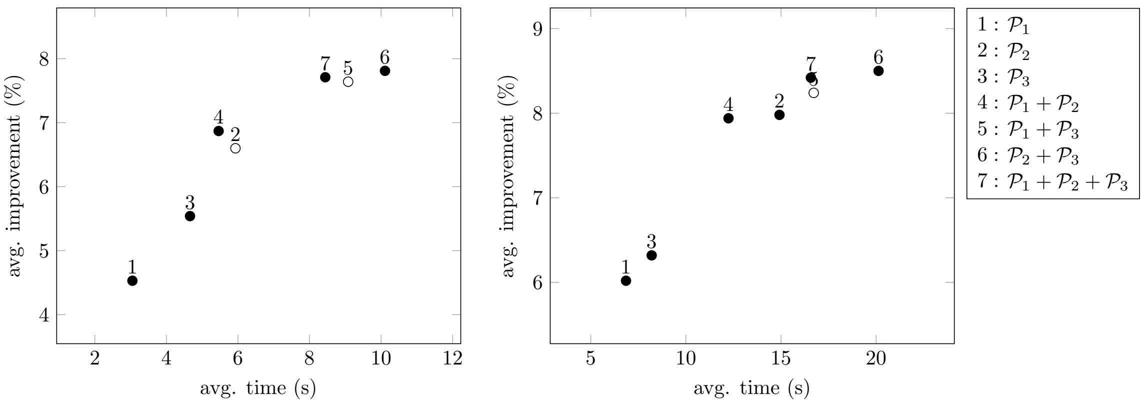


Figure 2.9: Impact of the perturbation operations. Each point represents a configuration and points with no fill represent those dominated by one or more settings. $\mathcal{P}_1$, $\mathcal{P}_2$ and $\mathcal{P}_3$ denote perturbations *Insert*, *Split* and *Sign Inversion*, respectively. Part (a) does not include the neighborhood *Swap* whereas part (b) does.

fraction of a second.

Table 2.11 shows the aggregate results obtained by the ILS implementations on the set of random instances. For each digraph, we report the minimum, average and maximum values of the average percentage gaps (there is an average gap for each value of $k \in \{3, 5, 7, 9\}$), as well as the average CPU time in seconds. Detailed results are reported in Appendix D. Moreover, for an appropriate comparison, we have imposed the average CPU time obtained by ILS_{RCC}, for each instance, as a stopping criterion for $\text{ILS}_{\text{adapt}}$.

Table 2.11: Aggregate results for each digraph. Each row reports statistics on the average gaps obtained for the group of four instances (one for each value of $k \in \{3, 5, 7, 9\}$).

$\lvert V\rvert$	d	d^-	ILS_{RCC}			$\text{ILS}_{\text{adapt}}$			t_{avg}
			min	avg	max	min	avg	max	
100	0.1	0.1	**0.82**	5.02	11.00	3.23	17.65	34.73	8.99
100	0.1	0.3	**0.02**	1.57	2.49	1.18	8.17	16.24	12.60
100	0.1	0.5	**0.02**	2.27	4.59	0.76	6.29	10.99	13.25
100	0.2	0.1	**0.69**	1.35	1.66	1.62	4.92	7.46	10.57
100	0.2	0.3	**0.00**	0.71	1.36	0.10	2.78	4.56	15.64
100	0.2	0.5	**0.11**	1.14	2.67	0.62	3.08	5.73	15.19
100	0.5	0.1	**0.12**	0.35	0.69	0.35	1.33	2.53	12.23
100	0.5	0.3	0.00	0.21	0.53	0.00	0.93	1.98	21.70
100	0.5	0.5	**0.15**	0.43	0.74	0.27	1.27	2.17	20.98
100	0.8	0.1	**0.00**	0.11	0.28	0.22	0.95	1.88	8.91
100	0.8	0.3	0.05	0.22	0.37	**0.03**	0.53	1.08	23.18

Table 2.11 – *Continued from previous page*

$\lvert V\rvert$	d	d^-	$\mathrm{ILS_{RCC}}$ min	avg	max	$\mathrm{ILS_{adapt}}$ min	avg	max	$\mathrm{t_{avg}}$
100	0.8	0.5	**0.00**	0.21	0.34	0.29	1.10	1.81	24.85
200	0.1	0.1	**0.52**	0.87	1.13	1.23	6.24	13.75	66.34
200	0.1	0.3	**0.03**	0.81	1.29	0.94	2.75	3.65	97.29
200	0.1	0.5	**0.39**	0.60	0.88	1.03	3.13	5.10	91.26
200	0.2	0.1	**0.15**	0.73	1.40	0.37	2.34	4.23	73.44
200	0.2	0.3	**0.16**	0.28	0.41	0.18	1.35	2.12	116.49
200	0.2	0.5	**0.19**	0.44	0.67	0.73	1.47	2.01	102.96
200	0.5	0.1	**0.04**	0.20	0.32	0.22	0.78	1.11	66.67
200	0.5	0.3	**0.05**	0.13	0.21	0.07	0.52	0.81	155.52
200	0.5	0.5	**0.05**	0.21	0.33	0.34	0.75	1.05	142.09
200	0.8	0.1	**0.01**	0.11	0.19	0.12	0.52	0.84	44.26
200	0.8	0.3	**0.01**	0.15	0.35	0.14	0.50	0.83	161.70
200	0.8	0.5	**0.04**	0.19	0.28	0.52	0.85	1.47	165.27
400	0.1	0.1	0.34	0.56	0.73	0.34	1.01	1.49	508.86
400	0.1	0.3	0.16	0.31	0.57	**0.09**	0.84	1.45	870.00
400	0.1	0.5	**0.09**	0.42	0.71	0.32	1.26	2.56	699.45
400	0.2	0.1	**0.07**	0.17	0.20	0.12	0.38	0.65	394.25
400	0.2	0.3	**0.04**	0.17	0.26	0.08	0.81	1.83	988.04
400	0.2	0.5	**0.11**	0.23	0.32	0.26	0.59	0.76	778.40
400	0.5	0.1	**0.05**	0.10	0.17	0.14	0.26	0.41	312.75
400	0.5	0.3	**0.03**	0.06	0.07	0.06	0.14	0.24	978.23
400	0.5	0.5	**0.08**	0.12	0.18	0.21	0.47	0.92	1093.10
400	0.8	0.1	**0.02**	0.05	0.08	0.07	0.13	0.20	183.53
400	0.8	0.3	**0.02**	0.05	0.07	0.12	0.15	0.19	881.24
400	0.8	0.5	**0.06**	0.11	0.16	0.13	0.30	0.40	1304.41
600	0.1	0.1	0.27	0.40	0.52	**0.20**	0.65	1.03	1338.98
600	0.1	0.3	**0.14**	0.21	0.35	0.18	0.69	1.27	3173.07
600	0.1	0.5	**0.15**	0.29	0.50	0.94	1.43	1.94	2470.35
600	0.2	0.1	**0.12**	0.15	0.20	0.26	0.32	0.39	1034.79
600	0.2	0.3	**0.06**	0.11	0.14	0.10	0.42	0.76	3313.68
600	0.2	0.5	**0.14**	0.22	0.27	0.38	0.56	0.73	2824.27
600	0.5	0.1	**0.03**	0.07	0.10	0.06	0.16	0.25	782.57
600	0.5	0.3	**0.02**	0.05	0.08	0.05	0.12	0.17	2822.82
600	0.5	0.5	**0.04**	0.06	0.11	0.21	0.30	0.36	3866.90
600	0.8	0.1	**0.02**	0.04	0.06	0.03	0.06	0.10	474.08

Table 2.11 – *Continued from previous page*

$\lvert V\rvert$	d	d^-	ILS$_{\text{RCC}}$ min	avg	max	ILS$_{\text{adapt}}$ min	avg	max	t$_{\text{avg}}$
600	0.8	0.3	**0.02**	0.05	0.06	0.04	0.08	0.11	2170.92
600	0.8	0.5	**0.05**	0.09	0.13	0.18	0.30	0.40	4681.35

The results obtained show that, on average, ILS$_{\text{RCC}}$ clearly outperforms ILS$_{\text{adapt}}$. When evaluating the performance of each individual instance, the ILS$_{\text{RCC}}$ found the best solution (one with a gap of 0%) for 179 instances (93.2% of the cases), where among them 166 (86.5% of the cases) are strictly better than the best ones achieved by ILS$_{\text{adapt}}$. Furthermore, we can also see that the average runtime increases with the value of d^-.

Figure 2.10 illustrates how the average gap between the average solution and the best known solution varies according to different values of d, d^- and k. We can observe that the instances appear to become easier when the value of d increases, as depicted in Figure 2.10a. Furthermore, from Figure 2.10b, it is visible that the instances with a smaller value of d^- appear to be harder. In fact, a huge difference in the number of arcs for each sign and the low number of total arcs seems to make the RCC decisions more tricky, perhaps because there are fewer tie situations (e.g. when both signals produce the same penalty in a cluster). Finally, larger values of k seem to increase the difficulty of the instances, as clearly shown in Figure 2.10c.

In Appendix D, we also report many improved upper bounds w.r.t those obtained in the experiment reported in Table 2.11. These improved solutions were found while experimenting with different settings of the algorithm, and also during the preliminary experiments described in Section 2.6.3.

Impact of the ADSs on the runtime performance

This section examines the average runtime performance of ILS$_{\text{RCC}}$ when incorporating the ADSs for efficiently computing the relaxed imbalance value of a neighbor solution during the local search.

Figure 2.11 depicts the CPU time of the versions of the algorithm using EBI and NBI, respectively, in the log scale. In Figure 2.11a, we illustrate the comparison for the small-size Monastery instances. Despite the considerable runtime difference, it can be seen that using NBI, i.e., the one that does not make use of ADSs to perform move

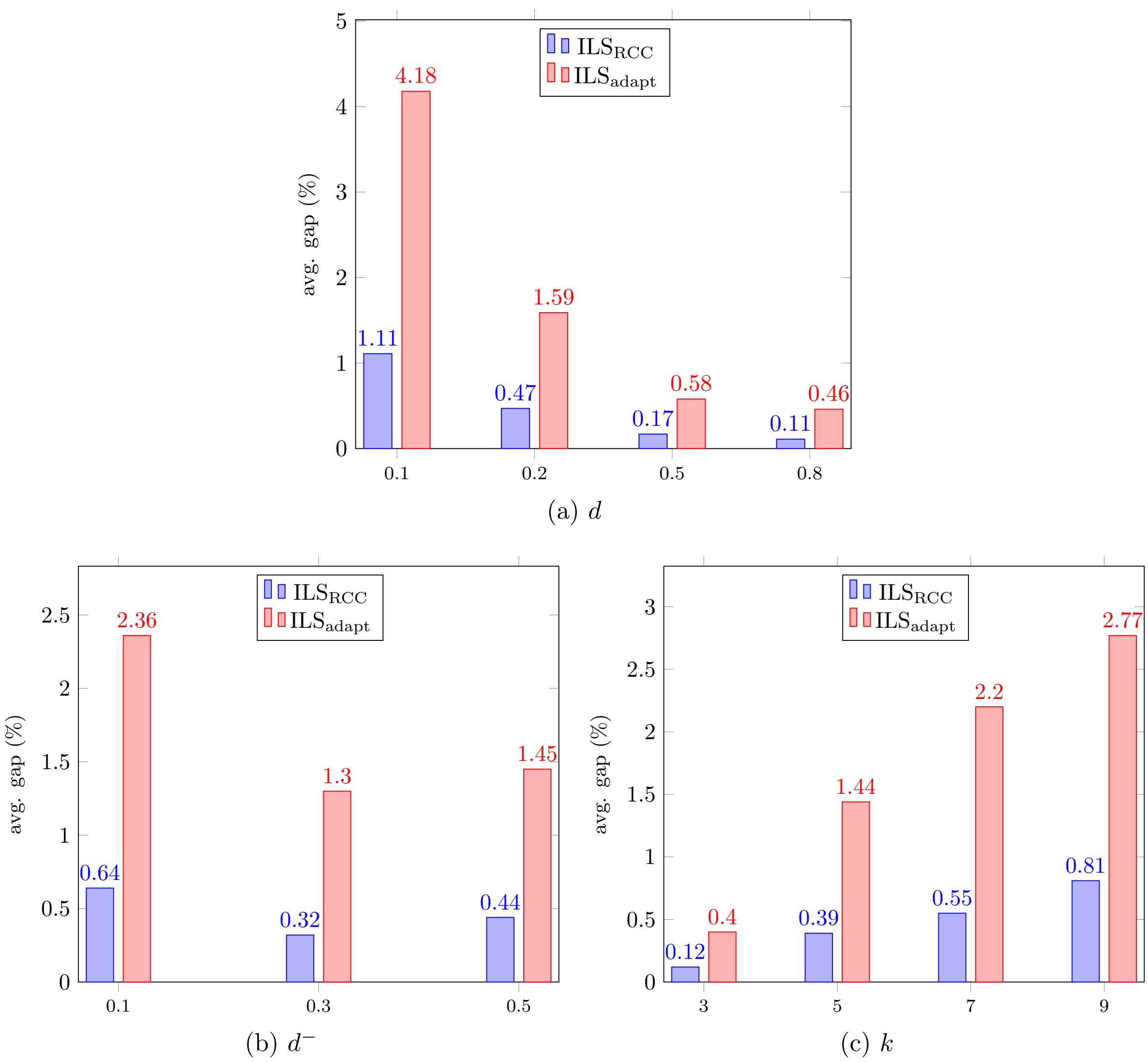
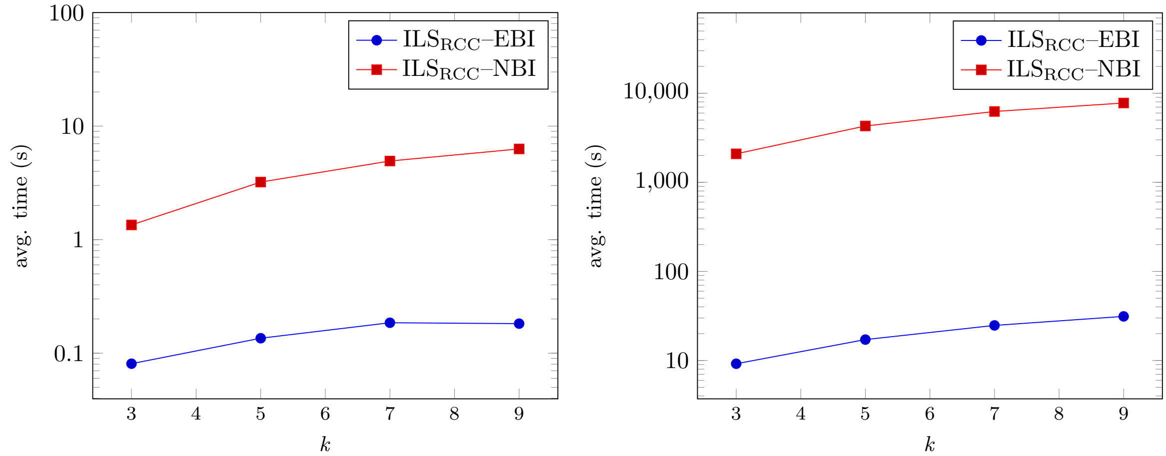


Figure 2.10: Average gap performance according to characteristics of the instance.

evaluation, is still doable in practice, as the average CPU time spent by the method is fairly acceptable. However, for the 100-vertex instances, the difference is astonishing, and visibly illustrates the benefits of incorporating the ADSs proposed in this work. Note that the disparity is likely to become even more prominent for larger instances.



(a) Monastery instances (18 vertices) (b) Random instances with 100 vertices

Figure 2.11: Impact of the ADSs on the average CPU time (semi-log plot).

Results for the symmetric RCC instances

Table 2.12 shows the summary of the results obtained for each set of benchmark instances. In this case, because the original algorithm from the literature is used, we refer to it as "ILS Levorato et al. [76]". Detailed results are provided in Appendix E. We report the number of strictly best solutions found by each version (**#best**), the number of cases in which the best solution found by each algorithm were equal (**ties**), the average percentage gap (**gap$_{\mathbf{avg}}$**) and the minimum, average and maximum CPU time, considering the average values of 10 runs for each instance and a time limit of 7200 seconds. The results illustrate that ILS$_{\text{RCC}}$ dominates ILS Levorato et al. [76] in terms of strictly best known solutions found, especially in Slashdot and Brazilian Congress benchmarks. To our knowledge, all best solutions found in this experiment are the best known.

Table 2.12: Summary of results for SRCC benchmarks

Benchmark	total	ILS$_{\text{RCC}}$		ILS Levorato et al. [76]		ties	time		
		#best	gap$_{\text{avg}}$	#best	gap$_{\text{avg}}$		min	avg	max
UNGA	63	**1**	0.02	0	1.72	62	0.5	4.6	13.0
Slashdot	7	**7**	9.72	0	40.08	0	16.9	2293.3	7200.0
BR Congress	14	**8**	0.59	0	1.16	6	61.3	232.7	534.1

2.7 Concluding remarks

This chapter presented exact and heuristic approaches for the relaxed correlation clustering (RCC) problem. In particular, we developed two integer linear programming formulations that obtained a superior performance when compared to the existing one, as well as an enhanced iterated local search (ILS) algorithm that substantially outperformed the previous ILS implementation from the literature. One key factor of our ILS is the efficient move evaluation scheme, which was crucial for improving the scalability of the method. Moreover, we also put forward a novel perturbation mechanism for the problem that helped the algorithm to find high quality solutions. The performance of ILS was also assessed in benchmark instances of the symmetrical version of RCC (SRCC) and the results achieved were always at least as good as the best known.

Future work includes the development of efficient parallel algorithms for tackling very large instances that may arise in real-life social networks. In addition, as the current integer linear programming formulations are still limited to small-size instances, there is still room for developing enhanced exact algorithms, perhaps similar to the combinatorial branch-and-bound of Brusco et al. [19], as an attempt to solve larger instances.

Chapter 3

Vehicle Routing Problem with Backhauls

In this chapter, two branch-cut-and-price (BCP) exact algorithms are depicted to solve the well-known *vehicle routing problem with backhauls* (VRPB) [37]. For the first time, all the literature instances are solved and BCP algorithms are proposed specifically to this problem. For future methodological advances, we also propose larger and challenging benchmark instances. Finally, we present three effective heuristics, where two of them take advantage, at different levels, of problem-specific information. One of the main contributions of this chapter is to show that it is worth considering specific characteristics of VRPB, both in exact and heuristic algorithms, in opposition to the idea that generic methods (e.g. algorithms for capacitated VRP or simultaneous pickup and delivery VRP) already achieve the best performance for the VRPB through a straightforward adaptation.

3.1 Introduction

In the classical *capacitated vehicle routing problem* (CVRP), introduced by Dantzig and Ramser [34], a homogeneous fleet of vehicles is considered to build a set of least-cost routes such that: (i) all customers are visited once by exactly one route, (ii) the capacity of the vehicles is respected, and (iii) each route starts and ends at the depot. Although some applications in distribution can be modeled as a CVRP, there are many applications with their own particularities such as those where customers require different types of services. The VRPB considers two types of customers: *linehaul* and *backhaul*.

The linehaul customers have a delivery demand which is loaded at the depot, whereas the backhaul customers have a pickup demand which should be transported to the depot. In the VRPB, necessarily a route must visit linehaul customers before backhaul customers. Moreover, at least one linehaul customer must be visited before possible backhaul customers, but a route may only be composed of linehauls. These restrictions allow avoiding en-route load rearrangements. For example, in beverage distribution, the collection of empty bottles should usually be performed after delivering full ones. Jacobs-Blecha and Goetschalckx [61] discussed how the grocery industry could save millions of dollars by exploiting backhauls. As in the CVRP, the objective is to minimize the total travel cost.

Koç and Laporte [66] presented a recent literature review of the VRPB, also covering variants such as the *mixed VRPB* (MVRPB), *VRPB with time windows* (VRPBTW) and the *heterogeneous fixed fleet VRPB* (HFFVRPB). Many studies have proposed (meta)heuristics for the VRPB. Constructive procedures were suggested in Deif and Bodin [37], Goetschalckx and Jacobs-Blecha [54], Jacobs-Blecha and Goetschalckx [61], and Toth and Vigo [112]. Osman and Wassan [85], Wassan [124] and Brandão [17] devised tabu search (TS) heuristics for the problem. Gajpal and Abad [51] proposed a heuristic called *multi-ant colony system* (MACS), whereas Zachariadis and Kiranoudis [127], put forward a local search-based heuristic that exploits a set of rich solution neighborhoods and makes use of auxiliary data structures to accelerate the move evaluation. Cuervo et al. [32] designed an *iterated local search* (ILS) algorithm that also makes use of auxiliary data structures in an oscillating local search. This method transits between the feasible and infeasible solution spaces through a dynamic mechanism of penalties. Brandão [16] implemented a deterministic ILS that, according to the author, is simple, fast, and almost parameter free. Belloso et al. [14] referred to VRPB as VRPCB (VRP with clustered backhauls) and proposed a biased-randomized metaheuristic framework (BRMF) on top of the popular Clarke and Wright heuristic [28]. Finally, unified VRP heuristics capable of solving the VRPB were devised by Ropke and Pisinger [99], Vidal et al. [122], and Christiaens and Vanden Berghe [27].

On the other hand, there are relatively few exact methods for the VRPB. Yano et al. [126] introduced a branch-and-bound algorithm for a particular case of the problem in which there should be at most four customers in a route. Goetschalckx and Jacobs-Blecha [54] proposed an integer linear programming (ILP) formulation which extends the model by Fisher and Jaikumar [49] for the CVRP. Toth and Vigo [113] proposed an ILP for the VRPB which is similar to the two index vehicle flow formulation for the asymmetric VRP by Laporte, Mercure, and Nobert [69]. The authors also devised a Lagrangian relaxation scheme which is strengthened by cutting planes. This relaxation is combined with another

The most important approaches in the literature for the VRPB.

Work	Year	Approach	Type
Deif and Bodin [37]	1984	Savings algorithm	Heuristic
Yano et al. [126]	1987	Branch-and-bound	Exact
Goetschalckx and Jacobs-Blecha [54]	1989	Integer linear programming formulation	Exact
Jacobs-Blecha and Goetschalckx [61]	1992	Generalized assignment-based heuristic	Heuristic
Toth and Vigo [112]	1996	Cluster-first-route-second algorithm	Heuristic
Toth and Vigo [113]	1997	Branch-and-bound + Lagrangian relaxation	Exact
Mingozzi, Giorgi, and Baldacci [82]	1999	Set partitioning-based approach	Exact
Geloğulları [53]	2001	Algorithm based on m-TSP relaxation	Exact
Osman and Wassan [85]	2002	Tabu search	Heuristic
Brandão [17]	2006	Tabu search	Heuristic
Ropke and Pisinger [99]	2006	Unified heuristic	Heuristic
Wassan [124]	2007	Tabu search	Heuristic
Gajpal and Abad [51]	2009	Multi-ant colony system	Heuristic
Zachariadis and Kiranoudis [127]	2012	Local search	Heuristic
Cuervo et al. [32]	2014	Iterated local search	Heuristic
Vidal et al. [122]	2014	Unified hybrid genetic search	Heuristic
Brandão [16]	2016	Deterministic iterated local search	Heuristic
Belloso et al. [14]	2017	Biased-randomized metaheuristic	Heuristic
Granada-Echeverri, Toro, and Santa [56]	2019	Integer linear programming formulation	Exact
Christiaens and Vanden Berghe [27]	2020	Ruin & recreate approach	Heuristic

one obtained by disregarding the capacity constraints of the model, producing an overall dual bounding procedure. Such procedure is used on a branch-and-bound algorithm to solve the VRPB to optimality. Mingozzi, Giorgi, and Baldacci [82] proposed a set partitioning (SP) formulation that makes use of variables for elementary paths over two subgraphs induced by the linehaul and backhaul customers, respectively. Two heuristics were combined to solve the dual problem and, through the resulting bound, they reduced the number of paths (variables) of the model without loss of optimality. Since the number of routes remained very large, an additional reduction was applied so that the resulting ILP could be solved using a MIP solver. Geloğulları [53] presented an exact algorithm for the asymmetric VRPB based on a relaxation named Multiple Traveling Salesman Problem (m-TSP). The method was not compared with the literature but tested on new randomly generated instances, involving up to 90 customers. Recently, an alternative mixed ILP for the VRPB was put forward by Granada-Echeverri, Toro, and Santa [56]. Table 3.1 summarizes, in chronological order, the most important approaches in the literature for the VRPB.

Koç and Laporte [66] pointed out the following future research perspective:

> *"The standard VRPB instances of Goetschalckx and Jacobs-Blecha [54] and Toth and Vigo [113] have been effectively solved by heuristics. However, it is our belief that further studies should focus on developing effective and powerful exact methods, such as branch-and-cut-and-price, to solve all available standard VRPB instances to optimality (see Poggi and Uchoa, 2014)."*

In view of this, this chapter proposes two BCP approaches for the VRPB. The algorithms incorporate elements of state-of-the-art BCP algorithms, such as rounded capacity cuts, limited-memory rank-1 cuts, strong branching, route enumeration, arc elimination using reduced costs and dual stabilization. As visible in our experiments, these methods can solve all instances from the literature to optimality, many of them for the first time. As a consequence, we decided to generate a novel and more challenging benchmark dataset with instances involving up to 1000 customers. Furthermore, we also report results for the VRPBTW and HFFVRPB thanks to a simple extension of one of our algorithms.

Concerning the heuristic solution of the VRPB, we conducted a study to evaluate the benefits of implementing specific procedures for the VRPB starting from a state-of-the-art matheuristic for the *asymmetric VRP with mixed backhauls* (AVRPMB). In other words, we are interested in investigating to what extent is it worth devising specific approaches that exploit the particularities of the VRPB given that one has a very competitive heuristic for the VRPMB. In this work, we implement three heuristic solution strategies for the VRPB. Extensive computational experiments were performed on classical VRPB benchmark instances and the three approaches were capable of obtaining all best known solutions and the result of one instance was improved. We also compare their performance and scalability for the new generated instances with up to 1000 customers.

The remainder of this chapter is organized as follows. Section 3.2 formally defines the problems considered in this chapter. Section 3.3 presents the set partitioning formulations used by the exact algorithms. Section 3.4 presents the proposed BCP algorithms. Section 3.5 presents the implemented heuristic strategies for the VRPB. Section 3.6 discusses the results of our extensive computational experiments on different benchmark instances. Finally, Section 3.7 concludes.

3.2 Problem definitions

In this section, the VRPB variants approached in this chapter are formally defined.

3.2.1 VRPB

Let $G = (V, A)$ be a directed graph and $V = \{0\} \cup L \cup B$, where vertex 0 represents the depot, while $L = \{1, 2, \dots, n\}$ and $B = \{n + 1, n + 2, \dots, n + m\}$ are the set of linehaul and backhaul vertices, respectively. Moreover, define $L_0 = L \cup \{0\}$ and $B_0 = B \cup \{0\}$, thus $A = A_L \cup A_{LB} \cup A_B$, such that:

- $A_L = \{(i,j) : i \in L_0, j \in L, i \neq j\}$,
- $A_{LB} = \{(i,j) : i \in L, j \in B_0\}$,
- $A_B = \{(i,j) : i \in B, j \in B_0, i \neq j\}$.

Graph G is not complete, since there are no arcs from B to L and no arcs from 0 to B. For each arc $a \in A$ there is a nonnegative traveling cost c_a. Let $\bar{V} = V \setminus \{0\}$ be the set of customers. Each vertex $j \in \bar{V}$ has a nonnegative d_j demand delivery (when $j \in L$) or pickup (when $j \in B$). Given a homogeneous fleet of K vehicles with capacity Q, the VRPB aims at finding K routes (elementary cycles in G passing by the depot) that minimize the total travel cost and satisfy the following constraints:

a) Each vertex $j \in \bar{V}$ must be visited by exactly one route.

b) A route has to visit linehaul customers before backhaul customers, i.e., after visiting a backhaul customer it is forbidden to visit a linehaul customer (implicit in the definition of G).

c) A route may only be composed by linehaul customers, but it cannot only be composed by backhaul customers (also implicit in the definition of G).

d) The sum of the delivery demands on a route does not exceed the vehicle capacity.

e) The sum of the pickup demands on a route does not exceed the vehicle capacity.

3.2.2 VRPBTW

The VRPBTW generalizes the VRPB by considering a time window $[a_i, b_i]$ and a service time s_i for each customer $i \in \bar{V}$. In the VRPBTW, the travel cost c_a of an arc a is interpreted as the travel time. A service can start to be performed from a_i until b_i, thus vehicles that arrive early must wait. Unlike in the VRPB, previous VRPBTW studies allowed routes containing only backhaul customers. Moreover, the number of vehicles is not specified *a priori*. We considered the hierarchical objective usually adopted by the main works in the literature [99, 111, 122], which prioritizes the minimization of the number of vehicles, and then the minimization of the total travel time. This objective is typical in situations where the fixed cost associated to vehicles or drivers is high. Although the algorithm proposed to tackle the VRPBTW can cope with the objective function adopted in the VRPB, the hierarchical objective allows for a fair comparison with previous works.

3.2.3 HFFVRPB

The HFFVRPB extends the VRPB by considering a finite set of vehicle types T, where each type $k \in T$ has u^k available vehicles with capacity Q^k and cost c_a^k, $\forall a \in A$. The composition of the heterogeneous fleet must respect the availability of each type of vehicle, but without necessarily using all vehicle types. We follow the same objective function as [114] and [89], which consists of minimizing the total travel cost.

3.3 Set partitioning formulations

Before introducing the SP-based formulations, we first present formulation $\mathcal{F}0$ by Toth and Vigo [113], in Equations (3.1)–(3.7). We define variable x_a equal to 1 if the arc $a \in A$ is traversed by some vehicle, otherwise it is equal to 0. Given a subset S of L or B, let $r(S) = \lceil \sum_{i \in S} d_i / Q \rceil$ be a lower bound on the minimum number of vehicles necessary to serve all customers in S. Also, let $\delta^-(S) = \{(i,j) \in A : i \in V \setminus S, j \in S\}$ and $\delta^+(S) = \{(i,j) \in A : i \in S, j \in V \setminus S\}$. For simplicity, let $\delta^-(\{i\}) = \delta^-(i)$ and $\delta^+(\{i\}) = \delta^+(i)$, $\forall i \in V$.

$$(\mathcal{F}0)\ \text{Min} \sum_{a \in A} c_a x_a \tag{3.1}$$

$$\text{s.t.} \sum_{a \in \delta^-(i)} x_a = 1 \qquad \forall i \in \bar{V}, \tag{3.2}$$

$$\sum_{a \in \delta^+(i)} x_a = 1 \qquad \forall i \in \bar{V}, \tag{3.3}$$

$$\sum_{a \in \delta^+(0)} x_a = K, \tag{3.4}$$

$$\sum_{a \in \delta^-(S)} x_a \geq r(S) \qquad \forall S \subseteq L, \tag{3.5}$$

$$\sum_{a \in \delta^-(S)} x_a \geq r(S) \qquad \forall S \subseteq B, \tag{3.6}$$

$$x_a \in \{0,1\} \qquad \forall a \in A. \tag{3.7}$$

Constraints (3.2)–(3.3) ensure that each customer is visited exactly once, while constraint (3.4) imposes that K vehicles must leave the depot. Constraints (3.5)–(3.6) are the *rounded capacity constraints* (RCC) and also guarantee subtour elimination. They are usually separated in a cutting plane fashion. Constraints (3.7) define the domain of the variables.

In what follows, we describe two SP formulations for the VRPB by extending $\mathcal{F}0$. Both formulations are compared in terms of linear relaxation and effectiveness of the application of rank-1 cuts.

3.3.1 Formulation $\mathcal{F}1$

Let Ω be the set of all q-routes in G, which are walks (paths that may be not elementary, i.e., a customer can be visited more than once) starting and ending at the depot and that do not violate the capacity constraints for both linehaul and backhaul customers. A customer $i \in \bar{V}$ visited k times consumes $k \times d_i$ load units. Let h_a^p be the number of times a q-path $p \in \Omega$ traverses the arc $a \in A$ and λ_p a binary variable which will take the value 1 if and only if q-route p belongs to the solution. $\mathcal{F}0$ can be extended by adding variables λ and constraints (3.8)–(3.9):

$$x_a = \sum_{p \in \Omega} h_a^p \lambda_p \qquad \forall a \in A, \tag{3.8}$$

$$\lambda_p \in \{0,1\} \qquad \forall p \in \Omega. \tag{3.9}$$

Formulation $\mathcal{F}1$ is then given by (3.1)–(3.9). Note that only elementary q-routes can take part of integer solutions because of constraint (3.2). By replacing the x variables using (3.8) and relaxing the integrality constraints, one obtains the following linear relaxation of $\mathcal{F}1$:

$$\text{Min} \sum_{p \in \Omega} \left(\sum_{a \in A} c_a h_a^p \right) \lambda_p \tag{3.10}$$

$$\text{s.t.} \sum_{a \in \delta^-(i)} \sum_{p \in \Omega} h_a^p \lambda_p = 1 \qquad \forall i \in \bar{V}, \tag{3.11}$$

$$\sum_{a \in \delta^+(0)} \sum_{p \in \Omega} h_a^p \lambda_p = K, \tag{3.12}$$

$$\sum_{a \in \delta^-(S)} \sum_{p \in \Omega} h_a^p \lambda_p \geq r(S) \qquad \forall S \subseteq L, \tag{3.13}$$

$$\sum_{a \in \delta^-(S)} \sum_{p \in \Omega} h_a^p \lambda_p \geq r(S) \qquad \forall S \subseteq B, \tag{3.14}$$

$$\lambda_p \geq 0 \qquad \forall p \in \Omega. \tag{3.15}$$

Constraints (3.13)–(3.14) are not necessary for correctness because any integer solution satisfying (3.11)-(3.12) corresponds to K feasible elementary q-routes. Nevertheless, they can cut fractional solutions and are important to strengthen the formulation.

Such constraints are added in a cutting plane fashion. On the other hand, the constraints that would be obtained from (3.3) are now completely redundant and can be dropped. In this kind of SP-based formulation, it is common to use relaxations such as q-routes instead of elementary routes, because the pricing subproblem becomes weakly $\mathcal{NP}$-hard and thus more computationally tractable [94]. The disadvantage, on the other hand, is that this worsens the linear relaxation.

3.3.2 Formulation $\mathcal{F}2$

In the SP-based formulation by Mingozzi, Giorgi, and Baldacci [82], there are variables associated with paths with only linehaul or backhaul customers. There are additional binary variables, one for each arc in A_{LB}, used in constraints that ensure that linehaul and backhaul paths should be connected to form a complete feasible route. We now describe a new formulation $\mathcal{F}2$ which follows a similar principle but does not use additional variables.

Let $G_L = (L_0, A_L)$ and $G_B = (L \cup B_0, A_{LB} \cup A_B)$ be subgraphs of G and let Ω_L and Ω_B be the sets of q-paths over G_L and G_B, respectively. For G_L, the q-paths are walks that start at the depot and end at some customer in L, not violating the linehaul capacity constraint. For G_B, the q-paths are walks that start at a linehaul customer and end at the depot, not violating the backhaul capacity constraint. The q-paths in Ω_B contain exactly one linehaul customer, which will be interpreted as *connecting vertices*. Given $i \in L$, the subset $\Omega_L^i \subseteq \Omega_L$ is composed by paths ending at i and $\Omega_B^i \subseteq \Omega_B$ by paths starting at i. A binary variable λ_p^L (λ_p^B) is equal to 1 if q-route $p \in \Omega_L$ ($p \in \Omega_B$) belongs to the solution, and equal to 0, otherwise. The constant h_a^p indicates how many times arc a appears in q-path p (it is necessarily zero when a and p are associated with distinct graphs: a q-path in a graph will never traverse an arc of the other graph). Formulation $\mathcal{F}0$ can be extended by including variables λ^L and λ^B, as well as constraints (3.16)–(3.19). Constraints (3.17), in particular, ensures that the chosen paths are properly connected.

$$x_a = \sum_{p \in \Omega_L} h_a^p \lambda_p^L + \sum_{p \in \Omega_B} h_a^p \lambda_p^B \qquad \forall a \in A, \tag{3.16}$$

$$\sum_{p \in \Omega_L^i} \lambda_p^L = \sum_{p \in \Omega_B^i} \lambda_p^B \qquad \forall i \in L, \tag{3.17}$$

$$\lambda_p^L \in \{0,1\} \qquad \forall p \in \Omega_L, \tag{3.18}$$

$$\lambda_p^B \in \{0,1\} \qquad \forall p \in \Omega_B. \tag{3.19}$$

Hence, $\mathcal{F}2$ is defined by (3.1)–(3.7) and (3.16)–(3.19). By eliminating the x variables using (3.16), relaxing the integrality constraints and performing some simplifications, it is possible to write the linear relaxation of $\mathcal{F}2$ as follows:

$$\begin{aligned}
\text{Min} \quad & \sum_{p\in\Omega_L}\left(\sum_{a\in A} c_a h_a^p\right)\lambda_p^L + \sum_{p\in\Omega_B}\left(\sum_{a\in A} c_a h_a^p\right)\lambda_p^B && && (3.20)\\
\text{s.t.} \quad & \sum_{a\in\delta^-(i)}\left(\sum_{p\in\Omega_L} h_a^p\lambda_p^L + \sum_{p\in\Omega_B} h_a^p\lambda_p^B\right) = 1 && \forall i\in\bar{V}, && (3.21)\\
& \sum_{a\in\delta^+(0)}\sum_{p\in\Omega_L} h_a^p\lambda_p^L = K, && && (3.22)\\
& \sum_{p\in\Omega_L^i}\lambda_p^L = \sum_{p\in\Omega_B^i}\lambda_p^B && \forall i\in L, && (3.23)\\
& \sum_{a\in\delta^-(S)}\sum_{p\in\Omega_L} h_a^p\lambda_p^L \geq r(S) && \forall S\subseteq L, && (3.24)\\
& \sum_{a\in\delta^-(S)}\sum_{p\in\Omega_B} h_a^p\lambda_p^B \geq r(S) && \forall S\subseteq B, && (3.25)\\
& \lambda_p^L \geq 0 && \forall p\in\Omega_L, && (3.26)\\
& \lambda_p^B \geq 0 && \forall p\in\Omega_B. && (3.27)
\end{aligned}$$

As in formulation $\mathcal{F}1$, constraints (3.24)–(3.25) should be dynamically added via cutting planes. The constraints that would be derived from (3.3) become redundant and can be dropped.

3.3.3 Strengthening the formulations

Before comparing the formulations, we will describe how to strengthen them through ng-routes and rank-1 cuts.

ng-routes

Strengthening the route relaxation without significantly affecting the complexity of the pricing subproblem is a challenging task. One of the most successful route relaxation schemes is the so-called ng-routes, introduced by Baldacci, Mingozzi, and Roberti [10] as an alternative to q-routes. Analogously, ng-paths can be defined as an alternative to q-paths. For each customer $i \in \bar{V}$, let $N_i \subseteq \bar{V}$ be the neighborhood of $i \in \bar{V}$ (a.k.a. ng-set), where N_i is typically composed by the closest customers to i. In a ng-route (or ng-path), a customer i can be revisited only after visiting a customer j such that $i \notin N_j$.

The size of the ng-sets controls the level of elementarity obtained, since larger sets allow fewer non-elementary routes. In one extreme, if ng-sets are empty, ng-routes are q-routes. On the other extreme, if all ng-sets are equal to $\bar{V}$, then ng-routes are elementary. In practice, ng-sets of size around 8 to 10 offer a good trade-off between formulation strength and complexity of the column generation procedure.

In the VRPB, it only makes sense to define ng-sets with customers of the same type: if $i \in L$ then $N_i \subseteq L$, while if $i \in B$ then $N_i \subseteq B$. Formulation $\mathcal{F}1$ can be strengthened by restricting Ω to ng-routes. Similarly, $\mathcal{F}2$ can be strengthened by restricting Ω_L and Ω_B to ng-paths.

Rank-1 cuts

By applying the Chvátal-Gomory rounding over the sum of inequalities (3.11) multiplied by $\rho \in \mathbb{R}^{|\bar{V}|}_{\geq 0}$, we obtain the rank-1 cut (3.28), which is valid for $\mathcal{F}1$.

$$\sum_{p \in \Omega} \left\lfloor \sum_{i \in \bar{V}} \sum_{a \in \delta^-(i)} \rho_i h_a^p \right\rfloor \lambda_p \leq \left\lfloor \sum_{i \in \bar{V}} \rho_i \right\rfloor \tag{3.28}$$

Analogously, the rank-1 cut (3.29), which is valid for $\mathcal{F}2$, can be derived from (3.21).

$$\sum_{p \in \Omega_L} \left\lfloor \sum_{i \in L} \sum_{a \in \delta^-(i)} \rho_i h_a^p \right\rfloor \lambda_p^L + \sum_{p \in \Omega_B} \left\lfloor \sum_{i \in B} \sum_{a \in \delta^-(i)} \rho_i h_a^p \right\rfloor \lambda_p^B \leq \left\lfloor \sum_{i \in \bar{V}} \rho_i \right\rfloor \tag{3.29}$$

Rank-1 cuts are a generalization of the Subset Row Cuts [62] and are known to be very strong, but separating them makes the pricing subproblems significantly more difficult. Hence, we use the limited memory technique proposed by [87] to mitigate their impact on the pricing. This technique consists in using a memory mechanism to define the coefficients of the variables of the cut. As a consequence, it becomes possible to control the strength of the cut and its impact on the structure of the subproblem.

3.3.4 Comparing $\mathcal{F}1$ and $\mathcal{F}2$

In this subsection, we assume that $\mathcal{F}1$ and $\mathcal{F}2$ use ng-routes and ng-paths defined over the same ng-sets.

Proposition 1. The linear relaxations of $\mathcal{F}1$ and $\mathcal{F}2$ are equally strong.

Proof. Let P_1 and P_2 be the polyhedra defined by the linear relaxations of $\mathcal{F}1$ and $\mathcal{F}2$, respectively. We show that for any solution of P_1 there is a solution of P_2 with the same objective value, and vice versa.

Given a solution $\bar{\lambda} \in P_1$, the function described in Algorithm 4 returns a solution $\mathtt{P}_2(\bar{\lambda}) = (\bar{\lambda}^L, \bar{\lambda}^B)$ in $\mathcal{F}2$ space. It is clear from lines 7–8 that constraints (3.17) are satisfied by that solution. It can be verified through inequalities (3.8) and (3.16) that both $\bar{\lambda}$ and $\mathtt{P}_2(\bar{\lambda})$ induce the same values for the arc variables x. This is true because an arc $a \in A$ can be part of paths either from Ω_L or Ω_B, but never from both sets. As $\bar{\lambda} \in P_1$, then the x solution satisfies (3.2)–(3.6). So, $\mathtt{P}_2(\bar{\lambda})$ should satisfy the corresponding constraints (3.21)–(3.25) and belongs to P_2. Moreover, $\bar{\lambda}$ and $\mathtt{P}_2(\bar{\lambda})$ have the same cost.

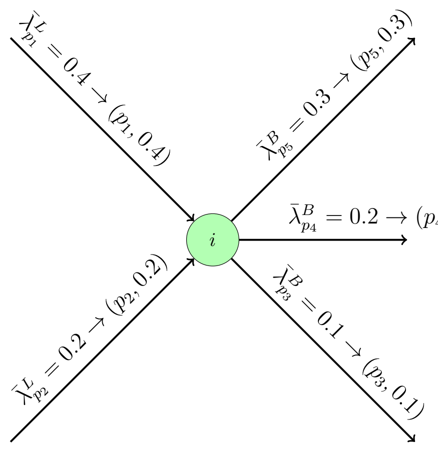
Let $(\bar{\lambda}^L, \bar{\lambda}^B)$ be a solution in P_2. The function described in Algorithm 5 returns a solution $\mathtt{P}_1(\bar{\lambda}^L, \bar{\lambda}^B) = \bar{\lambda}$ in $\mathcal{F}1$ space (Figure 3.1 illustrates how the algorithm works for a certain connecting vertex i). Note that lines 9 and 12 (that assume the existence of a suitable path p^2 to complete path p^1) are only correct because constraints (3.17) are satisfied by $(\bar{\lambda}^L, \bar{\lambda}^B)$. Again, it can be verified through inequalities (3.8) and (3.16) that both solutions $(\bar{\lambda}^L, \bar{\lambda}^B)$ and $\mathtt{P}_1(\bar{\lambda}^L, \bar{\lambda}^B)$ yield the same values for the arc variables x, so the latter solution belongs to P_1 and they have the same cost. □

Algorithm 4: Obtains the solution $(\bar{\lambda}^L, \bar{\lambda}^B) \in P_2$ corresponding to $\bar{\lambda} \in P_1$

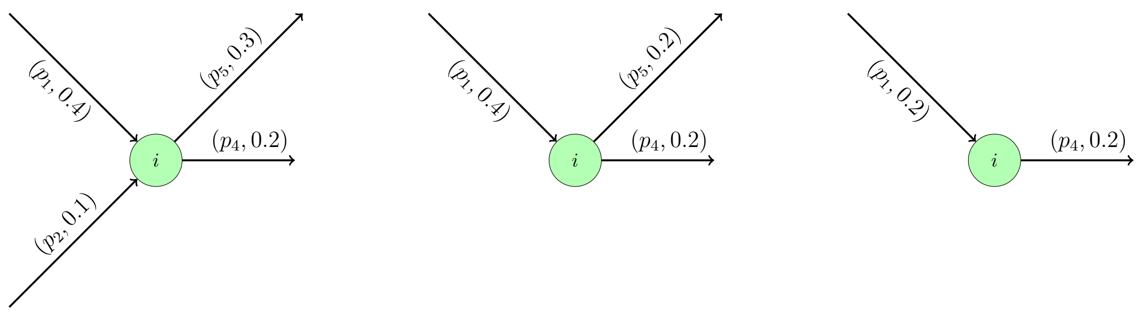
1 **Function** $\mathtt{P_2}(\bar{\boldsymbol{\lambda}})$
2   Let $\gamma = \{(p, \bar{\lambda}_p) : p \in \Omega,\ \bar{\lambda}_p > 0\}$ be the set that maps the routes to their values
3   Let $L(p) \in \Omega_L$ and $B(p) \in \Omega_B$ be the paths obtained by splitting route $p \in \Omega$ in its connecting vertex (the last linehaul customer)
4   Let $(\bar{\lambda}^L, \bar{\lambda}^B)$ be the solution to be built for P_2, such that $\bar{\lambda}^L_p$ is initially zero $\forall p \in \Omega_L$ and $\bar{\lambda}^B_p$ is initially zero $\forall p \in \Omega_B$
5   **while** $\gamma \neq \emptyset$ **do**
6   Let (p, ζ) be a pair in γ
7         $\bar{\lambda}^L_{L(p)} = \bar{\lambda}^L_{L(p)} + \zeta$
8         $\bar{\lambda}^B_{B(p)} = \bar{\lambda}^B_{B(p)} + \zeta$
9         $\gamma = \gamma \setminus \{(p, \zeta)\}$ `// Remove` p
10   **return** $(\bar{\lambda}^L, \bar{\lambda}^B)$

The functions defined in Algorithm 4 and Algorithm 5 define a one-to-one correspondence between solutions in P_1 and P_2. In fact, for all $\bar{\lambda} \in P_1$, $\mathtt{P}_1(\mathtt{P}_2(\bar{\lambda})) = \bar{\lambda}$; for all $(\bar{\lambda}^L, \bar{\lambda}^B) \in P_2$, $\mathtt{P}_2(\mathtt{P}_1((\bar{\lambda}^L, \bar{\lambda}^B))) = (\bar{\lambda}^L, \bar{\lambda}^B)$. That correspondence will also be used in the proof of the following result.

Proposition 2. Rank-1 cuts (3.28) are at least as strong as (3.29) and may be strictly stronger.



(a) $\bar{\lambda}_{p_2 \oplus p_3} = 0.1$



(b) $\bar{\lambda}_{p_2 \oplus p_5} = 0.1$ (c) $\bar{\lambda}_{p_1 \oplus p_5} = 0.2$ (d) $\bar{\lambda}_{p_1 \oplus p_4} = 0.2$

Figure 3.1: Illustration of Algorithm 5. Obtaining the values for $\bar{\lambda}_p$, such that $p \in \Omega$ and the connecting vertex of p is $i \in L$. In Figure 3.1a, paths p_3 and p_2 are chosen according to the lines 7 and 12, respectively. Next, the value for $\lambda_{p_2 \oplus p_3} = 0.1$ is defined, the pair $(p_3, 0.1)$ is removed from γ^i and $(p_2, 0.2)$ is updated to $(p_2, 0.1)$. Figures 3.1b, 3.1c and 3.1d illustrate the continuation of the algorithm, until γ^i is empty. The algorithm performs this process for every vertex $i \in L$ as connecting vertex.

Algorithm 5: Obtains the solution $\bar{\lambda} \in P_1$ corresponding to $(\bar{\lambda}^L, \bar{\lambda}^B) \in P_2$

Function $\mathtt{P_1}(\bar{\boldsymbol{\lambda}}^{\boldsymbol{L}}, \bar{\boldsymbol{\lambda}}^{\boldsymbol{B}})$
  Let $\gamma^i = \{(p, \bar{\lambda}_p^L) : p \in \Omega_L^i,\ \bar{\lambda}_p^L > 0\} \cup \{(p, \bar{\lambda}_p^B) : p \in \Omega_B^i,\ \bar{\lambda}_p^B > 0\}$, $i \in L$, be the sets that maps the paths related to each connecting vertex i to their values
  Let $p_l \oplus p_b$ be the route in Ω obtained by concatenating the paths $p_l \in \Omega_L$ and $p_b \in \Omega_B$
  Let $\bar{\lambda}$ be the solution to be built for P_1, such that $\bar{\lambda}_p$ is initially zero $\forall p \in \Omega$
  for $i \in L$ **do**
    while $\gamma^i \neq \emptyset$ **do**
      Let (p^1, ζ^1) be a pair in γ^i whose ζ^1 is minimum
      if $p^1 \in \Omega_L^i$ **then**
        Let (p^2, ζ^2) be any pair in γ^i such that $p^2 \in \Omega_B^i$
        $\bar{\lambda}_p = \zeta^1$, such that $p = p^1 \oplus p^2$
      else // $p^1 \in \Omega_B^i$
        Let (p^2, ζ^2) be any pair in γ^i such that $p^2 \in \Omega_L^i$
        $\bar{\lambda}_p = \zeta^1$, such that $p = p^2 \oplus p^1$
      $\gamma^i = \gamma^i \setminus \{(p^1, \zeta^1), (p^2, \zeta^2)\}$ // Remove p^1 and p^2
      if $\zeta^2 - \zeta^1 > 0$ **then**
        $\gamma^i = \gamma^i \cup \{(p^2, \zeta^2 - \zeta^1)\}$ // Reinsert p^2 with updated value
  return $\bar{\lambda}$

Proof. Consider the rank-1 cuts (3.28) and (3.29) corresponding to the same vector of multipliers ρ. Consider a path $p \in \Omega$ and its split paths $L(p) \in \Omega_L$ and $B(p) \in \Omega_B$. It is always true that:

$$\sum_{i \in \bar{V}} \sum_{a \in \delta^-(i)} \rho_i h_a^p = \sum_{i \in L} \sum_{a \in \delta^-(i)} \rho_i h_a^{L(p)} + \sum_{i \in B} \sum_{a \in \delta^-(i)} \rho_i h_a^{B(p)}. \tag{3.30}$$

If the condition

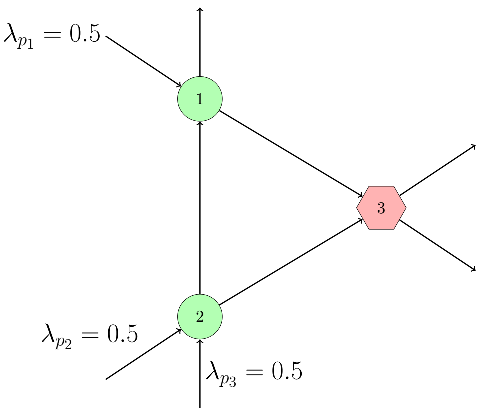
$$\left\lfloor \sum_{i \in \bar{V}} \sum_{a \in \delta^-(i)} \rho_i h_a^p \right\rfloor = \left\lfloor \sum_{i \in L} \sum_{a \in \delta^-(i)} \rho_i h_a^{L(p)} \right\rfloor + \left\lfloor \sum_{i \in B} \sum_{a \in \delta^-(i)} \rho_i h_a^{B(p)} \right\rfloor \tag{3.31}$$

is true for all $p \in \Omega$, then rank-1 cuts (3.28) and (3.29) are equally strong, in the sense that a solution $\bar{\lambda} \in P_1$ is cut by (3.28) if and only if the corresponding solution $\mathtt{P}_2(\bar{\lambda}) = (\bar{\lambda}^L, \bar{\lambda}^B)$ is cut by (3.29). Otherwise, if for some $p \in \Omega$ the left-hand-side of (3.31) is strictly larger than its right-hand-side, then (3.28) is strictly stronger than (3.29).

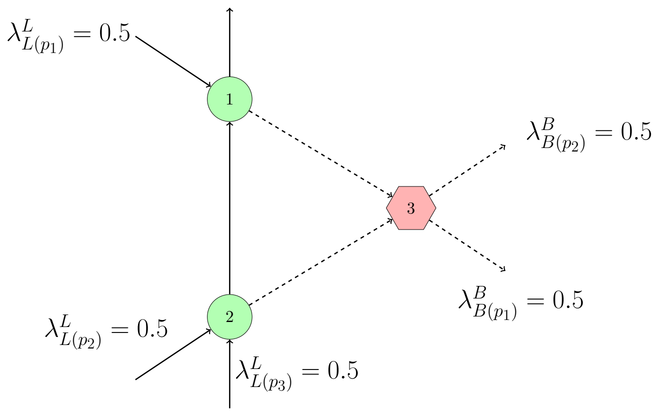
Let $C = \{i \in \bar{V} : \rho_i > 0\}$. If $C \subseteq L$, the second term in the right-hand-side of (3.30) is zero. So (3.31) is true and (3.28) and (3.29) are equally strong. The coefficient of λ_p in (3.28) will be identical to the coefficient of $\lambda_{L(p)}^L$ in (3.29), while the coefficient of $\lambda_{B(p)}^B$ will be zero. A similar reasoning shows that when $C \subseteq B$, rank-1 cuts (3.28) and (3.29) are also equally strong.

On the other hand, when C has customers of both types, (3.31) may not be true.

Figure 3.2 illustrates an example of rank-1 cut (a 3-Subset Row Cut), when $\rho_i = 1/2$ for $i \in C$, where C is composed by linehaul customers 1 and 2 and by backhaul customer 3. In this example, (3.28) cuts the fractional solution $\bar{\lambda} \in P_1$ (route p_1 passes by customers 1 and 3, route p_2 by 2 and 3, and route p_3 by 2 and 1) but the corresponding solution $\mathtt{P}_2(\bar{\lambda}) = (\bar{\lambda}^L, \bar{\lambda}^B)$ is not cut by (3.29). □



(a) $\lambda_{p_1} + \lambda_{p_2} + \lambda_{p_3} \leq 1$



(b) $\lambda^L_{L(p_3)} \leq 1$

Figure 3.2: Example of rank-1 cut with both types of customers, where the hexagon represents the backhaul customer. Note that the cut in 3.2a is effective, but 3.2b is not.

In the Appendix I, we also show that $\mathcal{F}1$, $\mathcal{F}2$ and Mingozzi, Giorgi, and Baldacci [82] SP formulations are equally strong.

3.4 Branch-cut-and-price algorithms

This section describes BCP$_{\mathcal{F}1}$ and BCP$_{\mathcal{F}2}$, two BCP algorithms for the VRPB based on $\mathcal{F}1$ and $\mathcal{F}2$, respectively. More precisely, we discuss elements related to pricing, cut generation, branching and path enumeration. Furthermore, we also describe how BCP$_{\mathcal{F}1}$ can be adapted to solve the HFFVRPB and VRPBTW.

3.4.1 Pricing subproblem

In both BCP algorithms, the pricing subproblems are modeled as a *resource constrained shortest path problem* (RCSP) which is defined as follows. Let $\mathcal{G} = (\mathcal{V}, \mathcal{A})$ be a directed graph, where $\mathcal{V}$ is the set of vertices, $\mathcal{A}$ the set of arcs and $\bar{c}_a \in \mathbb{R}$ is the cost of the arc $a \in \mathcal{A}$. $\mathcal{V}$ has special nodes v_{source} and v_{sink}, they can be the same vertex or two distinct vertices. For each arc $a \in \mathcal{A}$, there exists a resource consumption $q_a \in \mathbb{R}_+$. Also, an interval $[l_i, u_i]$ is associated with each vertex $i \in \mathcal{V}$. A resource constrained path $p = (v_{source} = v_0, v_1, \ldots, v_{k-1}, v_{sink} = v_k)$ over $\mathcal{G}$ is feasible if $k \geq 1$, $v_j \neq v_{source}, v_j \neq v_{sink}$, $1 \leq j \leq k-1$, and the accumulated resource consumption S_j at visit j, $0 \leq j \leq k$, where $S_0 = 0$ and $S_j = \max\{l_{v_j}, S_{j-1} + q_{(v_{j-1}, v_j)}\}$, does not exceed u_{v_j}. Note that this definition allows to "drop out resources" (i.e., to consume $l_{v_j} - S_{j-1} + q_{(v_{j-1}, v_j)}$ of the resource) to satisfy the lower bound l_i at a vertex i, if $l_{v_j} > S_{j-1} + q_{(v_{j-1}, v_j)}$. On the other hand, the upper limits on accumulated resource consumption are strict. The RCSP objective is to find a resource-constrained path with minimum cost.

RCSP graph for BCP$_{\mathcal{F}1}$

For BCP$_{\mathcal{F}1}$, the RCSP graph $\mathcal{G} = (\mathcal{V}, \mathcal{A}) = (V, A) = G$; $v_{source} = v_{sink} = 0$. Each arc $a = (i, j) \in \mathcal{A}$ has a capacity resource consumption given by $q_a = d_j$ and each vertex $i \in \mathcal{V}$ has a resource interval defined as:

$$[l_i, u_i] = \begin{cases} [0, 2Q], i = 0 \\ [d_i, Q], i \in L \\ [Q + d_i, 2Q], i \in B \end{cases}$$

Figure 3.3 illustrates the RCSP graph for BCP$_{\mathcal{F}1}$. It can be seen that a resource constrained path in that graph can visit customers in L until the capacity limit Q is reached. However, when the path visits the first backhaul customer, the values of l_i for $i \in B$ force any unused linehaul capacity to be dropped. Therefore, the total backhaul capacity is also limited by Q. The cost of an arc $\bar{c}_a$ is the reduced cost calculated through the dual variables associated with constraints (3.11)–(3.14).

RCSP graph for BCP$_{\mathcal{F}2}$

For BCP$_{\mathcal{F}2}$ there are two RCSP graphs. The first RCSP graph is $\mathcal{G}_L = (\mathcal{V}_L, \mathcal{A}_L)$, where $\mathcal{V}_L = L_0 \cup \{0'\}$ and $\mathcal{A}_L = A_L \cup \{(i, 0') : i \in L\}$; $v_{source} = 0$ and $v_{sink} = 0'$. Each arc $a = (i, j) \in \mathcal{A}_L$ has a capacity resource consumption given by $q_a = d_j$ (assuming that $d_{0'} = 0$) and each vertex $i \in \mathcal{V}_L$ has resource consumption interval $[d_i, Q]$.

The second RCSP graph is $\mathcal{G}_B = (\mathcal{V}_B, \mathcal{A}_B)$, where $\mathcal{V}_B = \{0'\} \cup L \cup B_0$ and $\mathcal{A}_B = A_{LB} \cup A_B \cup \{(0', i) : i \in \bar{V}\}$; $v_{source} = 0'$ and $v_{sink} = 0$. Each arc $a = (i, j) \in \mathcal{A}_B$ has a capacity resource consumption given by $q_a = d_j$ (assuming that $d_0 = 0$) and each vertex $i \in \mathcal{V}_B$ has a resource interval $[d_i, Q]$.

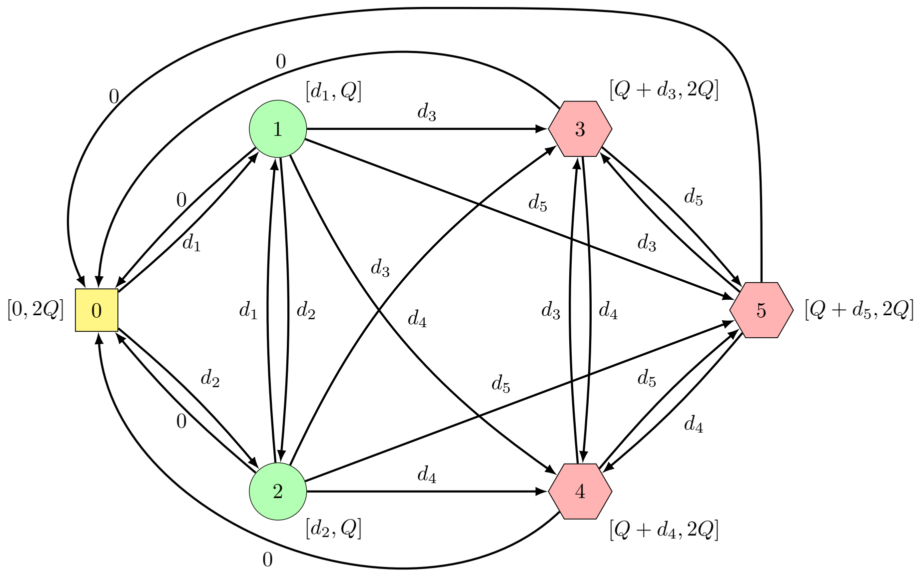


Figure 3.3: RCSP graph for $\mathrm{BCP}_{\mathcal{F}1}$.

Figure 3.4 illustrates the two RCSP graphs for $\mathrm{BCP}_{\mathcal{F}2}$. The cost of an arc $\bar{c}_a$ for both graphs is the reduced cost calculated through the dual variables associated with constraints (3.21)–(3.25).

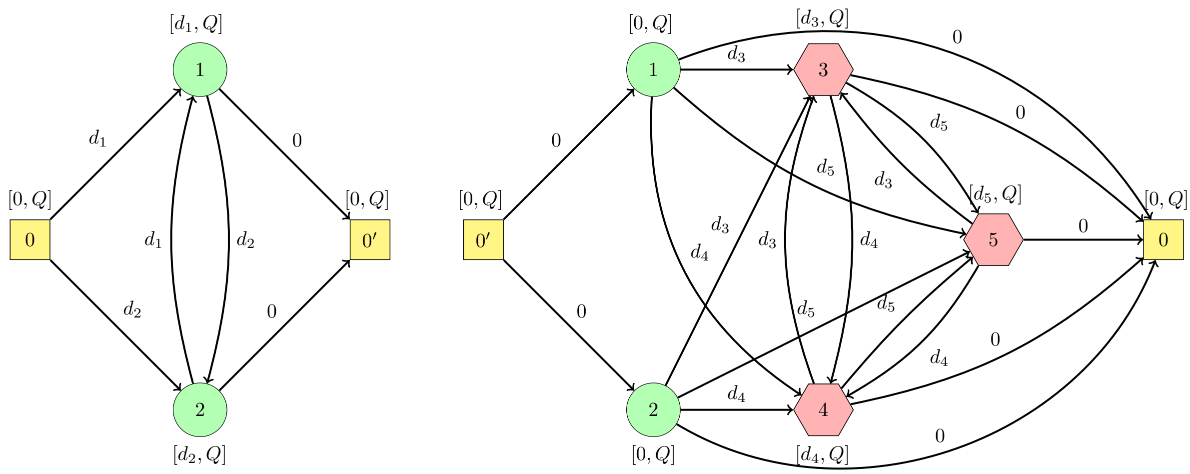


Figure 3.4: RCSP graphs for $\mathrm{BCP}_{\mathcal{F}2}$

Solving the pricing subproblems

The RCSP problems above defined are solved by a labeling algorithm, using the bucket graph based variant proposed by Sadykov, Uchoa, and Pessoa [100]. Such algorithm also handles ng-routes (for $\mathrm{BCP}_{\mathcal{F}1}$) and ng-paths (for $\mathrm{BCP}_{\mathcal{F}2}$). In both cases the

ng-sets have cardinality 8. As mentioned before, the ng-set of a linehaul customer only has linehaul customers, and the ng-set of a backhaul customer only has backhaul customers. Moreover, the labeling algorithm also considers the modification in the reduced costs induced by the dual variables of the limited memory rank-1 cuts added.

The big advantage of $\mathcal{F}2$ over $\mathcal{F}1$ is that it reduces the time spent solving pricing problems, the usual bottleneck of the BCP algorithms. In a labeling algorithm for the RCSP, the number of undominated labels grows more than linearly with the size of the paths (in fact, exponentially in the worst case). Therefore, solving two RCSPs with capacity limit Q (associated with the paths in Ω^L and Ω^B) is typically much faster than solving a single RCSP with limit $2Q$ (associated with the longer routes in Ω).

3.4.2 Cut generation, branching and path enumeration

In both BCP algorithms, rounded capacity cuts are separated by the heuristic procedure available in CVRPSEP [78]. Fractional solutions of $\mathcal{F}1$ and $\mathcal{F}2$ are first converted to arc variables x in order to perform that separation.

Limited memory rank-1 cuts are separated for sets C, such that $|C| \leq 5$, using the optimal multipliers given in Pecin et al. [88]. As shown in Proposition 2, rank-1 cuts for $\mathcal{F}2$ where C has both linehaul and backhaul customers are weak and not likely to be violated. This is the main potential disadvantage of $\mathcal{F}2$ over $\mathcal{F}1$.

In both BCP algorithms, branching is performed over aggregations of arc variables. For a pair of vertices i and j in V, $i < j$, $y_{ij} = x_{ij} + x_{ji}$ (if $(j, i) \notin A$, $y_{ij} = x_{ij}$) should be integer. A fractional y_{ij} is chosen by a strong branching procedure similar to the one in Pecin et al. [88].

Both BCP algorithms may also perform route enumeration, as in Baldacci, Christofides, and Mingozzi [9] and Contardo and Martinelli [29], when the gap between a node lower bound and the upper bound is sufficiently small. This means that all elementary q-routes in Ω (for BCP$_{\mathcal{F}1}$) or all elementary q-paths in Ω^L and Ω^B (for BCP$_{\mathcal{F}1}$) with reduced cost not higher than the gap are enumerated and stored in a pool. After that, the pricing is performed by inspection, which can save a lot of time. As the lower bounds increase, fixing by reduced cost reduces the size of the pools. Eventually, the pool size becomes small enough so that the restricted $\mathcal{F}1$ (or $\mathcal{F}2$) can be solved using a MIP solver, thus finishing the node. The enumeration is another significant potential advantage of $\mathcal{F}2$ over $\mathcal{F}1$. As there are much fewer paths in Ω^L and Ω^B than in Ω, it is possible to perform enumeration in $\mathcal{F}2$ earlier, with a larger gap.

3.4.3 VRPBTW and HFFVRPB

The BCP$_{\mathcal{F}1}$ approach can be directly adapted to solve the VRPBTW. This only requires an additional time resource. For a given arc $a = (i, j)$ in the RCSP graph, the consumption of this resource is $c_{ij} + s_i$. The resource consumption interval for that resource in each vertex is the associated customer time window. The hierarchical objective of the VRPBTW can be handled by running the algorithm for different values of K. Initially, $K = K^* - 1$, where K^* is the number of vehicles for the best known solution (BKS). The value of K is then iteratively decremented until the problem becomes infeasible (the last feasible solution found is the optimal one). If no feasible solution is found for $K < K^*$, the BCP algorithm must be executed with K^* using the total travel time of the BKS as upper bound (note that this bound is not valid if $K < K^*$).

On the other hand, $\mathcal{F}2$ cannot be adapted to solve the VRPBTW. This is due to the fact that the time resource is global, in the sense that it can not be split *a priori* between linehaul and backhaul customers (in contrast, there are separated capacities Q for linehaul and backhaul customers).

In order to adapt BCP$_{\mathcal{F}1}$ to the HFFVRPB, it is necessary to define a distinct RCSP graph for each type of vehicle $k \in T$, where each graph has specific arc costs. Constraints (3.12) should now limit the number of available vehicles for each vehicle type, as specified in the problem instance. Moreover, the value of $r(S)$ in the rounded capacity cuts (3.13) and (3.14) must be defined by means of $\max_{k \in T} Q^k$ instead of Q.

Adapting BCP$_{\mathcal{F}2}$ to the HFFVRPB would require a larger number of connecting constraints, like (3.23), to ensure that only linehaul and backhaul paths corresponding to the same vehicle type are connected. We therefore decided not to test BCP$_{\mathcal{F}2}$ for this variant.

3.5 Heuristic solution strategies

The VRPB can be seen as a special case of the *asymmetric VRP with mixed backhauls* (AVRPMB) where, in the latter, the precedence constraints are relaxed (i.e., backhauls can be visited before linehauls) and a non-empty route is allowed to have only backhauls. Moreover, the AVRPMB is a particular case of the *asymmetric VRP with simultaneous* *pickup and delivery* (VRPSPD). A number of methods were proposed for the symmetric version of these two problems. Note that a method capable of solving the AVRPMB, can be directly applied to the VRPB by penalizing the infeasible arcs of the latter problem. According to the survey by Battarra, Cordeau, and Iori [12], one of the best algorithms proposed for such problems is the matheuristic by Subramanian, Uchoa, and Ochi [106], hereafter referred to as ILS-SP, which combines ILS with a *set partitioning* (SP) approach. This method was successfully tested on a variety of VRPs, including the *capacitated VRP* (CVRP).

ILS-SP, which is thoroughly described in Subramanian, Uchoa, and Ochi [106], is a multi-start algorithm that includes a *Randomized variable neighborhood descent* (RVND) procedure with many classical local search operators. The inter-route neighborhood structures are Shift(λ_1,0), Swap(λ_1,λ_2), $\lambda_1, \lambda_2 \in [1, 2]$, and Cross (2-opt*), resulting in a total of 6 operators disregarding symmetries. The intra-route neighborhood structures are Reinsertion, Or-opt2, Or-opt3, Exchange and 2-opt. Regarding the perturbation procedures, at most two random Swap(1,1) and Shift(1,1) moves are consecutively applied to a local optimal solution. The SP approach tries to find the best combination of routes stored during the search using a *mixed integer programming* (MIP) solver. If the instance contains less than or equal to 150 customers, i.e., the instance is not large, the SP procedure is called at the end of the algorithm. Otherwise, the SP is called periodically during the restarts. One key aspect of the matheuristic is that the ILS procedure is called every time an incumbent solution is found by the MIP solver. This often improves the performance of the solver, as not only improved solutions can be found, but also the SP problem can be solved faster. The matheuristic also has an adaptive mechanism that dynamically controls the size of the pool of routes and makes use of *auxiliary data structures* that are crucial for improving local search performance.

The following subsections briefly describe three ILS-SP based matheuristics which explore problem-specific information at different levels.

3.5.1 First strategy

The first strategy applies the original ILS-SP directly to the VRPB. Note that a VRPB instance can be transformed into a AVRPMB instance by simply setting $c_a = \infty$, $\forall a \notin A_{0L} \cup A_{LB} \cup A_B$. Consequently, infeasible solutions are allowed at any ILS step. Although this may potentially increase the search space, unnecessary operations may be performed. In particular, moves that do not yield improvement are evaluated when the current solution is feasible. For example, the algorithm unnecessarily evaluates the cost of moving a backhaul customer in between two linehauls even when the solution is already feasible.

3.5.2 Second strategy

The second strategy, called ILS_B-SP, modifies ILS-SP to cope with the specific VRPB characteristics during the ILS phase. The construction and perturbation phases only allow feasible solutions to be generated. Furthermore, the search range of each of the 11 neighborhoods is limited to potential feasible moves regarding precedence constraints. It is important to emphasize that this is arguably not so straightforward to code. Finally, the algorithm considers a key additional auxiliary data structure that keeps track of the position of the last linehaul customer of each route.

3.5.3 Third strategy

ILS_B-SP_B modifies ILS_B-SP to cope with the specific VRPB characteristics during the SP phase. Instead of building the model using complete routes, SP_B considers linehaul and backhaul paths separately. Hence, paths originated from different routes can be combined to generate a new route. This allows for exploring further regions of the search space. The SP_B approach was formulated as in Mingozzi et al (1999).

Let $\mathcal{P}$, $\mathcal{L}$ and $\mathcal{B}$ be the set of all paths, linehaul paths and backhaul paths stored by ILS, respectively. Define $\mathcal{B}_0$ as the set of backhaul paths plus one path just with the depot. Let $\mathcal{L}_i$ and $\mathcal{B}_i$ be the set of linehaul paths containing customer $i \in L$ and customer $i \in B$, respectively. Define $\mathcal{L}'_i$ and $\mathcal{B}'_i$ as the set of linehaul paths ending at customer $i \in L$ and backhaul paths starting at customer $i \in B$, respectively. Each path $p \in \mathcal{P}$ has an associated cost c_p. Let y_p be a binary variable that assumes value 1 if $p \in \mathcal{P}$ is chosen, 0 otherwise. Let x_{ij} be binary variable that assumes value 1 if arc $(i, j) \in A_{LB}$ is in the solution, 0 otherwise.

The formulation can be written as follows:

$$\min \sum_{p \in \mathcal{L}} c_p y_p + \sum_{p \in \mathcal{B}} c_p y_p + \sum_{(i,j) \in A_{LB}} c_{ij} x_{ij} \tag{3.32}$$

$$\sum_{p \in \mathcal{L}_i} y_p = 1 \qquad i \in L, \tag{3.33}$$

$$\sum_{p \in \mathcal{B}_j} y_p = 1 \qquad j \in B, \tag{3.34}$$

$$\sum_{p \in \mathcal{L}'_i} y_p = \sum_{j \in B_0} x_{ij} \qquad i \in L, \tag{3.35}$$

$$\sum_{p \in \mathcal{B}'_j} y_p = \sum_{i \in L} x_{ij} \qquad j \in B, \tag{3.36}$$

$$\sum_{p \in \mathcal{L}} y_p = K, \tag{3.37}$$

$$y_p \in \{0,1\} \qquad p \in \mathcal{P}, \tag{3.38}$$

$$x_{ij} \in \{0,1\} \qquad (i,j) \in A_{LB}. \tag{3.39}$$

Objective function (3.32) minimizes the sum of the costs. Constraints (3.33)–(3.34) state that each customer must be exactly in one path. Constraints (3.35)–(3.36) link variables x and y. Constraint (3.37) ensures that there are K routes are in the solution. Finally, constraints (3.38)–(3.39) define the domain of the variables.

3.6 Computational experiments

The experiments were executed on a 2 Deca-core Haswell Intel Xeon E5-2680 v3 server with 2.50 GHz and 128 GB of RAM. Each algorithm was run on a single thread for each instance. To reduce the testing time, multiple runs (64) for different instances were performed simultaneously on the same machine, effectively reducing the amount of RAM allocated to each process. A time limit of 60 hours was imposed for the exact algorithms. The three ILS-SP matheuristics were executed 50 times for each instance.

The BCP algorithms were coded in Julia 1.2 interface for the generic VRP-Solver [92] which makes use of JuMP [43] and LightGraphs packages. The models used in the implementation are described in Appendix H. The solver utilizes the BaPCod C++ library [117] as BCP framework combined with the C++ implementations by Sadykov, Uchoa, and Pessoa [100] which contain: (i) a labeling algorithm for solving the pricing subproblems based on bucket graphs; (ii) path enumeration; (iii) a bucket arc elimination routine; (iv) a routine for separating limited-memory rank-1 cuts; and (v) dual price smoothing stabilization [93]. Moreover, CVRPSEP package [78] is used in the RCC separators and CPLEX 12.8 is used to solve the LP relaxations and the MIPs over the enumerated paths.

The ILS-SP metheuristics were coded in C++, where the same parameters used in Uchoa et al. [115] were adopted. The SP formulations were solved with CPLEX 12.4.

3.6.1 Benchmark instances

We considered four sets of VRPB instances. The first three are classical small and medium size datasets, whereas the fourth one is introduced in this work to test the limits of our methods on instances of larger scale.

- **GJB**. This dataset consists of 68 symmetric instances proposed by Goetschalckx and Jacobs-Blecha [54] including between 25 and 200 customers. The fleet size K is fixed and any feasible solution should have exactly K non-empty routes. We use double precision for the distance matrix.

- **TV**. This group is composed of 33 symmetric instances suggested by Toth and Vigo [113] varying between 21 and 100 customers. The convention regarding the number of vehicles is the same as in the previous dataset. The values of the distance matrix were rounded to the nearest integer.

- **FTV**. This group is composed of 24 asymmetric instances suggested by Toth and Vigo [113] varying between 33 and 70 customers. The convention regarding the number of vehicles is the same as in GJB and TV. The values of the distance matrix were rounded to the nearest integer.

- **X**. This new benchmark dataset contains 300 symmetric instances varying between 100 and 1000 customers. They were generated based on the CVRP instances proposed by Uchoa et al. [115]. For each CVRP instance, we created 3 VRPB ones with 50%, 66% and 80% of linehaul customers, respectively, following the same scheme as Toth and Vigo [113]. For example, we used the CVRP instance X-n101-k25 to generate the VRPB instances X-n101-50-k13, X-n101-66-k17, X-n101-80-k21. It is important to emphasize that the fleet size is not fixed for this dataset. We adopted the nearest integer precision convention for the distance matrix. This newly proposed benchmark is available at `http://www.vrp-rep.org/datasets/download/queiroga-et-al-2019.zip`.

VRPBTW and HFFVRPB instances

The experiments on the VRPBTW and HFFVRPB were conducted with the following benchmarks:

- **GDDS**. This dataset contains 15 instances proposed by Gélinas et al. [52] for the VRPBTW, all of them with 100 customers. All distances are calculated with double precision. The vehicle capacity and the time window of the depot were set to 200 and $[0, 230]$, respectively.
- **T**. This benchmark is composed of 18 instances proposed by Tütüncü [114] and contain between 50 to 100 customers. The double precision convention for the distance matrix was also adopted.

3.6.2 Results for the BCP algorithms

Now, we will present the results obtained by $\text{BCP}_{\mathcal{F}1}$ and $\text{BCP}_{\mathcal{F}2}$ algorithms for the VRPB, VRPBTW and HFFVRPB problems, respectively.

Results for the VRPB

In the tables presented hereafter, UB is the initial upper bound provided to the exact algorithms, $time_{ub}$ denotes the CPU time required by the heuristic(s) to obtain UB; $z(IP)$ indicates the value of the optimal solution or an improved upper bound, LB^{f}_{root} corresponds to the final lower bound (LB) found at the root node; *time* is the total CPU time, $time_{prc}$ is the total pricing time, and *nodes* represents the number of nodes in the tree.

Table 3.2 presents the results obtained by $\text{BCP}_{\mathcal{F}1}$ and $\text{BCP}_{\mathcal{F}2}$ for the GJB instances. The upper bounds are those from the best solutions found by ILS_{B}-SP (hence $time_{ub}$ is the sum of its 50 executions). All instances were solved to optimality by both algorithms. Note that almost all instances were solved to optimality at the root node, including most of the 200-customer ones. Regarding the CPU time, $\text{BCP}_{\mathcal{F}2}$ is clearly faster than $\text{BCP}_{\mathcal{F}1}$, except for very few cases (instances G4 and G5). $\text{BCP}_{\mathcal{F}2}$ can be around 7 times faster as it happened on instance O1. Hence, although the LB^{f}_{root} obtained by $\text{BCP}_{\mathcal{F}2}$ can be occasionally slightly weaker than the one achieved by $\text{BCP}_{\mathcal{F}1}$, it appears that the first has a better overall performance than the latter. Nonetheless, in practice, the bound LB^{f}_{root} obtained by $\mathcal{F}2$ can be better than the one obtained by $\mathcal{F}1$. This is possible because the cut generation may be interrupted early in $\text{BCP}_{\mathcal{F}1}$ due to the high CPU time required to solve the pricing subproblems. This is one of the criteria used by the BCP framework to stop the cut generation and perform branching.

Table 3.2: Results obtained for the GJB instances

Problem data							$z(IP)$	BCP$_{\mathcal{F}1}$			BCP$_{\mathcal{F}2}$		
Instance	$n+m$	n	m	K	UB	$time_{ub}$ (s)		LB^f_{root}	$time$ (s)	$nodes$	LB^f_{root}	$time$ (s)	$nodes$
A1	25	20	5	8	229,885.65	10	229,885.65	229,885.65	< 1	1	229,885.65	< 1	1
A2	25	20	5	5	180,119.21	9	180,119.21	180,119.21	< 1	1	180,119.21	< 1	1
A3	25	20	5	4	163,405.38	10	163,405.38	163,405.38	< 1	1	163,405.38	< 1	1
A4	25	20	5	3	155,796.41	7	155,796.41	155,796.41	< 1	1	155,796.41	< 1	1
B1	30	20	10	7	239,080.16	12	239,080.16	239,080.16	< 1	1	239,080.16	< 1	1
B2	30	20	10	5	198,047.77	11	198,047.77	198,047.77	< 1	1	198,047.77	< 1	1
B3	30	20	10	3	169,372.29	7	169,372.29	169,372.29	< 1	1	169,372.29	< 1	1
C1	40	20	20	7	250,556.77	20	250,556.77	250,556.77	< 1	1	250,556.77	< 1	1
C2	40	20	20	5	215,020.23	18	215,020.23	215,020.23	2	1	215,020.23	< **1**	1
C3	40	20	20	5	199,345.96	19	199,345.96	199,345.96	< 1	1	199,345.96	< 1	1
C4	40	20	20	4	195,366.63	17	195,366.63	195,366.63	< 1	1	195,366.63	< 1	1
D1	38	30	8	12	322,530.13	29	322,530.13	322,530.13	< 1	1	322,530.13	< 1	1
D2	38	30	8	11	316,708.86	24	316,708.86	316,708.86	< 1	1	316,708.86	< 1	1
D3	38	30	8	7	239,478.63	21	239,478.63	239,478.63	< 1	1	239,478.63	< 1	1
D4	38	30	8	5	205,831.94	22	205,831.94	205,831.94	6	1	205,831.94	**2**	1
E1	45	30	15	7	238,879.58	23	238,879.58	238,879.58	< 1	1	238,879.58	< 1	1
E2	45	30	15	4	212,263.11	26	212,263.11	212,263.11	< 1	1	212,263.11	< 1	1
E3	45	30	15	4	206,659.17	32	206,659.17	206,659.17	1	1	206,659.17	< **1**	1
F1	60	30	30	6	263,173.96	55	263,173.96	263,173.96	5	1	263,173.96	**3**	1
F2	60	30	30	7	265,214.16	55	265,214.16	265,214.16	2	1	265,214.16	< **1**	1
F3	60	30	30	5	241,120.78	52	241,120.78	241,120.78	2	1	241,120.78	**1**	1
F4	60	30	30	4	233,861.85	56	233,861.85	233,861.85	3	1	233,861.85	**2**	1
G1	57	45	12	10	306,305.40	73	306,305.40	306,305.40	5	1	306,305.40	5	1
G2	57	45	12	6	245,440.99	54	245,440.99	245,440.99	3	1	245,440.99	3	1
G3	57	45	12	5	229,507.48	49	229,507.48	229,507.48	3	1	229,507.48	**2**	1
G4	57	45	12	6	232,521.25	56	232,521.25	232,521.25	**3**	1	232,521.25	5	1
G5	57	45	12	5	221,730.35	61	221,730.35	221,730.35	**3**	1	221,730.35	4	1
G6	57	45	12	4	213,457.45	65	213,457.45	213,457.45	3	1	213,457.45	**2**	1
H1	68	45	23	6	268,933.06	99	268,933.06	268,933.06	8	1	268,933.06	**7**	1
H2	68	45	23	5	253,365.50	92	253,365.50	253,365.50	5	1	253,365.50	**2**	1
H3	68	45	23	4	247,449.04	94	247,449.04	247,449.04	4	1	247,449.04	**3**	1
H4	68	45	23	5	250,220.77	105	250,220.77	250,220.77	4	1	250,220.77	**3**	1
H5	68	45	23	4	246,121.31	101	246,121.31	246,121.31	5	1	246,121.31	**3**	1
H6	68	45	23	5	249,135.32	107	249,135.32	249,135.32	5	1	249,135.32	**3**	1
I1	90	45	45	10	350,245.28	240	350,245.28	350,245.28	19	1	350,245.28	**5**	1

(*Continues on the next page*)

Problem data							$z(IP)$	$BCP_{\mathcal{F}1}$			$BCP_{\mathcal{F}2}$		
Instance	$n+m$	n	m	K	UB	$time_{ub}$ (s)		LB^f_{root}	$time$ (s)	$nodes$	LB^f_{root}	$time$ (s)	$nodes$
I2	90	45	45	7	309,943.84	181	309,943.84	309,943.84	16	1	309,943.84	**3**	1
I3	90	45	45	5	294,507.38	206	294,507.38	294,507.38	37	1	294,507.38	**12**	1
I4	90	45	45	6	295,988.45	191	295,988.45	295,988.45	22	1	293,840.10	**9**	3
I5	90	45	45	7	301,236.01	192	301,236.01	301,236.01	12	1	301,236.01	**4**	1
J1	94	75	19	10	335,006.68	223	335,006.68	335,006.68	13	1	335,006.68	**12**	1
J2	94	75	19	8	310,417.21	272	310,417.21	310,417.21	48	1	310,417.21	**33**	1
J3	94	75	19	6	279,219.21	251	279,219.21	279,219.21	19	1	279,219.21	**12**	1
J4	94	75	19	7	296,533.16	299	296,533.16	294,480.85	367	5	294,168.05	**309**	7
K1	113	75	38	10	394,071.17	705	394,071.17	394,071.17	52	1	394,071.17	**23**	1
K2	113	75	38	8	362,130.00	407	362,130.00	362,130.00	36	1	362,130.00	**14**	1
K3	113	75	38	9	365,694.08	420	365,694.08	365,694.08	26	1	365,694.08	**12**	1
K4	113	75	38	7	348,949.39	402	348,949.39	348,949.39	67	1	348,949.39	**29**	1
L1	150	75	75	10	417,896.72	988	417,896.71	417,896.71	82	1	417,896.71	**44**	1
L2	150	75	75	8	401,228.80	1,089	401,228.80	401,228.80	110	1	401,228.80	**57**	1
L3	150	75	75	9	402,677.72	833	402,677.72	402,677.72	76	1	402,677.72	**35**	1
L4	150	75	75	7	384,636.33	854	384,636.33	384,636.33	67	1	384,636.33	**28**	1
L5	150	75	75	8	387,564.55	882	387,564.55	387,564.55	55	1	387,564.55	**23**	1
M1	125	100	25	11	398,593.19	1,627	398,593.19	398,593.19	95	1	398,593.19	**56**	1
M2	125	100	25	10	396,916.97	4,977	396,916.97	396,916.97	112	1	395,706.60	**85**	3
M3	125	100	25	9	375,695.42	1,558	375,695.42	373,010.93	6,210	41	372,016.21	**4,139**	39
M4	125	100	25	7	348,140.16	743	348,140.16	348,140.16	181	1	347,010.67	**160**	3
N1	150	100	50	11	408,100.62	1,323	408,100.62	408,100.62	112	1	406,628.97	**56**	3
N2	150	100	50	10	408,065.44	1,538	408,065.44	408,065.44	124	1	406,269.57	**77**	3
N3	150	100	50	9	394,337.86	1,045	394,337.86	394,337.86	169	1	394,337.86	**46**	1
N4	150	100	50	10	394,788.36	1,177	394,788.36	394,788.36	193	1	394,788.36	**50**	1
N5	150	100	50	7	373,476.30	1,053	373,476.30	373,476.30	247	1	373,476.30	**80**	1
N6	150	100	50	8	373,758.65	1,138	373,758.65	373,758.65	189	1	373,758.65	**65**	1
O1	200	100	100	10	478,126.75	2,035	478,126.75	475,781.67	14,629	29	476,500.02	**1,823**	11
O2	200	100	100	11	477,256.15	1,874	477,256.15	477,256.15	285	1	477,256.15	**77**	1
O3	200	100	100	9	457,294.48	2,046	457,294.48	457,294.48	207	1	457,294.48	**80**	1
O4	200	100	100	10	458,874.87	1,896	458,874.87	458,874.87	130	1	458,874.87	**39**	1
O5	200	100	100	7	436,974.20	2,041	436,974.20	436,974.20	524	1	436,974.20	**168**	1
O6	200	100	100	8	438,004.69	2,006	438,004.69	438,004.69	269	1	438,004.69	**108**	1
Mean									365.9			105.3	
Geometric mean									14.4			8.5	

Table 3.3 reports the results obtained by $BCP_{\mathcal{F}1}$ and $BCP_{\mathcal{F}2}$ for the TV instances. Here, we also use the upper bounds produced by ILS_B-SP. Once again, all instances were solved to optimality by both algorithms, where 9 of them were proven optimal for the

first time. Except for instance E-n101-B-66, all other cases were solved at the root node. Note that the values of LB^f_{root} are integer in this table because $c_a \in \mathbb{Z}^+$, $\forall a \in A$, thus enabling the lower bounds to be rounded up, as the objective value of all feasible solutions are integer.

Table 3.3: Results obtained for the TV instances

Problem data							$z(IP)$	BCP$_{\mathcal{F}1}$			BCP$_{\mathcal{F}2}$		
Instance	$n+m$	n	m	K	UB	$time_{ub}$ (s)		LB^f_{root}	$time$ (s)	$nodes$	LB^f_{root}	$time$ (s)	$nodes$
E-n22-50	21	10	11	3	371	4	371	371	2	1	371	2	1
E-n22-66	21	14	7	3	366	4	366	366	2	1	366	2	1
E-n22-80	21	17	4	3	375	4	375	375	**2**	1	375	3	1
E-n23-50	22	11	11	2	682	4	682	682	3	1	682	3	1
E-n23-66	22	15	7	2	649	4	649	649	3	1	649	3	1
E-n23-80	22	18	4	2	623	5	623	623	3	1	623	3	1
E-n30-50	29	14	15	2	501	7	501	501	4	1	501	**3**	1
E-n30-66	29	19	10	3	537	9	537	537	3	1	537	3	1
E-n30-80	29	23	6	3	514	9	514	514	**3**	1	514	5	1
E-n33-50	32	16	16	3	738	10	738	738	3	1	738	3	1
E-n33-66	32	21	11	3	750	11	750	750	3	1	750	3	1
E-n33-80	32	26	6	3	736	11	736	736	3	1	736	3	1
E-n51-50	50	25	25	3	559	34	559	559	4	1	559	**3**	1
E-n51-66	50	33	17	4	548	37	548	548	4	1	548	**3**	1
E-n51-80	50	40	10	4	565	52	565	565	**4**	1	565	6	1
E-n76-A-50	75	38	37	6	739	99	739	739	8	1	739	**6**	1
E-n76-A-66	75	50	25	7	768	100	768	768	6	1	768	**5**	1
E-n76-A-80	75	60	15	8	781	140	781*	781	5	1	781	**4**	1
E-n76-B-50	75	38	37	8	801	94	801	801	4	1	801	4	1
E-n76-B-66	75	50	25	10	873	119	873	873	**6**	1	873	7	1
E-n76-B-80	75	60	15	12	919	126	919	919	4	1	919	4	1
E-n76-C-50	75	38	37	5	713	106	713	713	13	1	713	**8**	1
E-n76-C-66	75	50	25	6	734	101	734	734	11	1	734	**9**	1
E-n76-C-80	75	60	15	7	733	151	733*	733	**23**	1	733	26	1
E-n76-D-50	75	38	37	4	690	103	690	690	6	1	690	**4**	1
E-n76-D-66	75	50	25	5	715	107	715*	715	26	1	715	**15**	1

(*Continues on the next page*)

Problem data							$z(IP)$	$BCP_{\mathcal{F}1}$			$BCP_{\mathcal{F}2}$		
Instance	$n+m$	n	m	K	UB	$time_{ub}$ (s)		LB^f_{root}	$time$ (s)	$nodes$	LB^f_{root}	$time$ (s)	$nodes$
E-n76-D-80	75	60	15	6	694	127	694*	694	14	1	694	**10**	1
E-n101-A-50	100	50	50	4	831	242	831*	831	39	1	831	**15**	1
E-n101-A-66	100	66	34	6	846	283	846	846	14	1	846	**9**	1
E-n101-A-80	100	80	20	6	856	837	856*	856	114	1	856	**72**	1
E-n101-B-50	100	50	50	7	923	733	923*	923	28	1	923	**22**	1
E-n101-B-66	100	66	34	9	982	3,012	982*	977	1,020	6	974	**243**	7
E-n101-B-80	100	80	20	11	1,008	870	1,008*	1,008	**44**	1	1,008	45	1
Average									43.3			16.8	
Geometric mean									7.7			6.5	

New proven optimal solutions are marked with an asterisk.

Table 3.4 reports the results obtained by $BCP_{\mathcal{F}1}$ and $BCP_{\mathcal{F}2}$ for the FTV asymmetric instances. Again, we use the upper bounds produced by ILS_B-SP. All instances were solved at the root node by both algorithms, where 3 of them were proven optimal for the first time.

Table 3.4: Results obtained for the FTV instances

Problem data							$z(IP)$	$BCP_{\mathcal{F}1}$			$BCP_{\mathcal{F}2}$		
Instance	$n+m$	n	m	K	UB	$time_{ub}$ (s)		LB^f_{root}	$time$ (s)	$nodes$	LB^f_{root}	$time$ (s)	$nodes$
FTV33_50	33	17	16	2	1,841	< 1	1,841	1,841	2	1	1,841	2	1
FTV33_66	33	22	11	2	1,899	< 1	1,899	1,899	2	1	1,899	< **1**	1
FTV33_80	33	27	6	2	1,704	< 1	1,704	1,704	< 1	1	1,704	< 1	1
FTV35_50	35	18	17	2	2,077	< 1	2,077	2,077	2	1	2,077	< **1**	1
FTV35_66	35	24	11	2	2,150	< 1	2,150	2,150	3	1	2,150	< **1**	1
FTV35_80	35	28	7	2	1,996	< 1	1,996	1,996	2	1	1,996	< **1**	1
FTV38_50	38	19	19	2	2,162	< 1	2,162	2,162	< 1	1	2,162	< 1	1
FTV38_66	38	26	12	2	2,132	< 1	2,132	2,132	7	1	2,132	**1**	1
FTV38_80	38	31	7	3	1,982	< 1	1,982	1,982	1	1	1,982	1	1
FTV44_50	44	22	22	2	2,348	< 1	2,348	2,348	44	1	2,348	**4**	1
FTV44_66	44	30	14	2	2,225	< 1	2,225	2,225	30	1	2,225	**10**	1
FTV44_80	44	36	8	3	2,184	< 1	2,184	2,184	6	1	2,184	**5**	1
FTV47_50	47	24	23	2	2,343	1	2,343	2,343	9	1	2,343	**3**	1
FTV47_66	47	32	15	2	2,427	1	2,427	2,427	3	1	2,427	**2**	1
FTV47_80	47	38	9	2	2,312	< 1	2,312	2,312	3	1	2,312	**2**	1
FTV55_50	55	28	27	2	2,425	2	2,425	2,425	40	1	2,425	**7**	1
FTV55_66	55	37	18	2	2,246	2	2,246	2,246	54	1	2,246	**12**	1

(*Continues on the next page*)

Problem data							$z(IP)$	BCP$_{\mathcal{F}1}$			BCP$_{\mathcal{F}2}$		
Instance	$n+m$	n	m	K	UB	$time_{ub}$ (s)		LB^f_{root}	$time$ (s)	$nodes$	LB^f_{root}	$time$ (s)	$nodes$
FTV55_80	55	44	11	2	2,264	2	2,264	2,264	13	1	2,264	**5**	1
FTV64_50	64	32	32	2	2,728	3	2,728	2,728	74	1	2,728	**13**	1
FTV64_66	64	43	21	2	2,673	3	2,673	2,673	43	1	2,673	**15**	1
FTV64_80	64	52	12	3	2,659	2	2,659*	2,659	32	1	2,659	**18**	1
FTV70_50	70	35	35	2	2,934	4	2,934*	2,934	145	1	2,934	**20**	1
FTV70_66	70	47	23	2	2,808	5	2,808	2,808	44	1	2,808	**10**	1
FTV70_80	70	56	14	2	2,684	3	2,684*[a]	2,684	25	1	2,684	**15**	1
Average									24.3			6.2	
Geometric mean									8.9			3.4	

New proven optimal solutions are marked with an asterisk.

[a]This value is different from the value 2,688 reported by Toth and Vigo [113]. We found that the heuristic LKH-3 [59] also reports a cost of 2,684.

Table 3.5 provides a comparison between BCP$_{\mathcal{F}1}$ and BCP$_{\mathcal{F}2}$ for the first 45 instances of the X set. They were solved to optimality by both methods. The upper bounds for these instances were the best ones provided by Vidal [120] by running the algorithm proposed in Vidal et al. [122] and the ILS$_{\text{B}}$-SP ($time_{ub}$ is the sum of the time for both methods). Note that instances X-n125-80-k23 and X-n162-66-k8 are particularly difficult and required more than 10,000 seconds to be solved, regardless of the method. Overall, BCP$_{\mathcal{F}2}$ visibly had a superior runtime performance than BCP$_{\mathcal{F}1}$, more specifically, the former was, on average, approximately 4 times faster than the latter.

Table 3.5: Comparison between the two BCP algorithms for the X instances. Only the first 45 instances of X were considered.

Instance	UB	$time_{ub}$ (s)	$z(IP)$	BCP$_{\mathcal{F}1}$				BCP$_{\mathcal{F}2}$			
				LB^f_{root}	$time$ (s)	$time_{prc}$ (s)	$nodes$	LB^f_{root}	$time$ (s)	$time_{prc}$ (s)	$nodes$
X-n101-50-k13	19,033	6,239	19,033	18,944	246	19	11	18,926	**73**	3	5
X-n101-66-k17	20,490	4,801	20,490	20,367	465	39	23	20,357	**162**	4	5
X-n101-80-k21	23,305	4,296	23,305	23,305	63	7	1	23,305	**33**	2	1
X-n106-50-k7	15,413	7,169	15,413	15,413	81	27	1	15,413	**20**	4	1
X-n106-66-k9	18,984	14,268	18,984	18,984	146	37	1	18,984	**40**	8	1
X-n106-80-k11	22,131	14,599	22,131	22,103	1,242	239	11	22,099	**397**	28	7
X-n110-50-k7	13,103	5,147	13,103	13,103	22	7	1	13,103	**10**	2	1
X-n110-66-k9	13,598	5,527	13,598	13,598	23	11	1	13,598	**9**	3	1
X-n110-80-k11	14,302	7,094	14,302	14,226	414	42	5	14,215	**281**	21	7
X-n115-50-k8	13,927	5,234	13,927	13,927	35	16	1	13,927	**22**	7	1

(Continues on the next page)

Instance	UB	$time_{ub}$ (s)	$z(IP)$	BCP$_{\mathcal{F}1}$ LB^f_{root}	$time$ (s)	$time_{prc}$ (s)	$nodes$	BCP$_{\mathcal{F}2}$ LB^f_{root}	$time$ (s)	$time_{prc}$ (s)	$nodes$
X-n115-66-k8	14,032	5,353	14,032	14,032	48	20	1	14,032	**25**	9	1
X-n115-80-k9	13,536	5,776	13,536	13,536	50	19	1	13,536	**31**	13	1
X-n120-50-k3	12,416	9,110	12,416	12,416	243	81	1	12,416	**73**	17	1
X-n120-66-k4	13,145	10,729	13,145	13,100	1,377	545	3	13,145	**325**	137	1
X-n120-80-k5	13,528	9,082	13,528	13,476	3,052	1,707	15	13,465	**2,737**	1,575	17
X-n125-50-k16	32,224	15,163	32,224	32,079	3,688	310	79	32,065	**915**	56	39
X-n125-66-k19	36,400	18,016	36,400	36,351	1,098	362	9	36,349	**271**	34	3
X-n125-80-k23	43,960	18,973	43,960	43,825	**10,323**	2,341	129	43,823	11,877	1,306	245
X-n129-50-k10	19,468	18,579	19,468	19,429	1,358	143	9	19,409	**335**	29	7
X-n129-66-k12	22,606	14,108	22,606	22,556	946	141	11	22,554	**226**	19	7
X-n129-80-k14	24,575	18,091	24,575	24,562	308	51	3	24,553	**108**	22	3
X-n134-50-k7	8,369	14,267	8,369	8,271	15,713	10,160	105	8,316	**868**	390	5
X-n134-66-k9	8,974	22,858	8,974	8,913	5,796	3,749	65	8,891	**621**	353	15
X-n134-80-k11	9,699	14,118	9,699	9,637	4,606	2,478	65	9,637	**1,222**	714	27
X-n139-50-k5	13,281	9,149	13,281	13,229	1,639	656	5	13,237	**290**	41	3
X-n139-66-k7	13,512	8,461	13,512	13,512	153	63	1	13,512	**51**	15	1
X-n139-80-k8	13,662	8,118	13,662	13,662	65	29	1	13,662	**40**	19	1
X-n143-50-k4	14,539	13,448	14,539	14,539	1,592	941	1	14,539	**214**	76	1
X-n143-66-k4	14,310	11,172	14,310	14,310	233	128	1	14,310	**82**	41	1
X-n143-80-k5	14,447	11,338	14,447	14,396	3,148	2,244	5	14,397	**2,822**	1,714	13
X-n148-50-k25	28,210	18,782	28,210	28,174	112	13	3	28,210	**30**	4	1
X-n148-66-k29	30,482	17,414	30,482	30,404	421	40	13	30,392	**112**	6	3
X-n148-80-k36	35,430	18,421	35,430	35,334	394	22	13	35,333	**318**	5	3
X-n153-50-k19	20,536	14,665	20,536	20,536	53	32	1	20,536	**23**	11	1
X-n153-66-k20	20,613	14,645	20,613	20,613	68	34	1	20,613	**31**	12	1
X-n153-80-k21	20,819	11,360	20,819	20,813	77	40	3	20,811	**57**	24	3
X-n157-50-k7	11,727	15,033	11,727	11,727	333	150	1	11,727	**37**	12	1
X-n157-66-k9	13,651	10,979	13,651	13,651	123	49	1	13,651	**43**	14	1
X-n157-80-k11	15,264	39,145	15,264	15,257	1,186	252	3	15,246	**733**	164	7
X-n162-50-k6	12,812	10,780	12,812	12,785	1,310	762	3	12,812	**157**	55	1
X-n162-66-k8	13,450	10,916	**13,417**	13,290	137,067	80,154	607	13,301	**19,365**	8,164	85
X-n162-80-k9	13,854	11,032	13,854	13,820	2,294	1,016	3	13,854	**812**	329	1
X-n167-50-k5	16,489	22,046	16,489	16,489	1,989	1,058	1	16,489	**336**	91	1
X-n167-66-k7	17,827	20,356	17,827	17,736	11,480	6,049	21	17,717	**3,411**	1,813	17
X-n167-80-k8	19,415	24,099	19,415	19,375	1,554	740	3	19,383	**770**	348	3
Average					4,814.1	2,600.5	27.6		**1,120.3**	393.6	12.2
Geometric mean					540.2	163.5	4.5		**177.8**	38.1	3.0

Because of the overall superior performance of BCP$_{\mathcal{F}2}$, we decided to run only this algorithm for the remaining X instances. Table 3.6 presents a summary of the results obtained by this method considering all instances of set X, while the table provided in the Appendix F shows the detailed results (except for those already reported in Table 3.5). On average, the results suggest the average gap does not seem to substantially vary according to the percentage of linehaul customers, but the average CPU time increases with the percentage of linehaul customers (instances with 80% of linehaul customers lasted four hours more than those with 50%). On the other hand, the more the instance is balanced, the higher the number of proven optimal solutions. Finally, one can observe that 14 best-known solutions were improved, considering the cases where their optimality was proven or not.

Table 3.6: Summary of the results obtained by BCP$_{\mathcal{F}2}$ for the 300 instances of group X, considering the percentage of linehaul customers.

	50%	66%	80%	All
Average gap (%)	0.53	0.46	0.50	0.50
Average time (min)	2,110.3	2,244.0	2,359.7	2,238.0
#Optima	46	40	37	123
#BKS improvements	6	3	5	14

Figure 3.5 shows the gaps for each instance, according to the percentage of linehaul customers. It is possible to verify that all instances involving up to 237 customers were solved to optimality for 50%, whereas this number decreases to 186 customers for 66% and 80%. Furthermore, one can observe that the average gaps were generally below 2.5%, even for the larger instances, but in the vast majority of the cases they were below 2.0%, thus ratifying the high quality of the bounds reported.

Figure 3.6 illustrates the behavior of the average gaps as the estimated size of the routes increases. The instances are classified as in Uchoa et al. [115] for the CVRP, where the desired values of n/K_{min} (n is the number of customers and K_{min} is the minimum number of routes) for the generated instances were partitioned into quintiles, classifying the group of instances as "very small", "small", "medium", "long" and "very long". Hence, if three instances of VRPB are derived from a "very small" instance of CVRP, then they are also classified as "very small". The box plots suggest that the smaller the size of the routes, the smaller the gaps and the higher the robustness obtained. In addition, note that at least 25% of the instances of each group were solved to optimality.

Furthermore, in order to assess the impact of the initial UBs, we report in Table 3.7 the amount of optimal solutions found by each BCP algorithm using such bounds or not. While the BCP algorithms could solve all classical instances even without the initial

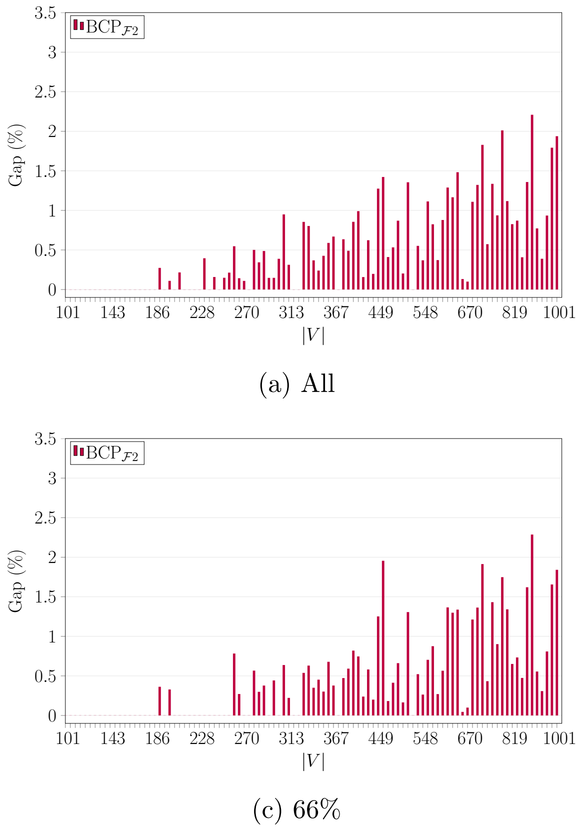


Figure 3.5: Average gaps for the X instances. In Figure 3.5a, the value reported is given for each value of $|V|$ as the average gap of the three related instances. The other figures show the gap of the instances associated with the corresponding percentage of linehauls.

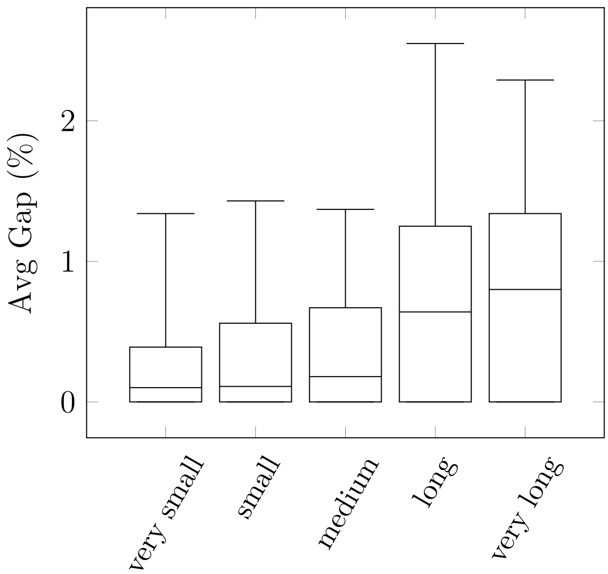


Figure 3.6: Average gaps with respect to the size of the route.

UBs, the latter play an important role when it comes to proving the optimality of some instances of set X. Detailed results are provided in Appendix G.

Table 3.7: Impact of the initial upper bounds on the number of optimal solutions found by the BCP algorithms

Benchmark	#Opt with UBs		#Opt without UBs	
	$\text{BCP}_{\mathcal{F}1}$	$\text{BCP}_{\mathcal{F}2}$	$\text{BCP}_{\mathcal{F}1}$	$\text{BCP}_{\mathcal{F}2}$
GJB	68	68	68	68
TV	33	33	33	33
X[a]	45	45	42	41
X[b]	–	78	–	49

[a]First 45 instances
[b]Last 255 instances

Results for the HFFVRPB and VRPBTW

Table 3.8 shows the results obtained for the HFFVRPB instances. All optimal solutions were found by the proposed algorithm. The instances with up to 75 customers were solved to optimality at the root node in a matter of seconds, whereas the 100-customer instances were solved in at most 1,983 seconds. The proposed algorithm was capable of improving the BKS of 6 instances, including all the 100-customer ones. Moreover, we also confirmed the observation made by Penna et al. [89] and proved that instances HFFVRPB3, HFFVRPB6, HFFVRPB8, HFFVRPB12 and HFFVRPB14 are indeed infeasible. For some instances, there is no value for $time_{ub}$ because their upper bounds were obtained from Tütüncü [114], whose CPU times were not reported.

The results obtained for the VRPBTW instances can be found in Table 3.9. In this table, LB^{f}_{root} and *nodes* stand for the final LB at the root node and the number of nodes for the iteration which considers the optimal number of vehicles (as described in Section 3.4.3, the BCP algorithm is executed iteratively for different fleet sizes). In contrast, *time* aggregates the CPU time for all iterations. The optimality of all instances was proven, where new improved solutions were found for instances BHR104A, BHR104B and BHR104C. Almost all instances were solved to optimality at the root node, most of them in a matter of seconds. We can also highlight that all iterations for non-optimal fleet sizes were concluded at the root node. Instance BHR104A appears to be the most challenging one, where the algorithm required more than 1,300 seconds to solve it.

Table 3.8: Results for the T instances

Problem data						BCP			
Instance	$n+m$	n	m	UB	$time_{ub}$ (s)	LB^t_{root}	$z(IP)$	$time$ (s)	$nodes$
HFFVRPB1	50	25	25	874.60	< 1	874.60	874.60	4	1
HFFVRPB2	50	34	16	911.20	< 1	911.20	911.20	5	1
HFFVRPB3	50	40	10	998.22	–	–	–	–	–
HFFVRPB4	50	25	25	1,050.60	< 1	1,050.60	1,050.60	23	1
HFFVRPB5	50	34	16	1,051.30	< 1	1,051.30	1,051.30	5	1
HFFVRPB6	50	40	10	1,183.36	–	–	–	–	–
HFFVRPB7	75	37	38	1,073.90	2	1,070.00	**1070.00**	25	1
HFFVRPB8	75	50	25	1,182.66	–	–	–	–	
HFFVRPB9	75	60	15	1,003.20	2	1,003.20	1,003.20	8	1
HFFVRPB10	75	37	38	1,553.00	2	1,553.00	1,553.00	7	1
HFFVRPB11	75	50	25	1,659.80	2	1,659.80	1,659.80	27	1
HFFVRPB12	75	60	15	1,917.54	–	–	–	–	–
HFFVRPB13	100	50	50	1,181.70	6	1,167.43	**1,180.30**	1,983	5
HFFVRPB14	100	67	33	1,109.02	–	–	–	–	–
HFFVRPB15	100	80	20	1,114.90	5	1,097.36	**1,105.10**	1,349	8
HFFVRPB16	100	50	50	1,314.50	5	1,305.98	**1,312.80**	879	2
HFFVRPB17	100	67	33	1,585.30	–	1,211.70	**1,211.70**	252	1
HFFVRPB18	100	80	20	1,615.08	–	1,279.36	**1,282.00**	448	2

Table 3.9: Results for GDDS instances

Problem data				BCP			
Instance	%BH	UB	$time_{ub}$ (s)	LB^f_{root}	$z(IP)$	$time$ (s)	$nodes$
BHR101A	10	22/1,818.86	77	1,818.86	22/1,818.86	4	1
BHR101B	30	23/1,959.52	103	1,959.52	23/1,959.52	3	1
BHR101C	50	24/1,939.10	76	1,939.10	24/1,939.10	3	1
BHR102A	10	19/1,653.18	79	1,653.18	19/1,653.18	7	1
BHR102B	30	22/1,750.70	70	1,750.70	22/1,750.70	4	1
BHR102C	50	22/1,775.76	71	1,775.76	22/1,775.76	4	1
BHR103A	10	15/1,385.38	84	1,385.38	15/1,385.38	6	1
BHR103B	30	15/1,390.32	86	1,390.32	15/1,390.32	8	1
BHR103C	50	17/1,456.48	79	1,456.48	17/1,456.48	5	1
BHR104A	10	10/1,203.44	110	1,182.87	**10/1,202.53**	1,330	11
BHR104B	30	11/1,154.84	104	1,258.48	**10/1,258.48**	55	1
BHR104C	50	11/1,191.38	143	1,188.78	**11/1,188.78**	23	1
BHR105A	10	15/1,560.15	123	1,547.48	15/1,560.15	83	3
BHR105B	30	16/1,583.30	104	1,583.30	16/1,583.30	16	1
BHR105C	50	16/1,709.66	128	1,709.66	16/1,709.66	52	1

3.6.3 Results for the ILS-SP matheuristics

In the tables presented hereafter, **Gap$_{\mathbf{avg}}$** is the gap between the average solution and the best known solution (BKS) and **CPU (s)** is the average CPU time in seconds. In what follows, we report aggregate results for each benchmark set. Detailed results are provided in Appendix J.

Results for the GJB and TV instances

Table 3.10 shows the results obtained on the GJB instances. All strategies found the BKSs and improved the result of one instance. Their performance in terms of solution quality was, on average, similar but ILS$_\mathrm{B}$-SP$_\mathrm{B}$ had a subtle advantage. Concerning the CPU time, ILS$_\mathrm{B}$-SP was, on average, slightly faster.

Table 3.10: Summary of the results found for the GJB instances

	ILS-SP	ILS$_\mathrm{B}$-SP	ILS$_\mathrm{B}$-SP$_\mathrm{B}$
Gap$_\mathrm{avg}$ (%)	0.015	0.015	**0.005**
CPU (s)	15.25	**11.24**	12.07
#Ties	67	67	67
#Improvements	1	1	1

Table 3.11 presents a comparison between our three strategies and best existing heuristics, namely, MACS [51], RPA [127], ILS-400 [32], ILS-1000 [32], *unified hybrid genetic search* (UHGS) [122], BRMF [14], and *slack induction by string removals* (SISRs) [27]. It can be observed that the proposed algorithms achieved similar performance than SISRs and outperformed those other ones from the literature both regarding solution quality as well as CPU time, which in turn were scaled (when possible) to the machine used in Cuervo et al. [32].

Table 3.12 provides the detailed results obtained on the GJB 200-customer instances. In this case, it can be clearly observed that ILS$_\mathrm{B}$-SP$_\mathrm{B}$ had the best performance in terms of solution quality, whereas ILS$_\mathrm{B}$-SP was visibly the fastest in terms of CPU time.

Figure 3.7 illustrates how the CPU time increase with the number of customers on the GJB instances. It is possible to verify that they are similar up to 113 customers and then the strategies seem to have a distinct performance from that point onwards. The non-monotonic behavior of the runtime is because the performance does not only depend on the number of customers, but also on the average number of customers per route.

Table 3.13 shows the aggregate results obtained on the TV instances, whereas Table

Table 3.11: Comparison with the literature: GJB instances up to 150 customers. The columns “Avg. best sol. cost" and “Avg. sol. cost" allow to compare with Cuervo et al. [32]

Method	#Best	Avg. best sol. cost	Avg. sol. cost	Gap_{avg} (%)	CPU (s)
MACS	46/62	290,655.29	290,920.90	0.093	37.35
RPA	**62/62**	290,576.06	291,927.72	0.354	35.08
ILS-400	58/62	290,593.84	291,332.41	–	14.31
ILS-1000	**62/62**	**290,576.22**	291,170.16	–	22.89
UHGS[a]	**61/61**	–	–	0.009	51.20
BRMF[b]	52/62	≈290,741.53	≈292,485.47	0.526	46.00[c]
SISRs[b]	**62/62**	≈290,577.10	≈290,586.29	0.003	**2.56**
ILS-SP	**62/62**	**290,576.22**	290,591.23	0.004	10.47
ILS_B-SP	**62/62**	**290,576.22**	290,588.41	0.003	7.71
ILS_B-SP_B	**62/62**	**290,576.22**	**290,585.62**	**0.002**	8.28

[a] Instance G1 was disregarded

[b] Approximate value because the authors reported the costs divided by 10^3

[c] Intel Core i7 CPU 2.79 GHz

Table 3.12: Detailed results for the 200-customer GJB instances

Instance	BKS	ILS-SP			ILS_B-SP			ILS_B-SP_B		
		Best	Gap_{avg} (%)	CPU (s)	Best	Gap_{avg} (%)	CPU (s)	Best	Gap_{avg} (%)	CPU (s)
O1	478,347.72	**478,126.75**	0.13	58.57	**478,126.75**	0.11	40.71	**478,126.75**	**0.01**	76.52
O2	477,256.15	477,256.15	**0.00**	57.02	477,256.15	**0.00**	37.49	477,256.15	**0.00**	42.52
O3	457,294.48	457,294.48	0.28	61.26	457,294.48	0.29	40.91	457,294.48	**0.10**	43.98
O4	458,874.87	458,874.87	0.24	59.31	458,874.87	0.34	37.92	458,874.87	**0.07**	42.32
O5	436,974.20	436,974.20	0.05	69.48	436,974.20	0.04	40.82	436,974.20	**0.01**	45.39
O6	438,004.69	438,004.69	0.11	68.21	438,004.69	0.07	40.13	438,004.69	**0.02**	44.62
Average			0.13	62.31		0.14	**39.66**		**0.03**	49.23

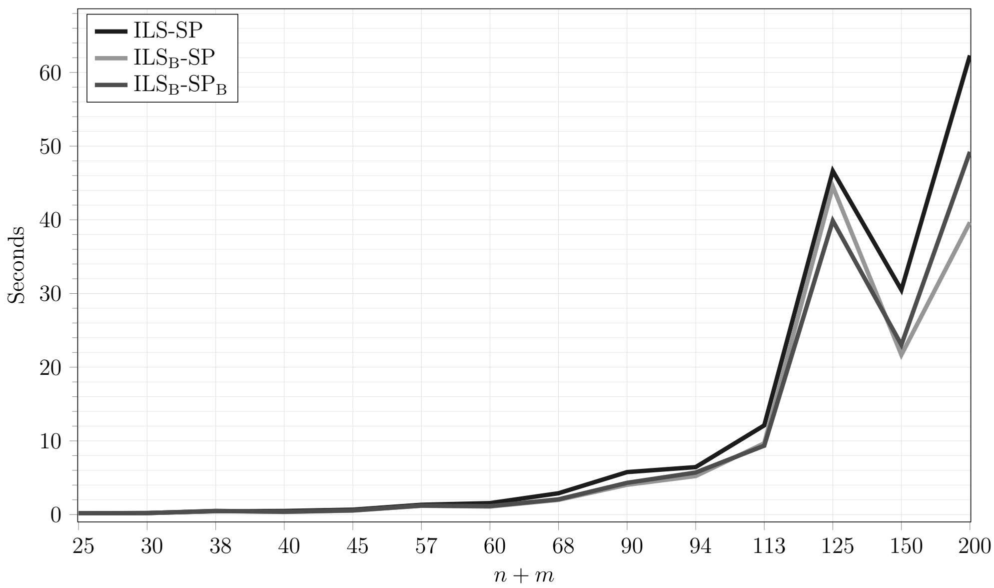


Figure 3.7: Average CPU time (s) for the GJB instances

3.14 presents a comparison between our three strategies and the best known heuristics and Table 3.15 provides the detailed results achieved on the TV 100-customer instances. The overall performance is quite similar to those observed for the GJB instances. Moreover, Figure 3.8 shows that the CPU time only starts to be distinct for the instances involving more than 75 customers.

Table 3.13: Summary of the results found for the TV instances

	ILS-SP	ILS_B-SP	ILS_B-SP_B
Gap_{avg} (%)	0.07	0.04	**0.03**
CPU (s)	4.86	**4.58**	5.94
#Ties	33	33	33
#Improvements	0	0	0

Table 3.14: Comparison with the literature: TV instances

Method	#Best	Avg. best sol. cost	Avg. sol. cost	Gap_{avg} (%)	CPU (s)
MACS	27/33	701.49	702.35	0.193	14.17
ILS-400	31/33	700.72	704.42	–	3.83
ILS-1000	32/33	700.64	703.52	–	7.35
BRMF	**33/33**	**700.61**	704.76	0.49	20[c]
ILS-SP	**33/33**	**700.61**	701.32	0.069	3.34
ILS_B-SP	**33/33**	**700.61**	700.95	0.038	**3.14**
ILS_B-SP_B	**33/33**	**700.61**	**700.91**	**0.034**	4.08

[c]Intel Core i7 CPU 2.79 GHz

Table 3.15: Detailed results for the 100-customer TV instances

		ILS-SP			ILS_B-SP			ILS_B-SP_B		
Instance	BKS	Best	Gap_{avg} (%)	CPU (s)	Best	Gap_{avg} (%)	CPU (s)	Best	Gap_{avg} (%)	CPU (s)
E-n101-A-50-k4	831	831	0.11	7.71	831	0.00	4.84	831	**0.00**	4.95
E-n101-A-66-k6	846	846	**0.00**	7.78	846	**0.00**	5.66	846	**0.00**	5.58
E-n101-A-80-k6	856	856	0.78	17.87	856	0.49	16.74	856	**0.47**	18.28
E-n101-B-50-k7	923	923	0.19	16.26	923	0.14	14.66	923	**0.02**	21.27
E-n101-B-66-k9	982	982	0.86	57.35	982	0.36	60.24	982	**0.34**	94.11
E-n101-B-80-k11	1,008	1,008	0.01	16.66	1,008	**0.00**	17.40	1,008	**0.00**	19.82
Average			0.32	20.60		0.16	**19.92**		**0.14**	27.34

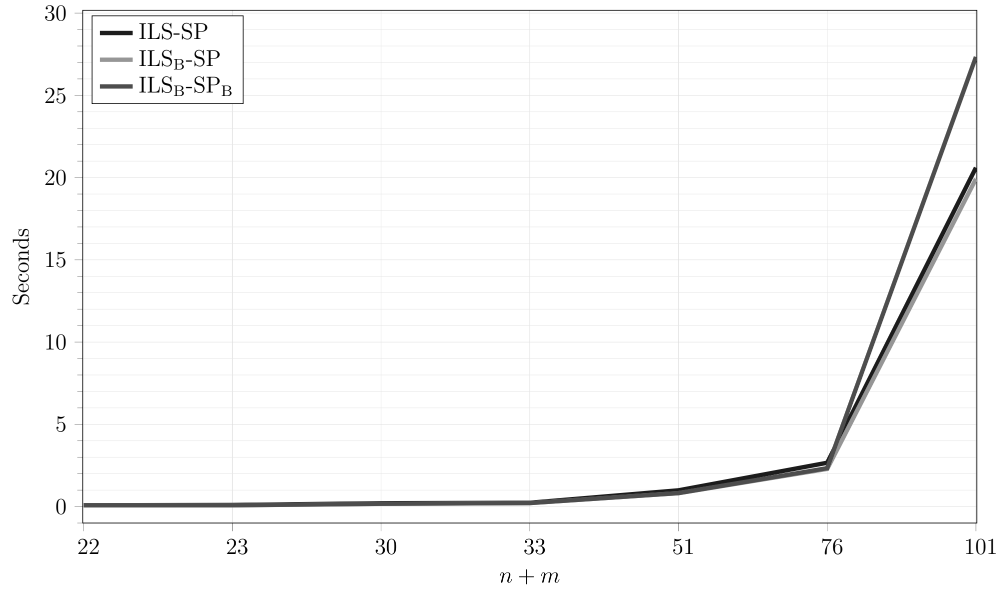


Figure 3.8: Average CPU time (s) for the TV instances

Results for the X instances

Table 3.16 contains the results achieved on the X instances. In this table, $\mathbf{Gap_{best}}$ denotes the average gap between the best solution and the BKS. ILS_B-SP_B obtained the best performance in terms of solution quality, especially when analyzing the behavior of average gaps depicted in Figure 3.9, as well as the number of ties and improved solutions. The Figure 3.9 has a cyclic behavior because the X benchmark is cyclical w.r.t. the percentage of linehaul customers (from 50% to 80%), which is an attribute correlated with the difficulty of the problem. Although not reported in the table, we highlight that ILS_B-SP_B obtained the best Gap_avg in 173 instances, against 111 and 35 of ILS_B-SP and ILS-SP, respectively. On the other hand, ILS_B-SP was, on average, clearly the fastest strategy, as can also be seen in Figure 3.10. Nonetheless, it is interesting to observe in Table 3.17 that the differences between the strategies tend to decrease as the number of linehaul customers increase.

Table 3.16: Summary of the results found for the X instances

	ILS-SP	ILS_B-SP	ILS_B-SP_B
Gap_best (%)	0.29	0.26	**0.25**
Gap_avg (%)	0.67	0.59	**0.56**
CPU (s)	2,600	**2,005**	2,660
#Ties	71	75	**83**
#Improvements	31	27	**33**

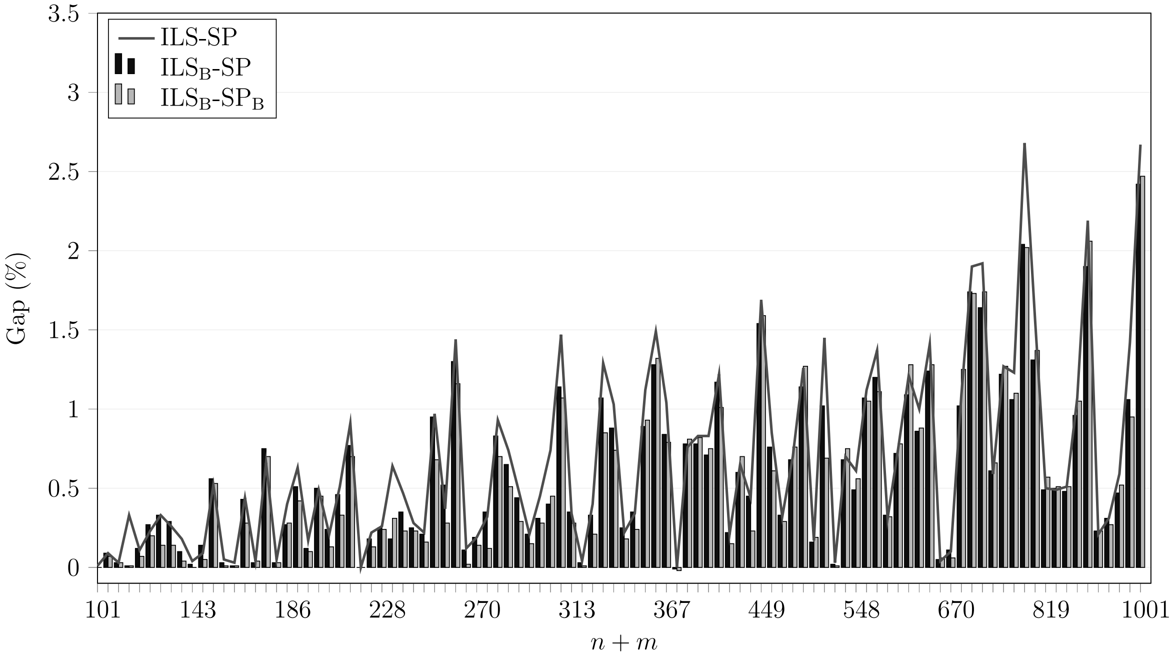


Figure 3.9: Average gap (%) for the X instances

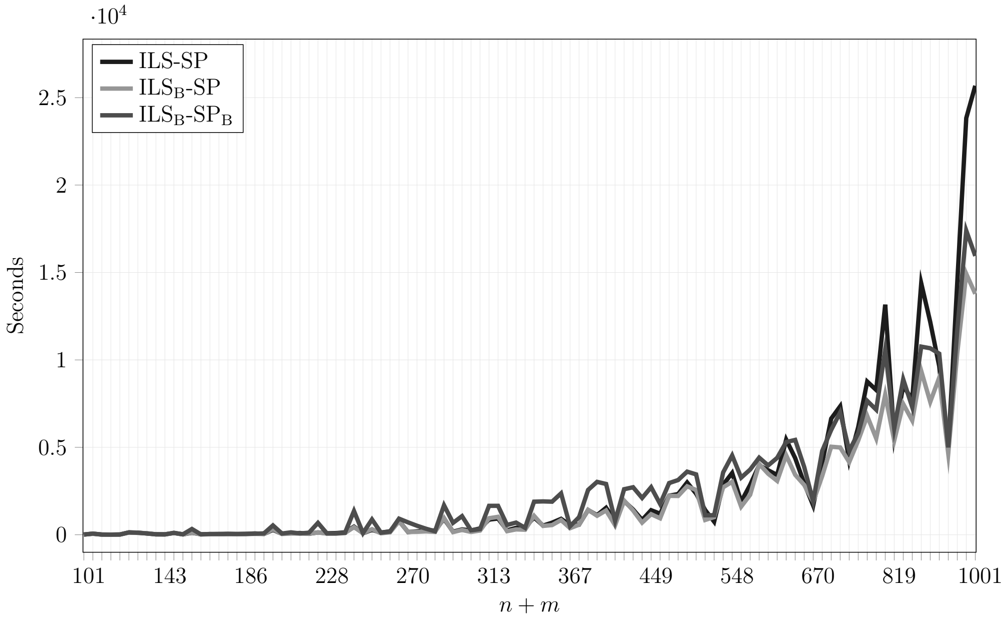


Figure 3.10: Average CPU time (s) for the X instances

Table 3.17: Summary of the results found for the X instances with 50%, 66% and 80% linehauls

	ILS-SP	ILS_B-SP	ILS_B-SP_B
50%			
Gap_best (%)	0.38	0.36	**0.33**
Gap_avg (%)	0.79	0.72	**0.66**
CPU (s)	2,788	**1,991**	2,775
#Ties	27	28	**33**
#Improvements	6	5	**12**
66%			
Gap_best (%)	0.27	**0.23**	**0.23**
Gap_avg (%)	0.64	0.56	**0.52**
CPU (s)	2,490	**1,858**	2,528
#Ties	22	26	**28**
#Improvements	**13**	10	11
80%			
Gap_best (%)	0.23	**0.19**	**0.19**
Gap_avg (%)	0.57	0.49	**0.48**
CPU (s)	2,521	**2,168**	2,677
#Ties	**22**	21	**22**
#Improvements	**12**	**12**	10

3.7 Concluding remarks

In this chapter, we proposed two branch-cut-and-price (BCP) approaches based on different mathematical formulations for the vehicle routing problem with backhauls (VRPB). While in one formulation the columns are based on complete routes ($\mathcal{F}_1$), in the other one the columns are based on separate linehaul and backhauls paths ($\mathcal{F}_2$). The BCP algorithms were implemented using the VRPSolver and they contain several successful methodological ingredients such as ng-routes/paths, limited memory rank-1 cuts, rounded capacity cuts, strong branching, route enumeration, arc elimination using reduced costs and dual stabilization.

Although it was proven that the linear relaxations of the formulations are equally strong, we demonstrated that rank-1 cuts for $\mathcal{F}_1$ may be stronger than the same type of cuts for $\mathcal{F}_2$. However, computational experiments on well-known benchmark instances revealed that the BCP algorithm over $\mathcal{F}_2$ has a better overall performance in practice. Nevertheless, both algorithms were capable of finding the optimal solutions for all instances, some of them for the first time. We also performed tests on a newly proposed set of instances that were derived from the X dataset of Uchoa et al. [115]. The BCP implementation based on $\mathcal{F}_2$ yielded better results than the one based on $\mathcal{F}_1$, confirming the efficiency of using separate variables for linehaul and backhaul paths. Finally, we conducted experiments on benchmark instances for the HFFVRPB, and of the VRPBTW. For these two problems, all benchmark instances were solved to optimality.

We also proposed three heuristic strategies for the VRPB. The first transforms a VRPB instance into an AVRPMB instance and then the ILS-SP is directly applied. The second adapts ILS-SP for the VRPB itself by only allowing feasible solutions to be explored in all steps of the algorithm. The third extends the second strategy by modifying the SP formulation used in the matheuristic to specifically tackle the VRPB. Their overall performance can be ranked according to the criteria presented in Table 3.18.

Table 3.18: Ranking of the strategies according to the criteria defined by Cordeau et al. [30]

Criterion	ILS-SP	ILS_B-SP	ILS_B-SP_B
Simplicity	**1st**	2nd	3rd
Flexibility	**1st**	2nd	3rd
Speed	2nd	**1st**	2nd
Accuracy	3rd	2nd	**1st**

Despite its generality, ILS-SP still seems a promising alternative for solving the VRPB. ILS_B-SP offers an interesting compromise between speed and accuracy. ILS_B-SP_B is useful to systematically find high quality solutions, especially for more challenging instances. Overall, all strategies managed to achieve extremely competitive results, outperforming the best methods on classical benchmark instances and even obtaining two new improved solutions. They also found high quality solutions for the newly introduced dataset involving up to 1,000 customers, with aggregate average gaps less than or equal to 0.67%.

In future studies, we suggest to extend the proposed BCP algorithms to other VRPB variants considering, for example, multiple depots and mixed routes (when the linehaul-backhaul precedence constraint is relaxed) [99]. An effective extension for problems with mixed routes might be especially challenging because of the existence of successful approaches for the VRP with simultaneous pickup and delivery [107, 108], which generalizes the VRP with mixed backhauls. On the other hand, variants with multiple depot and without mixed routes can take advantage of pricing via two graphs (inspired by formulation $\mathcal{F}2$).

Chapter 4

A POPMUSIC matheuristic for the capacitated vehicle routing problem

4.1 Introduction

The CVRP is one of the most widely studied problems in combinatorial optimization and operations research. The CVRP is the prototypical vehicle routing problem. New ideas are often first proposed and tested on CVRP and then generalized to other routing variants. It can be formally defined as follows. Let $G = (V, E)$ be a complete undirected graph, such that $V = \{0, 1, \ldots, n\}$ is the set of vertices and E is the set of edges, where vertex 0 represents a depot and $V_+ = \{1, \ldots, n\}$ a set of customers. There is a non-negative cost c_{ij} for each edge $\{i, j\} \in E$ and a demand d_i for each customer $i \in V_+$. The vehicle capacity is denoted by Q. A route is a path that begins and ends at the depot. A solution consists of a set of routes that respect the following constraints: (i) each customer must be visited exactly once by one of the routes; (ii) the sum of the customer demands in a route can not exceed the vehicle capacity. The objective is to find a set of routes with the minimum total cost.

Given that CVRP is NP-hard, most of the algorithms proposed for this problem are heuristics [70]. The best performing published algorithms are: the *iterated local search with set partitioning* (ILS-SP) [106], *knowledge-guided local search* (KGLS) [7], *hybrid genetic search* (HGS) [121], *slack induction by string removals* (SISRs) [27], *fast ILS localized optimization* (FILO) [1], and *adaptive iterated local search with path-relinking* (AILS-PR) [80]. ILS-SP combines the well-known ILS [77] with a *set partitioning* (SP) model. The SP model attempts to build unexplored solutions from the set of routes associated with the local minima found by previous runs of the local search. KGLS presents an efficient *guided local search* (GLS) with three complementary operators using ideas from sequential search and pruning, as well as a problem-specific knowledge to penalize “bad" edges. HGS is a population-based evolutionary search that also makes use of local search (in a step called *education*) and a sophisticated mechanism for controlling population diversity. Among the key components of HGS, we can mention the management of a subpopulation with infeasible solutions, as well as the individual evaluation (a.k.a. *fitness*) driven by the solution cost and its contribution to population diversity. More recently, Vidal [119] introduced a new neighborhood called SWAP* to the HGS, which significantly boosted the original method for the CVRP. SIRS is a *ruin & recreate* local search guided by simulated annealing (SA) [65]. The ruin procedure removes *strings* (sequence of consecutive customers) from routes (inducing a *capacity slack*), whereas the recreate procedure reinserts the removed customers in the ruined solution in a greedy manner. FILO is a scalable heuristic that employs novel and existing acceleration techniques during the main iterative part based on ILS, whereas it uses an SA-based acceptance criterion to get a continuous diversification. Finally, AILS-PR is a new hybridization of ILS with path-relinking, which is equipped with an automatic mechanism to control the diversity step to escape from local optima.

On the other hand, the exact methods for CVRP have advanced considerably in recent years [31, 95]. The state-of-the-art results are achieved by *branch-cut-and-price* algorithms [87, 92], which combine column and cut generation with several additional mechanisms. According to the experiments carried out in Uchoa et al. [115], this type of algorithm is able to produce optimal solutions for almost all instances with up to 250 customers, and in some cases, it can solve even larger instances (the largest one already solved has 654 customers). An important observation on the behavior of modern *branch-cut-and-price* algorithms for CVRP, explored in this work, is the following: while instances with more than 200 customers usually take hours or even days to be solved, many instances with up to 150 customers can be solved in few minutes, and many instances with up to 100 customers can be solved in seconds.

The algorithms that hybridize metaheuristics with mathematical programming approaches [63] are often known as *matheuristics*. Such methods have already been proposed for several optimization problems, including vehicle routing [4, 71]. According to Archetti and Speranza [4], one of the types of *matheuristics* is based on the decomposition of the original problem into smaller subproblems that can be solved (optimally or sub-optimally) through mathematical programming models. This chapter proposes a simple Partial OPtimization Metaheuristic Under Special Intensification Conditions (POPMUSIC) [110] for the CVRP that uses a modern branch-cut-and-price algorithm to solve subproblems (exactly or heuristically). The general idea of POPMUSIC is to optimize subproblems, defined by parts of a solution until a local minimum is reached. This type of algorithm has been shown to be effective for different problems [110], including vehicle routing variants [67, 86] and the famous traveling salesman problem [109]. The main difference between the proposed POPMUSIC and the existing ones for VRP (like Ostertag et al. [86] and Lalla-Ruiz and Voß [67]) is the use of the generic and state-of-the-art exact algorithm by Pessoa et al. [92] as subproblem solver.

The remainder of this chapter is organized as follows. In Section 4.2, the proposed POPMUSIC matheuristic for the CVRP is presented. Section 4.3 describes the modifications to the published branch-cut-and-price algorithm used for solving the subproblems. Section 4.4 presents and analyses the results of extensive computational experiments, including those ones for two extensions of the CVRP: heterogeneous fleet vehicle routing problem (HFVRP) and vehicle routing problem with backhauls (VRPB). Finally, in Section 4.5, the final conclusions are presented, as well as suggestions for future work.

4.2 A POPMUSIC matheuristic for the CVRP

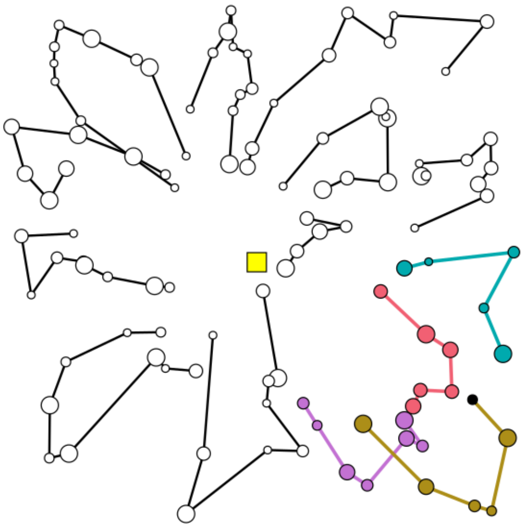
Algorithm 6 shows the pseudocode of the proposed POPMUSIC matheuristic for the CVRP, which has four inputs: (i) an initial solution S; (ii) an algorithm $\mathcal{A}$ to solve subproblems; (iii) initial value α for the current *target dimension* dim_{sp} (upper limit on the dimension of subproblems); (iv) step size δ to increase dim_{sp}. The algorithm's output is a (possibly) improved solution S obtained after solving a sequence of subproblems. A solution S is a set $\{r_1, \dots, r_K\}$ of K routes, whereas the set of customers visited by a route r is denoted by $C(r)$. A set $V_{sp} \subseteq V_+$ represents the CVRP subproblem associated with the subgraph $G[\{0\} \cup V_{sp}]$. We will refer to solutions for subproblems as subsolutions. In addition, in the description of the algorithm, we will consider that $c_{ji} = c_{ij}, \forall \{i, j\} \in E$, and $c_{ii} = 0, \forall i \in V_+$.

The algorithm keeps the current target dimension dim_{sp}, which is the upper limit for $|V_{sp}|$. dim_{sp} is initialized to α at line 4. The set $\mathbf{\Pi}$, initialized at line 5, keeps all subproblems already explored during the search together with their subsolutions. Formally, $\mathbf{\Pi}$ is a set of all pairs (V', S'), such that subproblem with set $V' \subset V_+$ of vertices is already solved, and S' is its subsolution. At first, a random permutation of the customers in V_+ produces the array L (line 7). For a given value of dim_{sp}, each customer $i \in V_+$ is used

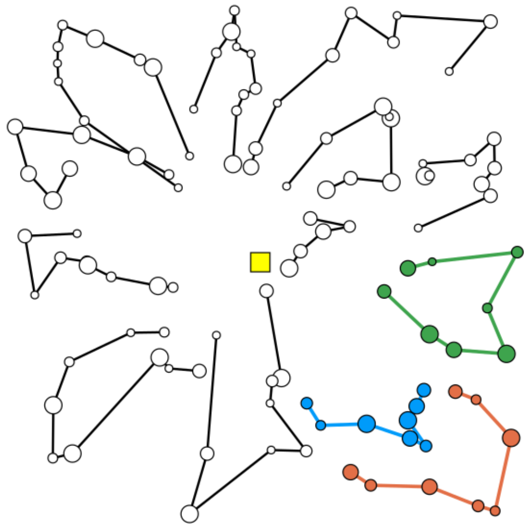
Algorithm 6: A POPMUSIC matheuristic for the CVRP

1 **Data:** V, E, c, d, Q
2 **Input parameters:** initial solution S, algorithm $\mathcal{A}$, α, δ
3 **Output:** (Possibly) improved solution S
4 $dim_{sp} \leftarrow \alpha$
5 $\mathbf{\Pi} \leftarrow \emptyset$
6 **while** *time limit is not exceeded* **and** $dim_{sp} \leq |V_+|$ **do**
7     $L \leftarrow$ a vector with a random order of V_+
8   **for** $z = 1, 2, \ldots, n$ **do**
9         $i \leftarrow L[z]$
10   /* Build the subproblem for seed i */
11         $V_{sp} \leftarrow \emptyset$
12         $R \leftarrow \emptyset$
13   **while** $|V_{sp}| < dim_{sp}$ **do**
14             $\hat{r} \leftarrow \operatorname*{argmin}_{r \in S, r \notin R} \left\{ \min_{j \in C(r)} c_{ij} \right\}$
15   **if** $|V_{sp}| + |C(\hat{r})| \leq dim_{sp}$ **then**
16                 $V_{sp} \leftarrow V_{sp} \cup C(\hat{r})$
17                 $R \leftarrow R \cup \{\hat{r}\}$
18   **else**
19   Go to the line 21
20   /* If the same or a larger subproblem has not yet been solved, solve V_{sp} */
21   **if** $V_{sp} \not\subseteq V'$ *for all* $(V', S') \in \mathbf{\Pi}$ **then**
22   Let S_{sp} be the subsolution for V_{sp} in S
23             $\mathbf{\Pi} \leftarrow \mathbf{\Pi} \cup (V_{sp}, S_{sp})$
24   Solve V_{sp} with the algorithm $\mathcal{A}$ using $cost(S_{sp})$ as the initial upper bound
25   Let S'_{sp} be the subsolution found by the algorithm $\mathcal{A}$, if any
26   **if** S'_{sp} *is found and* $cost(S'_{sp}) < cost(S_{sp})$ **then**
27   Replace (V_{sp}, S_{sp}) by (V_{sp}, S'_{sp}) in $\mathbf{\Pi}$
28   Update S by replacing subsolution S_{sp} by S'_{sp}
29   Go to the line 7
30   /* Increase the current target dimension */
31     $dim_{sp} \leftarrow dim_{sp} + \delta$

as a *seed* to construct a subproblem V_{sp} at lines 11–19. A subproblem V_{sp} with at most dim_{sp} customers is constructed iteratively by including the routes in the current solution S that are closest to vertex i. The distance from i to each route $r \in S$ is determined by the smallest cost of edges connecting i and the vertices in r (line 14). The routes already included in V_{sp} are stored in R to avoid repetitions (line 17). The construction of subproblem V_{sp} is finished when the next selected route $\hat{r}$ cannot be included in subproblem due to the upper limit dim_{sp} on the subprobem dimension (line 19). Figure 4.1a illustrates the construction of the subproblem for an instance having 109 customers and the current target dimension $dim_{sp} = 30$. First, the route containing the seed (in black) is included in the subproblem. The second selected route is the red one, while the third is the blue one, and the fourth is the purple one. Adding a fifth route would exceed dim_{sp}, so the obtained subproblem has 24 customers.



(a) Initial solution and a constructed subproblem. Seed customer is marked in black.



(b) Improved solution after finding a better subsolution

Figure 4.1: Constructing and solving a subproblem. Depot is the yellow square, and customers are circles with diameter proportional to its demand. For the sake of visualization, the edges adjacent to the depot are not depicted.

The algorithm solves generated subproblem V_{sp} only if it is neither equal nor contained in any subproblem V' already solved before (line 21). Indeed, the $\mathbf{\Pi}$-based condition avoids wasting time on subproblems, i.e., current subsolutions of which are unlikely to be improved because the same or a larger subproblem has been solved already. The solved subproblems together with their solutions are added to set $\mathbf{\Pi}$ at line 23. At line 24, the algorithm $\mathcal{A}$ tries to improve the subsolution S_{sp} of S for current subproblem V_{sp}. As algorithm $\mathcal{A}$, we use a branch-cut-and-price based heuristic described in Section 4.3. It is important here to use the cost of the known solution S_{sp} for subproblem V_{sp} to improve the performance of the branch-cut-and-price algorithm. Finally, if the solution S'_{sp} found by $\mathcal{A}$ is better than S_{sp}, then S is updated, and the search is restarted for the same target dimension dim_{sp}: all customers will be used again as seeds without increasing dim_{sp}. Figure 4.1b depicts an example of such an improved solution. If all seeds fail to produce an improving subsolution, then the target dimension dim_{sp} is increased by δ, so that larger subproblems can be explored (line 31). The algorithm is interrupted when the time limit is reached or when the target dimension exceeds the number of customers (line 6). From now on, we refer to Algorithm 6 as POP.

4.3 A branch-cut-and-price heuristic to solve subproblems

The algorithm $\mathcal{A}$ in POP, used for solving the subproblems, is an adaptation of the generic Branch-Cut-and-Price (BCP) algorithm proposed by Pessoa et al. [92], which is a state-of-the-art exact algorithm for many VRP variants, including the CVRP. BCP is a well-known technique that incorporates column and cut generation in a branch-and-bound procedure. In particular, the BCP by Pessoa et al. [92] includes advanced elements, such as: (i) ng-path relaxation [10]; (ii) rank-1 cuts with limited memory [22, 62, 88]; (iii) path enumeration [9, 29]; (iv) rounded capacity cuts [68]; (v) bucket graph based bi-directional labeling algorithm [100]; (vi) edge elimination based on reduced costs [60, 100]. The reader is referred to Pessoa et al. [92] for more details about the BCP algorithm.

Since the methodology proposed in this work is a matheuristic, optimality does not need to be preserved by the BCP. Thus, we turn the BCP algorithm into a heuristic (named BCP_H) by:

- Imposing a branch-and-bound node limit of 10 and time limit of 3,600 seconds;
- Using the *false gap mechanism*, described next;
- Using a restricted master heuristic, described below.

As mentioned above, the BCP algorithm uses an elimination procedure that removes edges from graph G by exploiting reduced cost arguments. In particular, if the minimum reduced cost of a path passing by an edge $e \in E$ is not smaller than the gap between the current upper bound and the lower bound obtained by the column generation procedure, then edge e can safely be removed from the graph G, as no improving solution contains this edge. Removing edges makes subsequent calls to the labeling algorithm used for solving the pricing problem faster.

In addition to the edge elimination, path enumeration is also dependent on the gap. This procedure tries to enumerate all possible paths with reduced cost smaller than the current gap between upper and lower bounds. If path enumeration is successful (i.e., the number of enumerated paths is less than, say, one million; enough to store them in a table), the pricing problem from now on is solved by inspection. The inspection of enumerated paths is usually much faster to perform than to call the labeling algorithm. If the number of enumerated routes is sufficiently small (less than 10,000), the current node in the search tree can be finished by adding all enumerated routes to the restricted master and solving it as an IP (using a general solver like CPLEX).

The previous two paragraphs show the importance of having very good upper bounds (and, therefore, smaller gaps) for reducing the running time of the BCP algorithm. In fact, that is why POP solves smaller subproblems first (easy even with not so good upper bounds), so the solution of larger subproblems can benefit from already improved upper bounds. To further reduce the running time, the false gap mechanism artificially decreases the gaps when performing edge elimination and route enumeration. The false gap is defined as $FG = (UB - LB)/FGF$, where the false gap factor $FGF > 1$ is a parameter. Application of the false gap mechanism can result in removing edges or paths which participate in an improving solution. However, experiments indicate that such an outcome occurs rarely when one uses a moderate value for FGF (we tested $FGF = 3$).

Another difference from the default BCP algorithm by Pessoa et al. [92] consists in using an additional heuristic (similar to the one proposed in [91]). It is called after the convergence of column and cut generation at every node of the search tree. The idea is to further decrease the false gap (dividing it by two in each iteration) until it is possible to complete the path enumeration. Then, the 10,000 routes with smaller reduced costs are used to create an IP that is solved by CPLEX.

4.4 Computational experiments

The proposed algorithm POP was coded in Julia language version 1.4.2. The algorithm BCP_H to solve subproblems was obtained by parameterizing the CVRP demo application of VRPSolver [23]. The parameters are described in Section 4.4.3. VRPSolver, freely available for academic use, implements the generic BCP algorithm proposed by Pessoa et al. [92]. It makes use of the BaPCod C++ library [117] as a BCP framework combined with the C++ implementations by [100] for solving pricing problems, route enumeration, and separation of rank-1 cuts. It also uses CVRPSEP package [78] for separating rounded capacity cuts. Finally, VRPSolver uses CPLEX 12.9 to solve the LP relaxations and the MIPs over the enumerated paths.

All experiments with POP were performed on a 2 Deca-core Haswell Intel Xeon E5-2680 v3 server with 2.50 GHz and 128 GB of RAM, where each algorithm was executed on a single thread for each instance. Parallel runs for several different instances were performed on the same machine to speed up the experiments, effectively reducing the amount of memory allocated to each process.

4.4.1 Benchmark instances

The tests were performed on the 57 largest instances of the benchmark set X [115], ranging from 303 to 1001 vertices. Indeed, set X is currently the main benchmark used to assess the performance of all recent exact and heuristic algorithms for the CVRP. We skipped the 43 instances with less than 300 customers because most of those instances are now relatively easy for modern heuristics and even for modern exact algorithms. In fact, 39 of them have proved optimal solutions.

For a deeper analysis of some experiments, we split the 57 instances into two subsets: the subset X_S of 29 instances with $n/K_{min} \leq 10.8$ (i.e., instances with short routes), and the subset X_L composed by the other 28 instances (i.e., instances with long routes). The K_{min} value is an instance attribute that means the minimum possible number of routes that a solution can have. For example, the instance X-n561-k42 belongs to X_L because $n/K_{min} = 561/42 = 13.6 > 10.8$. Extensive experiments presented in [87] indicate that modern branch-cut-and-price algorithms for CVRP, like the one we use to solve subproblems in POP, perform considerably better on instances with shorter routes. Therefore, route size is a factor that is likely to affect the overall performance of POP.

Moreover, in the preliminary experiments used for calibration, we consider a small representative subset X_R having only seven instances. The choice of X_R is described in Appendix K.

The gap of a solution S is calculated as $100 \cdot ((cost(S) - \text{BKS})/\text{BKS})$, where BKS is the best known solution in the CVRPLIB[1], only disregarding the solutions found by executions based on the proposed POP approach. Several optimization groups compete for improving the best known solutions for the instances in CVRPLIB. In fact, there were 24 updates in 2020 by seven distinct groups. Updating a BKS in CVRPLIB does not require the publication of an article; one only has to send the improved solution to be checked, even if the improvement is by only one unit. It is not necessary to describe how the solution was obtained. The competing groups may perform long runs of their methods, try several random number seeds, and even resort to special calibration. Thus, those BKSs are likely to be very close to optimum values.

[1] BKSs available in the CVRP Library (`http://vrp.atd-lab.inf.puc-rio.br/`) on October 31, 2020

4.4.2 Obtaining an initial solution

The initialization of POP is a critical issue. Preliminary experiments showed that it did not work so well as a stand-alone algorithm. It means that if it is initialized with a low-quality solution S obtained by a simple constructive heuristic, the overall performance of POP is not competitive with the best existing heuristics. In fact, we are proposing POP essentially as an effective way of improving solutions that are already reasonably good, possibly obtained by running some heuristic.

We report results obtained by different variants of POP^1, which uses the HGS heuristic by [121] to obtain the initial solution. However, as shown in Appendix L, the HGS is more effective if the entire algorithm is restarted (with a different random number seed) after $50,000$ iterations without any improvement (a method hereafter called HGS^r). Notation POP^1_t defines the variant that starts POP with the solution obtained by HGS^r in t hours. We tested 4 values for t: 0.01 (36 seconds), 0.125 (450 seconds), 0.5 (1800 seconds), and 2 (7,200 seconds). Of course, the initialization time is included in the overall time. For example, in variant $\text{POP}^1_{0.5}$, which is run for 32 hours per instance, HGS^r obtains the initial solution in 0.5 hours, and then POP spends 31.5 hours improving the initial solution.

The results obtained by several variants POP^1 over the time horizon of 32 hours are compared with those by HGS^r itself. We also perform some comparisons with a second heuristic, the ILS-SP proposed by [106]. As shown in [115], although the HGS is on average substantially better than the ILS-SP, there are some instances (usually those with very short routes) where the ILS-SP is superior.

4.4.3 Parameterization of the subproblem solver

The default parameterization of the VRPSolver CVRP demo is calibrated to find optimal solutions for hard instances having around 200-300 customers. As POP needs to solve many smaller problems, we propose an alternative parameterization that works better inside POP. Appendix M presents the default and the proposed parameterizations, whose performances are compared in Figure 4.2 by running $\text{POP}^1_{0.5}$ on the X_R instances over 8 hours (per instance). The convergence curves of the algorithms show the average gap found at different times.

The figure shows the superior performance obtained when the heuristic version BCP_H is used to solve the subproblems. BCP_H is obtained from the exact

BCP using the proposed parameterization and by setting three additional parameters: `RCSPfalseGapFactor` to 3 (this activates the false gap mechanism described in Section 4.3), `MaxNbOfBBtreeNodeTreated` to 10 (maximum number of nodes in the branch-and-bound tree), and `GlobalTimeLimit` to 3600 seconds (maximum time for solving a subproblem) in the proposed parameterization. All POPMUSIC results hereafter are obtained using algorithm BCP_H as the subproblem solver.

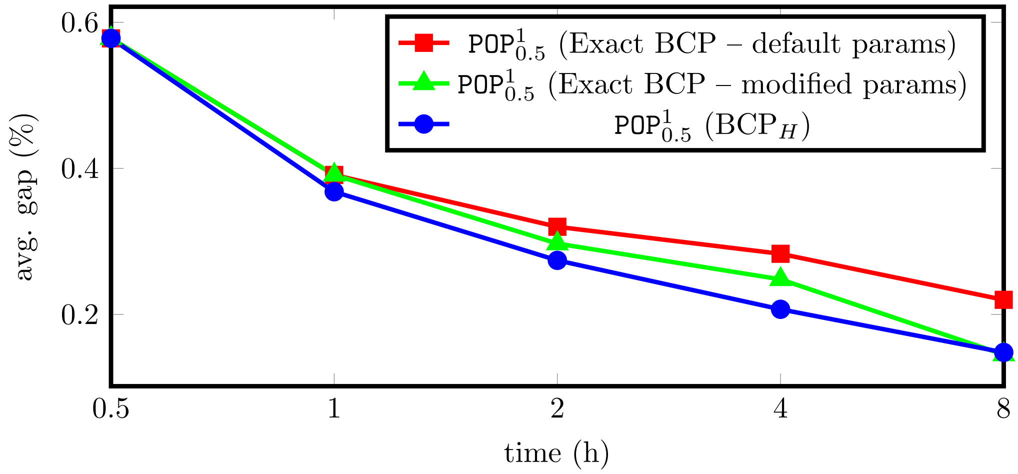


Figure 4.2: $\text{POP}^1_{0.5}$ with three different parameterizations of VRPSolver. The time axis is on a log_2 scale.

4.4.4 Calibrating parameters α and δ

Table 4.1 shows the performance of $\text{POP}^1_{0.5}$ for different values of parameters α and δ. Each setting was applied to the X_R instances over the horizon of 8 hours. The setting $(\alpha = 50, \delta = 40)$ achieved the best performance for 2, 4, and 8 hours. Therefore, all POPMUSIC results below are obtained with parameterization $(\alpha = 50, \delta = 40)$.

Table 4.1: Avg. gap (%) of $\text{POP}^1_{0.5}$ on X_R instances for different values of α and δ.

Time (h)	$\alpha = 25$ $\delta = 10$	$\alpha = 25$ $\delta = 25$	$\alpha = 25$ $\delta = 40$	$\alpha = 50$ $\delta = 10$	$\alpha = 50$ $\delta = 25$	$\alpha = 50$ $\delta = 40$	$\alpha = 75$ $\delta = 10$	$\alpha = 75$ $\delta = 25$	$\alpha = 75$ $\delta = 40$
1	0.399	**0.380**	0.448	0.387	0.390	0.383	0.475	0.460	0.440
2	0.327	0.303	0.294	0.310	0.308	**0.263**	0.344	0.328	0.326
4	0.272	0.245	0.244	0.274	0.242	**0.209**	0.241	0.236	0.248
8	0.198	0.170	0.165	0.213	0.193	**0.162**	0.183	0.214	0.198

4.4.5 Comparison of the algorithms ILS-SP, HGS^r, and POP^1 over 32 hours

Figure 4.3 and Table 4.2 show the gap convergence curves for HGS^r and POP^1 over the horizon of 32 hours.

Table 4.2: Average gap (%) of HGS^r and POP^1 executions at different times.

Instances	Time (h)	HGS^r	POP$^1_{0.01}$	POP$^1_{0.125}$	POP$^1_{0.5}$	POP$^1_2$
All	0.01	1.996	1.996	–	–	–
	0.125	0.836	1.180	0.836	–	–
	0.25	0.626	0.931	**0.564**	–	–
	0.5	0.484	0.708	**0.457**	0.484	–
	1	0.396	0.579	0.355	**0.317**	–
	2	0.330	0.420	0.271	**0.240**	0.330
	4	0.283	0.293	0.206	0.182	**0.171**
	8	0.236	0.201	0.164	0.138	**0.126**
	16	0.210	0.137	0.117	0.099	**0.090**
	32	0.184	0.091	0.084	0.076	**0.064**
X_S	0.01	1.790	1.790	–	–	–
	0.125	0.704	0.754	0.704	–	–
	0.25	0.547	0.524	**0.398**	–	–
	0.5	0.425	0.350	**0.303**	0.425	–
	1	0.345	0.268	**0.228**	0.232	–
	2	0.298	0.196	**0.163**	0.178	0.298
	4	0.262	**0.101**	0.114	0.131	0.108
	8	0.196	**0.058**	0.069	0.084	0.081
	16	0.176	**0.039**	0.048	0.056	0.055
	32	0.162	**0.017**	0.022	0.036	0.037
X_L	0.01	2.209	2.209	–	–	–
	0.125	0.973	1.621	0.973	–	–
	0.25	**0.708**	1.354	0.735	–	–
	0.5	0.546	1.080	0.617	0.546	–
	1	0.448	0.902	0.487	**0.404**	–
	2	0.363	0.652	0.383	**0.305**	0.363
	4	0.306	0.492	0.301	**0.235**	0.237
	8	0.278	0.349	0.263	0.193	**0.173**
	16	0.244	0.238	0.190	0.144	**0.127**
	32	0.206	0.167	0.149	0.117	**0.092**

The performance of POP$^1_{0.01}$ deserves a separate analysis. It illustrates the behavior of POP as an "almost stand-alone" matheuristic, starting from a medium quality solution. Such solutions can be rapidly obtained by any modern heuristic for the CVRP. The initial solutions provided by running HGS^r for 36 seconds have an average gap of about 2% from the BKS.

- The performance of POP$^1_{0.01}$ on instances with shorter routes (set X_S) is very good. After 900 seconds, it already provides solutions that are significantly better than those from HGS^r. After 4 hours, it is also consistently better than POP$^1_{0.125}$, POP$^1_{0.5}$, and POP$^1_2$ and reaches the excellent average gap of 0.017% in 32 hours. It is quite

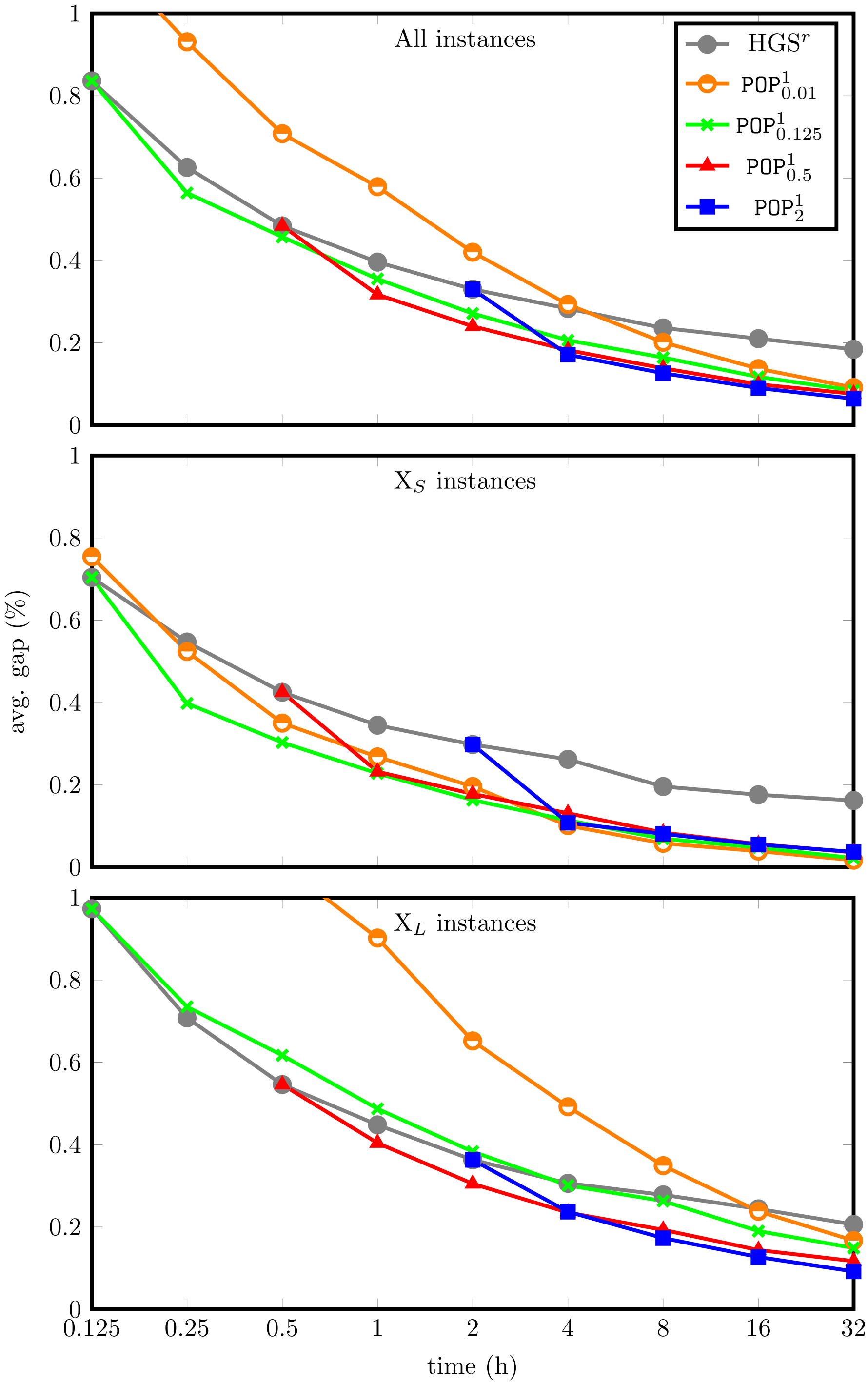


Figure 4.3: Convergence curves of POP^1 and HGS^r.

Table 4.3: Best solutions found by ILS-SP, HGS^r, and POP^1 after 32 hours.

Instance	BKS	ILS-SP	HGS^r	POP$^1_{0.01}$	POP$^1_{0.125}$	POP$^1_{0.5}$	POP$^1_2$
X-n303-k21	21736	21840	21739	21863	21837	21750	21751
X-n308-k13	25859	25881	25861	25876	25876	25876	25862
X-n313-k71	94044	94105	94046	94053	94053	94046	94046
X-n317-k53	78355*	78355	78355	78355	78355	78355	78355
X-n322-k28	29834*	29872	29848	29887	29887	29887	29880
X-n327-k20	27532	27743	27555	27573	27576	27573	27576
X-n331-k15	31102*	31108	31103	31103	31103	31103	31103
X-n336-k84	139135	139253	139210	139164	**139111**	139175	**139125**
X-n344-k43	42056	42096	42069	**42055**	**42050**	**42050**	42056
X-n351-k40	25919	26131	25935	**25896**	**25896**	**25896**	**25896**
X-n359-k29	51505	51997	51521	51583	51583	51583	51505
X-n367-k17	22814	22912	22814	22814	22821	22821	22814
X-n376-k94	147713*	147713	147713	147713	147713	147713	147713
X-n384-k52	65941	66382	66048	65941	65999	65956	65947
X-n393-k38	38260*	38273	38260	38260	38260	38260	38260
X-n401-k29	66163	66614	66222	66181	66220	66257	**66156**
X-n411-k19	19718	19811	**19717**	**19712**	19718	**19712**	**19712**
X-n420-k130	107798*	107798	107813	107798	107798	107798	107798
X-n429-k61	65449	65759	65489	65527	65467	65467	65455
X-n439-k37	36391*	36402	36395	36395	36395	36395	36395
X-n449-k29	55233	56131	55336	55332	55236	55259	55258
X-n459-k26	24139	24421	24184	24193	24208	24209	24160
X-n469-k138	221824*	221940	222203	221824	221824	221824	221824
X-n480-k70	89458	89821	89542	**89449**	**89449**	**89449**	**89449**
X-n491-k59	66510	67128	66633	66555	66539	66572	66514
X-n502-k39	69230	69315	69254	69232	69232	69232	69232
X-n513-k21	24201	24275	24201	24248	24249	24201	24201
X-n524-k153	154593*	154698	154774	154593	154593	154593	154593
X-n536-k96	94921	95697	95059	94948	**94915**	95205	95205
X-n548-k50	86700*	86710	86737	86701	86701	86701	86701
X-n561-k42	42717	43016	42744	42758	42773	42758	42758
X-n573-k30	50673	51074	50782	50807	50882	50742	50735
X-n586-k159	190423	190767	190581	**190365**	**190340**	**190375**	**190379**
X-n599-k92	108489	109147	108781	108498	108558	108517	**108462**
X-n613-k62	59535	60318	59671	59561	59544	59606	59656
X-n627-k43	62164	62762	62369	62182	62213	62245	62266
X-n641-k35	63694	64449	64019	63773	63989	63919	63863
X-n655-k131	106780*	106780	106810	106780	106780	106780	106780
X-n670-k130	146332	147286	147144	146346	146340	146411	146461
X-n685-k75	68205	68682	68436	68260	68315	68318	68354
X-n701-k44	81934	82907	82310	82085	81984	81970	82021
X-n716-k35	43412	44091	43572	43443	43491	43498	43489
X-n733-k159	136250	136900	136365	**136245**	**136237**	136278	**136223**
X-n749-k98	77365	78177	77706	77380	77399	**77360**	**77342**
X-n766-k71	114454	115413	114701	114573	114707	114640	114682
X-n783-k48	72445	73627	72809	72696	72592	72605	72704
X-n801-k40	73305	73939	73548	73446	73445	73368	73362
X-n819-k171	158247	159249	158696	**158128**	**158191**	**158222**	**158211**
X-n837-k142	193810	194901	194264	193820	**193793**	**193800**	193822
X-n856-k95	88965	89143	89062	89030	89030	89030	89030
X-n876-k59	99299	100357	99748	99583	99437	99479	99428
X-n895-k37	53860	54777	54266	54112	54080	54125	54045
X-n916-k207	329247	330773	329902	329305	**329213**	329305	329289
X-n936-k151	132725	134564	133440	132859	132882	132863	132942
X-n957-k87	85465	85887	85633	85485	85468	85492	85473
X-n979-k58	118987	120015	119339	119430	119059	119073	119040
X-n1001-k43	72359	73810	72766	72714	72506	72460	72486
Avg. gap (%)		0.629	0.184	0.091	0.084	0.076	**0.064**
Median gap (%)		0.607	0.117	0.027	0.031	0.032	**0.009**
Avg. gap (%) in X_S		0.468	0.162	**0.017**	0.022	0.036	0.037
Median gap (%) in X_S		0.474	0.117	0.005	**0.000**	0.002	**0.000**
Avg. gap (%) in X_L		0.796	0.206	0.167	0.149	0.117	**0.092**
Median gap (%) in X_L		0.779	0.138	0.150	0.135	0.091	**0.073**

interesting to note that the final gap after 32 hours obtained by POP^1_t on instances X_S gets *worse* as t increases. It seems that worse initial solutions used in $\text{POP}^1_{0.01}$

are still flexible enough to be transformed into good final solutions by the POP local search mechanism. On the other hand, the much better initial solutions used by $\text{POP}^1_{0.5}$, and POP^1_2 seem to be biased towards certain local minima that may not be so globally good.

- On the other hand, $\text{POP}^1_{0.01}$ performs poorly on instances with long routes (set X_L). It takes 4 hours to obtain an average gap of 0.492%, and it reaches the performance of HGS^r only after 16 hours. It is also consistently worse than POP^1_t, for $t \in \{0.125, 0.5, 2\}$.

The variants $\text{POP}^1_{0.125}$, $\text{POP}^1_{0.5}$, and POP^1_2 have a more robust performance. When the complete instance set X is considered, all of them are consistently better than HGS^r alone (i.e., after POP starts, their average gaps are smaller at all times). This is also true when X_S and X_L instances are considered separately. The only exception is the variant $\text{POP}^1_{0.125}$, which requires four hours to overcome HGS^r on X_L instances.

Table 4.3 reports the best solutions found by the algorithms ILS-SP, HGS^r, and POP^1 in 32 hours. BKSs marked with a * are proven optimal solutions. Solutions marked in bold are improvements over the BKSs. The variant POP^1_2 achieved the best average and median final gaps, with the exception of the average gap for instances X_S, where it is worse than the variants $\text{POP}^1_{0.01}$, $\text{POP}^1_{0.125}$, and $\text{POP}^1_{0.5}$.

4.4.6 Comparison of the algorithms HGS20 and POPMUSIC over 32 hours

When the work described in this chapter was already advanced, we were told[2] about the existence of a new implementation of HGS [119]. The new version, specialized to CVRP, is faster and includes one additional neighborhood called SWAP*. We will refer to that algorithm as HGS20. In fact, the performance of HGS20 is much superior to HGS^r, and thus it can definitely be considered as a state-of-the-art heuristic for CVRP. In this section, we test if POP can still improve HGS20 solutions.

On September 17, 2020, Thibaut Vidal kindly sent us the detailed results of ten 20-hour runs of the algorithm HGS20 on each of the X instances. Those runs are performed on Intel Xeon Gold 6148 @2.40GHz processors (PassMark single thread rating 2056) that are roughly equivalent to our processors (PassMark single thread rating 1840). Moreover, we have also received the solutions obtained after 0.125, 0.5, and 2 hours. Thus, we use

[2]Personal communication from Thibaut Vidal.

them as initial solutions in the variant POP_t^2. Figure 4.4 depicts the performance of the HGS20 and the variants POP^2 over 32 hours (note that HGS stops at 20 hours). We now analyze these results.

- For each running time, HGS20 obtains solutions with about half of the average gap of the solutions obtained by HGS^r, which is a remarkable improvement.
- Considering all X instances, the variant $\mathrm{POP}_{0.125}^2$ is consistently worse than HGS20 alone, producing inferior solutions for all times.
- Considering all X instances, the variant $\mathrm{POP}_{0.5}^2$ is slightly better than the HGS20 alone. On X_L instances, it only starts to be better at 16 hours.
- Finally, the variant POP_2^2 is clearly better than the HSG20 alone, even on X_L instances. This indicates that the proposed approach POP is indeed powerful. It is able to improve solutions obtained by a highly performing heuristic, at least in long runs (more than 2 hours).

Table 4.5 reports instance-by-instance statistics on the solutions found by the HGS20 (after 20 hours) and the variants POP^2 (after 20 and 32 hours). For the HGS20, the average cost and the best cost among ten runs are provided. For the variant POP_2^2, we give the average cost and the best cost for three runs. We did not have computational resources for running each instance ten times. In order to provide a direct comparison between methods, we also computed the average cost and the best cost of the first three runs of the HGS20.

It is obvious that one can always obtain better solutions by performing multiple runs of any randomized method and picking the best one. But it is still interesting to note that the variant POP_2^2 seems to be particularly well suited for performing multiple runs. In fact, for this variant, the average gap for a single 32-hour run is 0.042%, while the average gap for the best of only three runs decreases to 0.018%, a very substantial decrease. For the instances in the set X_S, the approach produces a remarkable gap of -0.001%.

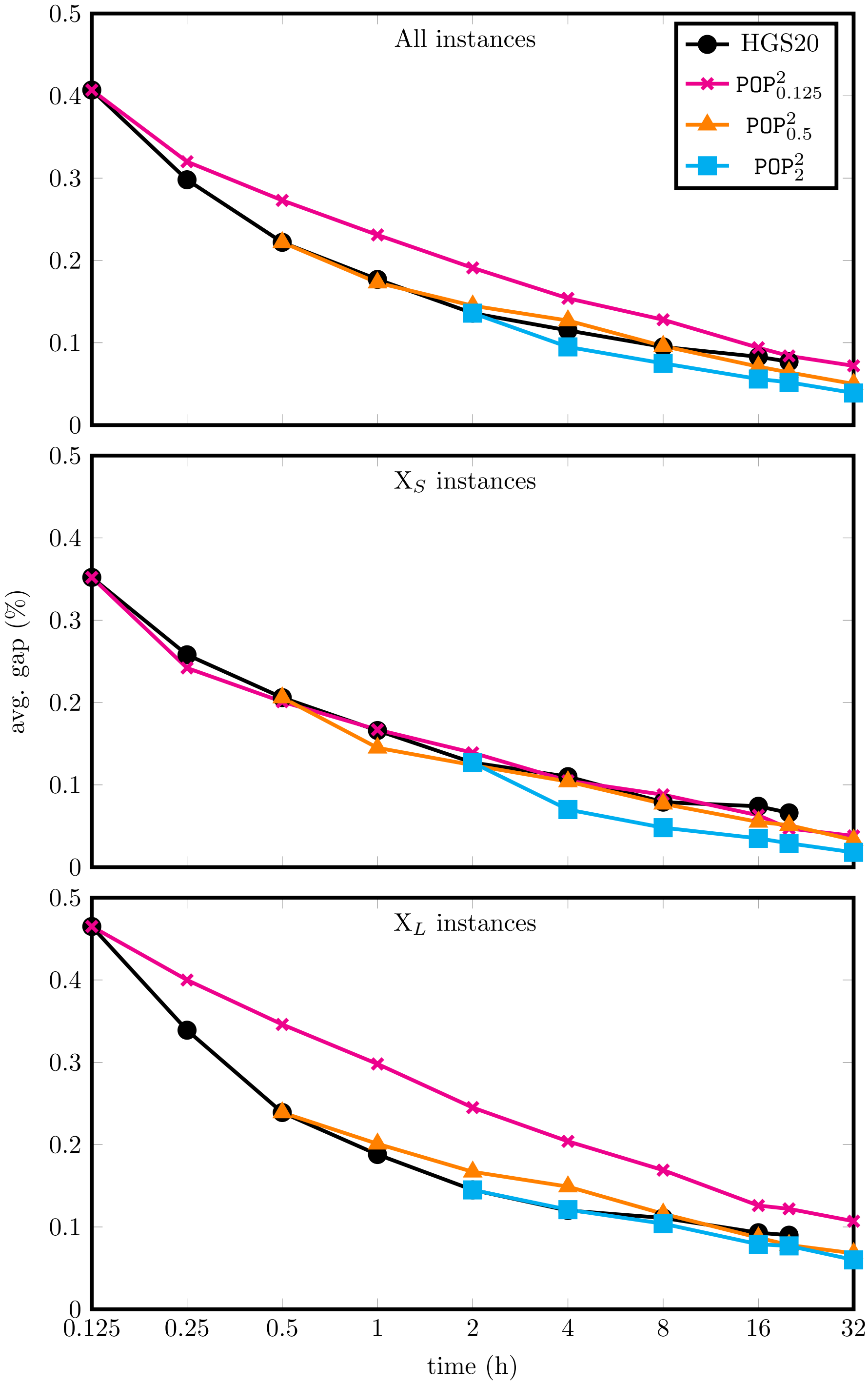


Figure 4.4: Convergence curves of POP^2 and HGS20.

Table 4.4: Average gap (%) of HGS20 and POP^2 executions at different times.

Instances	Time (h)	HGS20	$POP^2_{0.125}$	$POP^2_{0.5}$	POP^2_{2}
All	0.125	0.407	0.407	–	–
	0.25	**0.298**	0.320	–	–
	0.5	0.222	0.273	0.222	–
	1	0.177	0.231	**0.173**	–
	2	0.136	0.191	0.145	0.136
	4	0.115	0.154	0.127	**0.095**
	8	0.095	0.128	0.096	**0.075**
	16	0.083	0.094	0.071	**0.056**
	20	0.077	0.084	0.064	**0.052**
	32	–	0.072	0.050	**0.039**
X_S	0.125	0.352	0.352	–	–
	0.25	0.258	**0.242**	–	–
	0.5	0.206	**0.201**	0.206	–
	1	0.166	0.167	**0.145**	–
	2	0.127	0.139	**0.124**	0.127
	4	0.110	0.105	0.104	**0.070**
	8	0.079	0.088	0.077	**0.048**
	16	0.074	0.063	0.055	**0.035**
	20	0.066	0.047	0.051	**0.029**
	32	–	0.038	0.033	**0.018**
X_L	0.125	0.465	0.465	–	–
	0.25	**0.339**	0.400	–	–
	0.5	0.239	0.346	0.239	–
	1	**0.188**	0.298	0.201	–
	2	0.145	0.245	0.167	0.145
	4	**0.120**	0.204	0.149	0.121
	8	0.111	0.169	0.116	**0.104**
	16	0.093	0.126	0.087	**0.079**
	20	0.090	0.122	0.078	**0.077**
	32	–	0.107	0.068	**0.060**

Table 4.5: Detailed statistics for HGS20 and POP_2^2. Best gaps for 20 hours are underlined.

Instance	BKS	HGS20				POP_2^2			
		20 hours				20 hours		32 hours	
		avg. cost (10×)	best (10×)	avg cost (3×)	best (3×)	avg cost (3×)	best (3×)	avg cost (3×)	best (3×)
X-n303-k21	21736	21737.4	21736	21737.3	21736	21738.0	21738	21738.0	21738
X-n308-k13	25859	25859.0	25859	25859.0	25859	25859.7	25859	25859.7	25859
X-n313-k71	94044	94044.0	94044	94044.0	94044	94044.0	94044	94044.0	94044
X-n317-k53	78355*	78355.0	78355	78355.0	78355	78355.0	78355	78355.0	78355
X-n322-k28	29834*	29834.0	29834	29834.0	29834	29834.0	29834	29834.0	29834
X-n327-k20	27532	27532.0	27532	27532.0	27532	27532.0	27532	27532.0	27532
X-n331-k15	31102*	31102.0	31102	31102.0	31102	31102.3	31102	31102.3	31102
X-n336-k84	139135	139155.8	139137	139156.0	139137	139147.3	**139125**	139147.3	**139125**
X-n344-k43	42056	42053.6	**42050**	42051.7	**42050**	42052.0	**42050**	42052.0	**42050**
X-n351-k40	25919	25925.2	**25909**	25917.3	**25909**	25903.7	**25896**	25903.7	**25896**
X-n359-k29	51505	51535.2	51513	51534.0	51513	51518.7	51505	51518.7	51505
X-n367-k17	22814	22814.0	22814	22814.0	22814	22814.0	22814	22814.0	22814
X-n376-k94	147713*	147713.0	147713	147713.0	147713	147713.0	147713	147713.0	147713
X-n384-k52	65941	65977.1	65957	65979.3	65978	65971.0	65941	65971.0	65941
X-n393-k38	38260*	38260.0	38260	38260.0	38260	38260.0	38260	38260.0	38260
X-n401-k29	66163	66196.9	66180	66203.7	66192	66188.7	66180	66185.0	66178
X-n411-k19	19718	19712.8	**19712**	19712.0	**19712**	19713.7	**19712**	19713.7	**19712**
X-n420-k130	107798*	107804.7	107798	107802.0	107798	107822.0	107798	107798.0	107798
X-n429-k61	65449	65455.4	65449	65451.7	65449	65459.0	65455	65459.0	65455
X-n439-k37	36391*	36394.5	36391	36395.0	36395	36395.0	36395	36395.0	36395
X-n449-k29	55233	55294.1	55265	55282.7	55268	55291.3	55272	55288.0	55262
X-n459-k26	24139	24140.1	24139	24141.0	24139	24157.3	24139	24157.3	24139

Continued on next page

Table 4.5 – continued from previous page

Instance	BKS	HGS20				POP$_2^2$			
		20 hours				20 hours		32 hours	
		avg. cost (10×)	best (10×)	avg cost (3×)	best (3×)	avg cost (3×)	best (3×)	avg cost (3×)	best (3×)
X-n469-k138	221824*	221939.6	221848	221936.7	221855	221839.3	221824	221839.3	221824
X-n480-k70	89458	89459.2	**89457**	89461.3	**89457**	89457.0	**89449**	89449.0	**89449**
X-n491-k59	66510	66561.2	66521	66560.3	66521	66520.7	**66489**	66520.7	**66489**
X-n502-k39	69230	69228.5	**69227**	69229.3	**69228**	69226.0	**69226**	69226.0	**69226**
X-n513-k21	24201	24201.0	24201	24201.0	24201	24201.0	24201	24201.0	24201
X-n524-k153	154593*	154605.0	154605	154605.0	154605	154593.0	154593	154593.0	154593
X-n536-k96	94921	94991.5	94940	94991.0	94972	94948.0	**94915**	94943.0	**94915**
X-n548-k50	86700*	86710.0	86704	86706.0	86704	86700.7	86700	86700.7	86700
X-n561-k42	42717	42720.7	42717	42720.3	42717	42717.0	42717	42717.0	42717
X-n573-k30	50673	50747.1	50736	50742.0	50739	50741.0	50739	50738.3	50733
X-n586-k159	190423	190398.9	**190340**	190422.0	**190407**	190359.7	**190349**	190337.0	**190316**
X-n599-k92	108489	108554.2	108490	108562.0	108518	108486.0	**108457**	108484.7	**108453**
X-n613-k62	59535	59619.0	59549	59636.0	59602	59586.7	59536	59582.0	59536
X-n627-k43	62164	62273.3	62241	62275.3	62264	62264.7	62224	62254.0	62223
X-n641-k35	63694	63789.4	63738	63791.7	63758	63815.0	63763	63798.7	63763
X-n655-k131	106780*	106787.6	106780	106789.7	106786	106780.0	106780	106780.0	106780
X-n670-k130	146332	146641.3	146510	146642.7	146624	146514.3	146404	146514.0	146404
X-n685-k75	68205	68312.0	68272	68324.0	68317	68295.0	68257	68281.0	68257
X-n701-k44	81934	82107.6	81998	82152.0	82123	82115.3	82030	82080.0	82030
X-n716-k35	43412	43468.3	43446	43481.0	43460	43455.7	43445	43433.3	**43409**
X-n733-k159	136250	136306.9	136281	136304.0	136298	136213.7	**136195**	136213.7	**136195**
X-n749-k98	77365	77563.4	77463	77543.0	77463	77379.3	**77350**	77342.0	**77294**
X-n766-k71	114454	114687.2	114635	114689.0	114640	114678.7	114658	114627.0	114597

Continued on next page

Table 4.5 – continued from previous page

Instance	BKS	HGS20				POP$^2_2$			
		20 hours				20 hours		32 hours	
		avg. cost (10×)	best (10×)	avg cost (3×)	best (3×)	avg cost (3×)	best (3×)	avg cost (3×)	best (3×)
X-n783-k48	72445	72649.8	72550	72665.0	72620	72572.0	72524	72563.7	72515
X-n801-k40	73305	73377.2	73308	73366.7	73353	73387.3	73385	73349.0	73313
X-n819-k171	158247	158331.3	158263	158318.0	158263	158328.7	158298	158298.3	**158225**
X-n837-k142	193810	194023.3	193973	193985.0	193973	193822.0	**193756**	193813.7	**193739**
X-n856-k95	88965	88986.4	88966	88983.7	88966	88989.7	88989	88989.7	88989
X-n876-k59	99299	99557.8	99490	99540.0	99510	99447.0	99428	99419.7	99405
X-n895-k37	53860	54041.2	54007	54028.3	54007	54021.3	53969	54000.0	53960
X-n916-k207	329247	329565.5	329481	329552.7	329539	329325.0	329288	329304.0	329249
X-n936-k151	132725	133116.7	132998	133161.7	133124	132933.7	132900	132898.0	132861
X-n957-k87	85465	85505.4	85473	85504.3	85496	85496.0	85492	85494.3	85487
X-n979-k58	118987	119154.9	119120	119159.0	119130	119146.0	119038	119125.7	119022
X-n1001-k43	72359	72614.5	72541	72625.7	72541	72485.7	72449	72458.0	72427
Avg. gap (%)		0.079	0.043	0.080	0.060	0.052	0.027	**0.042**	**0.018**
Median gap (%)		0.047	0.008	0.045	0.014	0.016	0.000	**0.011**	**0.000**
Avg. gap (%) in X_S		0.068	0.031	0.068	0.048	0.029	0.007	**0.021**	**-0.001**
Median gap (%) in X_S		0.042	0.008	0.040	0.010	0.009	0.000	**0.002**	**0.000**
Avg. gap (%) in X_L		0.092	0.055	0.093	0.073	0.076	0.048	**0.064**	**0.037**
Median gap (%) in X_L		0.049	0.007	0.051	0.026	0.038	0.018	**0.034**	**0.010**

4.4.7 Directly improving BKSs in CVRPLIB

In the final experiment, we run algorithm POP using the BKS in CVRPLIB as the initial solution. Besides testing the open X instances, we also test very large instances with up to 30,000 customers in the XXL set [6] and also the open instances in the Golden set [55]. Table 4.6 presents the improved BKSs found by POP after 32 hours for X and Golden instances, and after 96 hours for the XXL instances.

Table 4.6: BKSs directly improved by POP

Instance	BKS	Improved BKS
X-n536-k96	94868	**94864**
X-n733-k159	136190	**136188**
X-n766-k71	114454	**114418**
X-n783-k48	72394	**72393**
X-n936-k151	132725	**132715**
X-n979-k58	118987	**118976**
X-n1001-k43	72359	**72355**
Antwerp1	477306	**477277**
Brussels1	501854	**501771**
Flanders1	7241290	**7240874**
Ghent1	469586	**469532**
Leuven1	192851	**192848**
Golden_16	1611.70	**1611.28**

4.4.8 Results for HFVRP and VRPB

The generality of the proposed POPMUSIC framework was tested by applying it to solve the HFVRP and VRPB, which are two classical extensions of the CVRP.

HFVRP

HFVRP extends the CVRP by considering a set $M = \{1, \ldots, m\}$ of different types of vehicles. For each $u \in M$, there are K_u available vehicles, an integer capacity Q_u, a fixed cost f_u per vehicle, and a travel cost c_{ij}^u associated to the edge $\{i, j\} \in E$ which is obtained by multiplying c_{ij} by a factor F_u. The objective is to minimize the sum of fixed and travel costs.

The proposed algorithm for the HFVRP will be referred to as POP^h. POP^h makes use of the state-of-the-art hybrid ILS (HILS) matheuristic proposed by Penna et al. [89] to produce the initial solutions and the VRPSolver HFVRP application [23, 92] as algorithm

$\mathcal{A}$ (subproblem solver). For POP^h, Algorithm 6 should be adapted as follows. The (M, Q, K, f, F) data inputs are now available, such that M is used to index the other ones to support different types of vehicles. A route r_i of a solution (or subsolution) has now an extra information $M(r_i)$ to indicate its vehicle type. Let p_u be the total number of vehicles of type u used by the routes in S (current solution) and q_u the same for R (selected routes to build the subproblem), then the HFVRP subproblem will have $K'_u = K_u - p_u + q_u$ available vehicles of type u. To avoid additional algorithmic changes, we use the original costs c_{ij} instead of c^u_{ij} to build the subproblems. POP^h uses the same VRPSolver and POP parameterizations as the proposed algorithm for the CVRP, i.e., the setting described in Appendix M and $(\alpha = 50, \delta = 40)$.

We conducted experiments on the large benchmark instances XH (based on the X instances) proposed by Pessoa, Sadykov, and Uchoa [90]. As in the CVRP, we considered only the 57 largest instances of XH (hereafter XH means this subset). In XH, there are 35 fleet size and mix (FSM) instances (denoted by XH-FSM) whose the available fleet is unlimited (i.e. $K_u = \infty, \forall u \in M$), and 22 instances (named XH-HVRP) with a potentially limited fleet. As usual in the HFVRP literature, there is no rounding applied to the travel costs.

Here, the BKS column for instances with up to 490 customers is set to the best upper bound reported by Pessoa, Sadykov, and Uchoa [90]. To our knowledge, there is no upper bound available in the literature for the other instances, so we set the BKS as the best solution found by the HILS in 32 hours. To force HILS to run up to 32h, we set the parameter $I_{MS} = \infty$ (number of restarts). Table 4.7 and Figure 4.5 show the comparison of HILS with POP$^h_{0.5}$. POP$^h_{0.5}$ was clearly superior than HILS, in such a way that its curve was dominant during all its execution. Notice that POP$^h_{0.5}$ managed to achieve a negative average gap in about four hours of execution (∼12% of the total time) for XH-FSM, and two hours of execution (∼6% of the total time) for XH-HVRP. Also, 46 new best solutions were found, including a remarkable gap of -8.46% obtained for the instance X670-FSMF. On the other hand, HILS found the new best solutions for four instances: X401-FSMFD, X627-HVRP, X876-FSMF, and X979-HVRP. The detailed results are reported in Appendix N.

VRPB

The proposed algorithm for the VRPB will be referred to as POP^b. In POP^b, we use the ILS$_\mathrm{B}$-SP$_\mathrm{B}$ described in Section 3.5.3 to produce the initial solutions. The algorithm

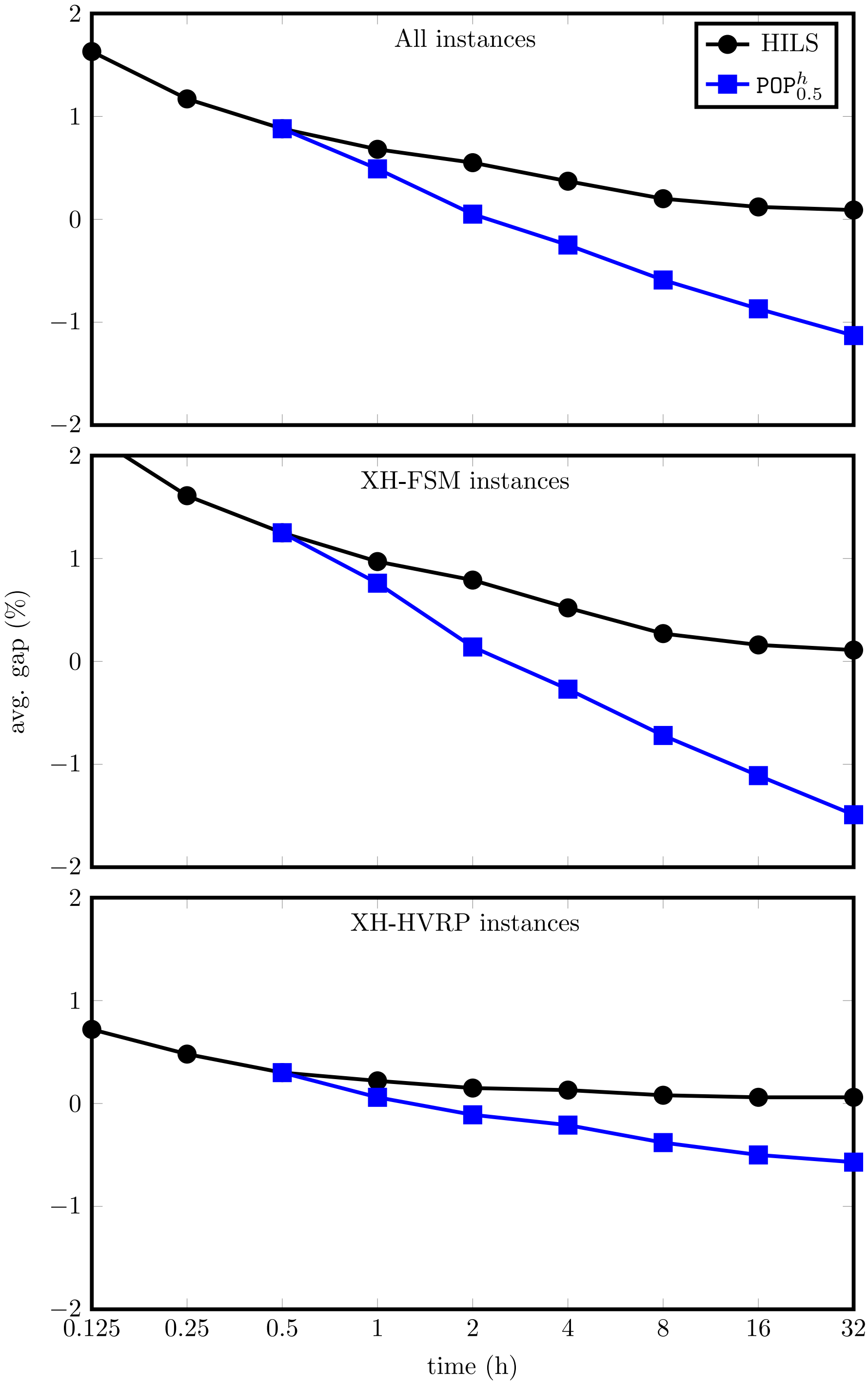


Figure 4.5: Convergence curves of HILS and $\mathrm{POP}^h_{0.5}$ for the XH instances of the HFVRP.

Table 4.7: Average gap (%) of HILS and POP^h executions at different times for the XH instances of the HFVRP.

Time (h)	XH		XH-FSM		XH-HVRP	
	HILS	POP$^h_{0.5}$	HILS	POP$^h_{0.5}$	HILS	POP$^h_{0.5}$
0.125	1.63		2.20		0.72	
0.25	1.17		1.61		0.48	
0.5	0.88	0.88	1.25	1.25	0.30	0.30
1	0.68	**0.49**	0.97	**0.76**	0.22	**0.06**
2	0.55	**0.05**	0.79	**0.14**	0.15	**-0.11**
4	0.37	**-0.25**	0.52	**-0.27**	0.13	**-0.21**
8	0.20	**-0.59**	0.27	**-0.72**	0.08	**-0.38**
16	0.12	**-0.87**	0.16	**-1.11**	0.06	**-0.50**
32	0.09	**-1.13**	0.11	**-1.49**	0.06	**-0.57**

$\mathcal{A}$ is the best BCP algorithm proposed in Section 3.4 using the same parameterization of the CVRP and HFVRP, as well as ($\alpha = 50, \delta = 40$). Note that when $|B| = 0$, the subproblem can be seen exactly as a CVRP subproblem, and then we used the original solver $\mathcal{A}$ (BCP$_H$).

As in the HVRP, the experiments were performed on the X-based instances proposed in Chapter 3 for the VRPB (hereafter named XB). Again, the experiments are limited only to instances with more than 300 customers, totaling 171 instances (since XH has 3 instances for each original one in X). In XB, the costs are rounded as in the X instances. For comparison, ILS$_\text{B}$-SP$_\text{B}$ was forced to run up to 32h by setting $maxIter = \infty$ (number of restarts). The column BKS is defined by the best upper bound reported in Chapter 3. Table 4.8 and Figure 4.6 show a clear advantage of POP$^b_{0.5}$ w.r.t. ILS$_\text{B}$-SP$_\text{B}$, where a negative gap of -0.01% was achieved by the former at 32 hours of execution. According to the detailed results in Appendix N, POP$^b_{0.5}$ found 84 new best solutions, whereas ILS$_\text{B}$-SP$_\text{B}$ found only 5 ones.

Table 4.8: Average gap (%) of ILS$_\text{B}$-SP$_\text{B}$ and POP$^b_{0.5}$ at different times for the XB instances of the VRPB.

Time (h)	ILS$_\text{B}$-SP$_\text{B}$	POP$^b_{0.5}$
0.125	1.27	
0.25	1.01	
0.5	0.90	0.90
1	0.83	**0.58**
2	0.75	**0.42**
4	0.69	**0.28**
8	0.64	**0.15**
16	0.60	**0.06**
32	0.54	**-0.01**

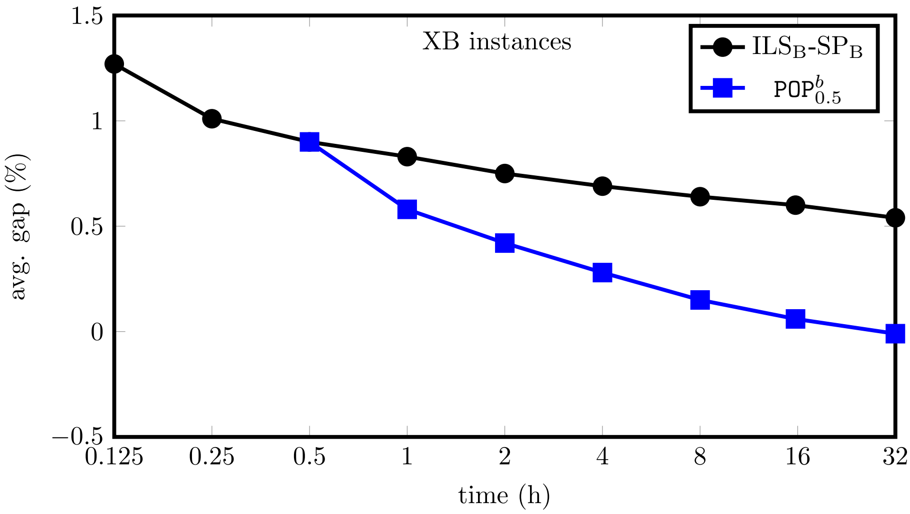


Figure 4.6: Convergence curves of ILS_B-SP_B and $\text{POP}^b_{0.5}$ for the XB instances of the VRPB.

4.5 Concluding remarks

In this chapter, we propose a POPMUSIC matheuristic for the classical and highly competitive CVRP. The algorithm is designed to improve a reasonably good initial solution given as an input. The results show that our approach outperforms one of the best published heuristics for the CVRP in medium and long runs. POP matheuristic is also competitive in long runs with a state-of-the-art heuristic, which is specialized to the CVRP. The results are especially good for instances with relatively short routes. Moreover, several best known solutions were improved for literature instances with up to 20,000 customers. This shows a very good scalability of the approach. The generality of the proposed POPMUSIC framework was shown through experiments with the HFVRP and VRPB, which are well-known extensions of the CVRP. Indeed, several new best solutions were found by POP for large instances based on the X benchmark, even using the same parameterization of the CVRP. These results revealed that POP is a promising approach to tackle similar problems by replacing the initial solution generator and the subproblem solver, along with possible small changes.

POP matheuristic exploits a characteristic of the modern exact algorithms for vehicle routing problems, and in particular, those for the CVRP. If a tight upper bound on the optimum value is provided, those exact algorithms are usually capable of solving to optimality medium-size instances with up to 100–150 customers in a few minutes. Instances with less than 100 customers are usually solved in seconds. Thus, exact (or heuristics such as BCP_H) approaches become competitive with the best heuristics for solving such instances. An important advantage of exact approaches is that they "know" when to stop after proving that an improving solution does not exist, whereas traditional metaheuristics do not possess that information.

POP has interesting features that may be explored in future works:

- It is very different from other existing and well-performing heuristics for the CVRP. This means that their strengths and weaknesses may be complementary. This opens possibilities for many types of hybridization. It could be something simple, like just determining the instance characteristics (besides route size) that make it more or less suited to POP, in order to decide which method should be applied. But it could be something deeper, a full integration where POP and some traditional heuristic could take turns on improving parts of a solution and exchange information. This seems to be quite a promising direction for research.
- It is easy to implement, provided that an exact code (and hence the heuristics derived from it) for solving the subproblems is at hand. Given that VRPSolver branch-cut-and-price algorithm is available for academic use and can solve many routing variants other than CVRP, it is natural to try algorithms similar POP on those variants. Of course, there is no guarantee that a straightforward adaptation will obtain good results. Thus, there is room for research on extensions of POP that are more suited for other particular routing problems.
- It is naturally parallelizable. The current sequential version of POP may take a few hours to obtain high-quality solutions and can not be used in practical situations that require faster solutions. That limitation could be much reduced by solving several subproblems in parallel, a natural feature of any POPMUSIC approach. In fact, the larger the instance, the more parallelizable the method becomes.

The last remark is that the underlying implementation of the BCP solver used in POP was not changed by us other than by modifying external parameters. Thus, there is a large potential to improve the efficiency of POP by "going inside the black box". A property that may be exploited is that the subproblems to be solved are often very similar to some already solved subproblem. Keeping information from previous runs, like the generated columns and cuts, may accelerate the algorithm.

Chapter 5

Concluding remarks

This thesis presented exact and heuristic approaches for different combinatorial optimization problems.

For the relaxed correlation clustering (RCC) problem—an NP-Hard problem with applications in social networks—, the two new integer linear programming formulations obtained a superior performance when compared to the existing one. Also, the developed heuristic based on iterated local search (ILS) substantially outperformed the previous ILS implementation from the literature.

For the vehicle routing problem with backhauls (VRPB)—a VRP problem with two types of customers —, the two depicted branch-cut-and-price (BCP) algorithms were able to find, for the first time, optimal solutions for all instances in the literature. Tests performed on a newly proposed set of instances showed that the more specialized BCP algorithm yielded better results than the other BCP implementation. Also, the three evaluated ILS-based heuristic strategies achieved extremely competitive results, and the more specialized method (the one using a customized set partitioning) was able to systematically find high quality solutions, especially for more challenging instances.

For the capacitated vehicle routing problem (CVRP), the proposed Partial OPtimization Metaheuristic Under Special Intensification Conditions (POPMUSIC) outperformed one of the best published metaheuristics in medium and long runs. The proposed POPMUSIC uses a generic state-of-the-art exact algorithm for the CVRP (and similar problems) to solve subproblems during its execution. Several best known solutions were improved for literature instances with up to 20,000 customers. Moreover, the method was successfully applied, after straightforward modifications, to the heterogeneous fleet vehicle routing problem (HFVRP) and VRPB.

As for future work, the following lines of research are suggested: (i) development of parallel heuristics and enhanced exact algorithms to solve larger RCC instances; (ii) extension of the proposed BCP algorithms to other VRPB variants considering, for example, multiple depots and mixed routes; (iii) hybridization of the proposed POPMUSIC with well-performing metaheuristics for CVRP and its extensions; (iv) parallelization of subproblem solving within the proposed POPMUSIC.

Bibliography

[1] ACCORSI, Luca and VIGO, Daniele. *A Fast and Scalable Heuristic for the Solution of Large-Scale Capacitated Vehicle Routing Problems*. Tech. rep. University of Bologna, 2020.

[2] ALES, Zacharie, KNIPPEL, Arnaud, and PAUCHET, Alexandre. "Polyhedral combinatorics of the K-partitioning problem with representative variables". In: *Discrete Applied Mathematics* 211 (2016), pp. 1–14.

[3] ALTAFINI, Claudio. "Dynamics of Opinion Forming in Structurally Balanced Social Networks". In: *PLOS ONE* 7.6 (June 2012), pp. 1–9.

[4] ARCHETTI, Claudia and SPERANZA, M. Grazia. "A survey on matheuristics for routing problems". In: *EURO Journal on Computational Optimization* 2.4 (Nov. 2014), pp. 223–246. ISSN: 2192-4414.

[5] ARINIK, Nejat, FIGUEIREDO, Rosa, and LABATUT, Vincent. "Signed Graph Analysis for the Interpretation of Voting Behavior". In: *International Conference on Knowledge Technologies and Data-driven Business (i-KNOW)*. International Workshop on Social Network Analysis and Digital Humanities (SnanDig). Graz, Austria, Oct. 2017.

[6] ARNOLD, Florian, GENDREAU, Michel, and SÖRENSEN, Kenneth. "Efficiently solving very large-scale routing problems". In: *Computers & Operations Research* 107 (2019), pp. 32–42. ISSN: 0305-0548. DOI: `https://doi.org/10.1016/j.cor.2019.03.006`.

[7] ARNOLD, Florian and SÖRENSEN, Kenneth. "Knowledge-guided local search for the vehicle routing problem". In: *Computers & Operations Research* 105 (2019), pp. 32–46. ISSN: 0305-0548. DOI: `https://doi.org/10.1016/j.cor.2019.01.002`.

[8] BAHIENSE, Laura et al. "A branch-and-cut algorithm for equitable coloring based on a formulation by representatives". In: *Electronic Notes in Discrete Mathematics* 35 (2009), pp. 347–352.

[9] BALDACCI, Roberto, CHRISTOFIDES, Nicos, and MINGOZZI, Aristide. "An exact algorithm for the vehicle routing problem based on the set partitioning formulation with additional cuts". In: *Mathematical Programming* 115.2 (Oct. 2008), pp. 351–385. ISSN: 1436-4646. DOI: 10.1007/s10107-007-0178-5.

[10] BALDACCI, Roberto, MINGOZZI, Aristide, and ROBERTI, Roberto. "New Route Relaxation and Pricing Strategies for the Vehicle Routing Problem". In: *Operations Research* 59.5 (2011), pp. 1269–1283.

[11] BANSAL, Nikhil, BLUM, Avrim, and CHAWLA, Shuchi. "Correlation Clustering". In: *Machine Learning* 56.1 (July 2004), pp. 89–113. ISSN: 1573-0565.

[12] BATTARRA, Maria, CORDEAU, Jean-François, and IORI, Manuel. "Pickup-and-Delivery Problems for Goods Transportation". In: *Vehicle Routing*. Ed. by Paolo TOTH and Danielle VIGO. 2014. Chap. 6, pp. 161–191.

[13] BEIER, Thorsten, HAMPRECHT, Fred A., and KAPPES, Jörg H. "Fusion Moves for Correlation Clustering". In: *CVPR. Proceedings*. 1. 2015, pp. 3507–3516.

[14] BELLOSO, Javier et al. "A biased-randomized metaheuristic for the vehicle routing problem with clustered and mixed backhauls". In: *Networks* 69.3 (2017), pp. 241–255.

[15] BONAMI, Pierre et al. "On the Solution of a Graph Partitioning Problem under Capacity Constraints". In: *Combinatorial Optimization*. Ed. by A. Ridha MAHJOUB et al. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 285–296. ISBN: 978-3-642-32147-4.

[16] BRANDÃO, José. "A deterministic iterated local search algorithm for the vehicle routing problem with backhauls". In: *TOP* 24.2 (July 2016), pp. 445–465. ISSN: 1863-8279.

[17] BRANDÃO, José. "A new tabu search algorithm for the vehicle routing problem with backhauls". In: *European Journal of Operational Research* 173.2 (2006), pp. 540–555. ISSN: 0377-2217.

[18] BRODER, Andrei Z. "The r-Stirling numbers". In: *Discrete Mathematics* 49.3 (1984), pp. 241–259. ISSN: 0012-365X.

[19] BRUSCO, Michael et al. "Two Algorithms for Relaxed Structural Balance Partitioning: Linking Theory, Models, and Data to Understand Social Network Phenomena". In: *Sociological Methods & Research* 40.1 (2011), pp. 57–87.

[20] BRUSCO, Michael J. and DOREIAN, Patrick. "Partitioning signed networks using relocation heuristics, tabu search, and variable neighborhood search". In: *Social Networks* 56 (2019), pp. 70–80. ISSN: 0378-8733.

[21] BULHÕES, Teobaldo et al. "Branch-and-cut approaches for p-cluster editing". In: *Discrete Applied Mathematics* 219 (2017), pp. 51–64.

[22] BULHÕES, Teobaldo et al. "On the complete set packing and set partitioning polytopes: Properties and rank 1 facets". In: *Operations Research Letters* 46.4 (2018), pp. 389–392. ISSN: 0167-6377. DOI: `https://doi.org/10.1016/j.orl.2018.04.006`.

[23] BULHÕES, Teobaldo et al. *VRPSolver: a Branch-Cut-and-Price based exact solver for vehicle routing and some related problems*. `https://vrpsolver.math.u-bordeaux.fr/`. Accessed: 2020-10-16. 2020.

[24] CAMPÊLO, Manoel, CAMPOS, Victor A., and CORREA, Ricardo C. "On the asymmetric representatives formulation for the vertex coloring problem". In: *Discrete Applied Mathematics* 156 (2008), pp. 1097–1111.

[25] CAMPÊLO, Manoel B., CORRÊA, Ricardo C., and FROTA, Yuri. "Cliques, holes and the vertex coloring polytope". In: *Inf. Process. Lett.* 89.4 (2004), pp. 159–164.

[26] CARTWRIGHT, Dorwin and HARARY, Frank. "Structural balance: a generalization of Heider's theory." In: *Psychological review* 63.5 (1956), pp. 277–293.

[27] CHRISTIAENS, Jan and VANDEN BERGHE, Greet. "Slack Induction by String Removals for Vehicle Routing Problems". In: *Transportation Science* 54.2 (2020), pp. 417–433.

[28] CLARKE, G. and WRIGHT, J. W. "Scheduling of Vehicles from a Central Depot to a Number of Delivery Points". In: *Operations Research* 12.4 (1964), pp. 568–581.

[29] CONTARDO, Claudio and MARTINELLI, Rafael. "A new exact algorithm for the multi-depot vehicle routing problem under capacity and route length constraints". In: *Discrete Optimization* 12 (2014), pp. 129–146. ISSN: 1572-5286. DOI: `https://doi.org/10.1016/j.disopt.2014.03.001`.

[30] CORDEAU, J-F et al. "A guide to vehicle routing heuristics". In: *J. Oper. Res. Soc.* 53.5 (2002), pp. 512–522. ISSN: 1476-9360.

[31] COSTA, Luciano, CONTARDO, Claudio, and DESAULNIERS, Guy. "Exact branch-price-and-cut algorithms for vehicle routing". In: *Transportation Science* 53.4 (2019), pp. 946–985.

[32] CUERVO, Daniel Palhazi et al. "An iterated local search algorithm for the vehicle routing problem with backhauls". In: *European Journal of Operational Research* 237.2 (2014), pp. 454–464. ISSN: 0377-2217.

[33] DAMBACHER, Jeffrey M., LI, Hiram W., and ROSSIGNOL, Philippe A. "Relevance of Community Structure in Assessing Indeterminacy of Ecological Predictions". In: *Ecology* 83.5 (2002), pp. 1372–1385. ISSN: 00129658, 19399170.

[34] DANTZIG, G. B. and RAMSER, J. H. "The Truck Dispatching Problem". In: *Management Science* 6.1 (1959), pp. 80–91.

[35] DASGUPTA, Bhaskar et al. "Algorithmic and complexity results for decompositions of biological networks into monotone subsystems". In: *BioSystems* 90 (2007), pp. 161–178.

[36] DAVIS, James A. "Clustering and Structural Balance in Graphs". In: *Human Relations* 20.2 (1967), pp. 181–187.

[37] DEIF, I and BODIN, L. "Extension of the Clarke and Wright algorithm for solving the vehicle routing problem with backhauling". In: *Proceedings of the Babson conference on software uses in transportation and logistics management*. Babson Park, MA. 1984, pp. 75–96.

[38] DOREIAN, Patrick. "A multiple indicator approach to blockmodeling signed networks". In: *Social Networks* 30.3 (2008), pp. 247–258. ISSN: 0378-8733.

[39] DOREIAN, Patrick and MRVAR, Andrej. "A partitioning approach to structural balance". In: *Social Networks* 18.2 (1996), pp. 149–168. ISSN: 0378-8733.

[40] DOREIAN, Patrick and MRVAR, Andrej. "Partitioning signed social networks". In: *Social Networks* 31.1 (2009), pp. 1–11. ISSN: 0378-8733.

[41] DOREIAN, Patrick and MRVAR, Andrej. "Structural Balance and Signed International Relations". In: *Journal of Social Structure* 16 (2015), p. 2.

[42] DOREIAN, Patrick and MRVAR, Andrej. "Testing Two Theories for Generating Signed Networks Using Real Data". In: 2014.

[43] DUNNING, I., HUCHETTE, J., and LUBIN, M. "JuMP: A Modeling Language for Mathematical Optimization". In: *SIAM Review* 59.2 (2017), pp. 295–320.

[44] FACCHETTI, Giuseppe, IACONO, Giovanni, and ALTAFINI, Claudio. "Computing global structural balance in large-scale signed social networks". In: *Proceedings of the National Academy of Sciences* 108.52 (2011), pp. 20953–20958. ISSN: 0027-8424.

[45] FAN, Neng and PARDALOS, Panos M. "Linear and quadratic programming approaches for the general graph partitioning problem". In: *Journal of Global Optimization* 48.1 (Sept. 2010), pp. 57–71. ISSN: 1573-2916.

[46] FIGUEIREDO, Rosa and FROTA, Yuri. "The maximum balanced subgraph of a signed graph: Applications and solution approaches". In: *European Journal of Operational Research* 236.2 (2014), pp. 473–487. ISSN: 0377-2217.

[47] FIGUEIREDO, Rosa, FROTA, Yuri, and LABBÉ, Martine. "A branch-and-cut algorithm for the maximum k-balanced subgraph of a signed graph". In: *Discrete Applied Mathematics* (2018). ISSN: 0166-218X.

[48] FIGUEIREDO, Rosa and MOURA, Gisele. "Mixed integer programming formulations for clustering problems related to structural balance". In: *Social Networks* 35.4 (2013), pp. 639–651. ISSN: 0378-8733.

[49] FISHER, Marshall L. and JAIKUMAR, Ramchandran. "A generalized assignment heuristic for vehicle routing". In: *Networks* 11.2 (1981), pp. 109–124.

[50] FROTA, Yuri et al. "A branch-and-cut algorithm for partition coloring". In: *Networks* 55 (2010), pp. 194–204.

[51] GAJPAL, Yuvraj and ABAD, P.L. "Multi-ant colony system (MACS) for a vehicle routing problem with backhauls". In: *European Journal of Operational Research* 196.1 (2009), pp. 102–117. ISSN: 0377-2217.

[52] GÉLINAS, Sylvie et al. "A new branching strategy for time constrained routing problems with application to backhauling". In: *Annals of Operations Research* 61.1 (Dec. 1995), pp. 91–109. ISSN: 1572-9338.

[53] GELOĞULLARI, Cumhur Alper. "An exact algorithm for the vehicle routing problem with backhauls". PhD thesis. bilkent university, 2001.

[54] GOETSCHALCKX, Marc and JACOBS-BLECHA, Charlotte. "The vehicle routing problem with backhauls". In: *European Journal of Operational Research* 42.1 (1989), pp. 39–51. ISSN: 0377-2217.

[55] GOLDEN, Bruce L. et al. "The Impact of Metaheuristics on Solving the Vehicle Routing Problem: Algorithms, Problem Sets, and Computational Results". In: *Fleet Management and Logistics*. Ed. by Teodor Gabriel CRAINIC and Gilbert LAPORTE. Boston, MA: Springer US, 1998, pp. 33–56. ISBN: 978-1-4615-5755-5. DOI: 10.1007/978-1-4615-5755-5_2.

[56] GRANADA-ECHEVERRI, M, TORO, E, and SANTA, J. “A mixed integer linear programming formulation for the vehicle routing problem with backhauls”. In: *International Journal of Industrial Engineering Computations* 10.2 (2019), pp. 295–308.

[57] HARARY, Frank, LIM, Meng-Hiot, and WUNSCH, Donald C. “Signed graphs for portfolio analysis in risk management”. In: *IMA Journal of Management Mathematics* 13 (2003), pp. 1–10.

[58] HEIDER, Fritz. “Attitudes and Cognitive Organization”. In: *The Journal of Psychology* 21.1 (1946). PMID: 21010780, pp. 107–112.

[59] HELSGAUN, Keld. *LKH-3 heuristic for solving constrained traveling salesman and vehicle routing problems.* 2020. URL: `http://akira.ruc.dk/~keld/research/LKH-3/` (visited on 03/26/2020).

[60] IRNICH, Stefan et al. “Path-reduced costs for eliminating arcs in routing and scheduling”. In: *INFORMS Journal on Computing* 22.2 (2010), pp. 297–313.

[61] JACOBS-BLECHA, Charlotte and GOETSCHALCKX, Marc. *The Vehicle Routing Problem with Backhauls: Properties and Solution Algorithms*. Tech. rep. Georgia Tech Research Corporation, 1992.

[62] JEPSEN, Mads et al. “Subset-row inequalities applied to the vehicle-routing problem with time windows”. In: *Operations Research* 56.2 (2008), pp. 497–511.

[63] JOURDAN, Laetitia, BASSEUR, Matthieu, and TALBI, E-G. “Hybridizing exact methods and metaheuristics: A taxonomy”. In: *European Journal of Operational Research* 199.3 (2009), pp. 620–629.

[64] KIM, Sungwoong et al. “Higher-Order Correlation Clustering for Image Segmentation”. In: *Advances in Neural Information Processing Systems 24*. Ed. by J. SHAWE-TAYLOR et al. Curran Associates, Inc., 2011, pp. 1530–1538.

[65] KIRKPATRICK, S., GELATT, C. D., and VECCHI, M. P. “Optimization by Simulated Annealing”. In: *Science* 220.4598 (1983), pp. 671–680. ISSN: 0036-8075. DOI: `10.1126/science.220.4598.671`.

[66] KOÇ, Çağrı and LAPORTE, Gilbert. “Vehicle routing with backhauls: Review and research perspectives”. In: *Computers & Operations Research* 91 (2018), pp. 79–91. ISSN: 0305-0548.

[67] LALLA-RUIZ, Eduardo and VOSS, Stefan. "A POPMUSIC approach for the Multi-Depot Cumulative Capacitated Vehicle Routing Problem". In: *Optimization Letters* 14.3 (Apr. 2020), pp. 671–691. ISSN: 1862-4480.

[68] LAPORTE, G. and NOBERT, Y. "A branch and bound algorithm for the capacitated vehicle routing problem". In: *Operations-Research-Spektrum* 5.2 (June 1983), pp. 77–85. ISSN: 1436-6304. DOI: 10.1007/BF01720015.

[69] LAPORTE, Gilbert, MERCURE, Hélène, and NOBERT, Yves. "An exact algorithm for the asymmetrical capacitated vehicle routing problem". In: *Networks* 16.1 (1986), pp. 33–46.

[70] LAPORTE, Gilbert, ROPKE, Stefan, and VIDAL, Thibaut. "Chapter 4: Heuristics for the Vehicle Routing Problem". In: *Vehicle Routing*. 2014, pp. 87–116. DOI: 10.1137/1.9781611973594.ch4.

[71] LEGGIERI, Valeria and HAOUARI, Mohamed. "A matheuristic for the asymmetric capacitated vehicle routing problem". In: *Discrete Applied Mathematics* 234 (2018). Special Issue on the Ninth International Colloquium on Graphs and Optimization (GO IX), 2014, pp. 139–150. ISSN: 0166-218X. DOI: https://doi.org/10.1016/j.dam.2016.03.019.

[72] LEMANN, Thomas B and SOLOMON, Richard L. "Group characteristics as revealed in sociometric patterns and personality ratings". In: *Sociometry* 15.1/2 (1952), pp. 7–90.

[73] LEVORATO, Mario. "Efficient solutions to the correlation clustering problem". Available at http://www.ic.uff.br/PosGraduacao/frontend-tesesdissertacoes/download.php?id=700.pdf&tipo=trabalho. MA thesis. Niterói, Rio de Janeiro, Brazil: Universidade Federal Fluminense, 2015.

[74] LEVORATO, Mario and FROTA, Yuri. "Brazilian Congress structural balance analysis". In: *Journal of Interdisciplinary Methodologies and Issues in Science* (2017).

[75] LEVORATO, Mario et al. "An ILS Algorithm to Evaluate Structural Balance in Signed Social Networks". In: *Proceedings of the 30th Annual ACM Symposium on Applied Computing*. SAC '15. Salamanca, Spain: ACM, 2015, pp. 1117–1122. ISBN: 978-1-4503-3196-8.

[76] LEVORATO, Mario et al. "Evaluating balancing on social networks through the efficient solution of correlation clustering problems". In: *EURO Journal on Computational Optimization* 5.4 (2017), pp. 467–498.

[77] LOURENÇO, Helena Ramalhinho, MARTIN, Olivier C, and STÜTZLE, Thomas. "Iterated local search: Framework and applications". In: *Handbook of metaheuristics*. Springer, 2019, pp. 129–168.

[78] LYSGAARD, Jens. *CVRPSEP: A package of separation routines for the Capacitated Vehicle Routing Problem*. English. Tech. Report. Aarhus University, Denmark, 2003.

[79] MAURYA, Mano Ram, RENGASWAMY, Raghunathan, and VENKATASUBRAMANIAN, Venkat. "Application of signed digraphs-based analysis for fault diagnosis of chemical process flowsheets". In: *Engineering Applications of Artificial Intelligence* 17.5 (2004), pp. 501–518. ISSN: 0952-1976.

[80] MÁXIMO, Vinícius R. and NASCIMENTO, Mariá C.V. "A hybrid adaptive iterated local search with diversification control to the capacitated vehicle routing problem". In: *European Journal of Operational Research* (2021). ISSN: 0377-2217.

[81] MCKINNEY, John C. "An Educational Application of a Two-Dimensional Sociometric Test". In: *Sociometry* 11.4 (1948), pp. 356–367. ISSN: 00380431.

[82] MINGOZZI, Aristide, GIORGI, Simone, and BALDACCI, Roberto. "An Exact Method for the Vehicle Routing Problem with Backhauls". In: *Transportation Science* 33.3 (1999), pp. 315–329.

[83] MLADENOVIĆ, Nenad and HANSEN, Pierre. "Variable neighborhood search". In: *Computers & operations research* 24.11 (1997), pp. 1097–1100.

[84] NEWCOMB, Theodore M. *The acquaintance process.* New York: Holt, Rinehart & Winston, 1961.

[85] OSMAN, Ibrahim H. and WASSAN, Niaz A. "A reactive tabu search meta-heuristic for the vehicle routing problem with back-hauls". In: *Journal of Scheduling* 5.4 (2002), pp. 263–285.

[86] OSTERTAG, A et al. "POPMUSIC for a real-world large-scale vehicle routing problem with time windows". In: *Journal of the Operational Research Society* 60.7 (2009), pp. 934–943.

[87] PECIN, Diego et al. "Improved branch-cut-and-price for capacitated vehicle routing". In: *Mathematical Programming Computation* 9.1 (Mar. 2017), pp. 61–100. ISSN: 1867-2957.

[88] PECIN, Diego et al. "Limited memory Rank-1 Cuts for Vehicle Routing Problems". In: *Operations Research Letters* 45.3 (2017), pp. 206–209. ISSN: 0167-6377. DOI: `https://doi.org/10.1016/j.orl.2017.02.006`.

[89] PENNA, Puca Huachi Vaz et al. "A hybrid heuristic for a broad class of vehicle routing problems with heterogeneous fleet". In: *Annals of Operations Research* 273.1 (Feb. 2019), pp. 5–74.

[90] PESSOA, Artur, SADYKOV, Ruslan, and UCHOA, Eduardo. "Enhanced Branch-Cut-and-Price algorithm for heterogeneous fleet vehicle routing problems". In: *European Journal of Operational Research* 270.2 (2018), pp. 530–543. ISSN: 0377-2217.

[91] PESSOA, Artur, UCHOA, Eduardo, and POGGI DE ARAGÃO, Marcus. "A robust branch-cut-and-price algorithm for the heterogeneous fleet vehicle routing problem". In: *Networks: An International Journal* 54.4 (2009), pp. 167–177.

[92] PESSOA, Artur et al. "A generic exact solver for vehicle routing and related problems". In: *Mathematical Programming* 183.1 (Oct. 2020), pp. 483–523. ISSN: 1436-4646.

[93] PESSOA, Artur et al. "Automation and combination of linear-programming based stabilization techniques in column generation". In: *INFORMS Journal on Computing* 30.2 (2018), pp. 339–360.

[94] POGGI, Marcus and UCHOA, Eduardo. "Chapter 3: New Exact Algorithms for the Capacitated Vehicle Routing Problem". In: *Vehicle routing: problems, methods, and applications*. Ed. by Paolo TOTH and Daniele VIGO. SIAM, 2014. Chap. 3, pp. 59–86.

[95] POGGI, Marcus and UCHOA, Eduardo. "Chapter 3: New exact algorithms for the capacitated vehicle routing problem". In: *Vehicle Routing: Problems, Methods, and Applications, Second Edition*. SIAM, 2014, pp. 59–86.

[96] QUEIROGA, Eduardo, SADYKOV, Ruslan, and UCHOA, Eduardo. *A modern POPMUSIC matheuristic for the capacitated vehicle routing problem*. Tech. rep. L-2020-2. Niterói, Brazil: Cadernos do LOGIS-UFF, Nov. 2020, p. 28.

[97] QUEIROGA, Eduardo et al. "Integer programming formulations and efficient local search for relaxed correlation clustering". In: *Journal of Global Optimization* (2021), pp. 1–48.

[98] QUEIROGA, Eduardo et al. "On the exact solution of vehicle routing problems with backhauls". In: *European Journal of Operational Research* 287.1 (2020), pp. 76–89.

[99] ROPKE, Stefan and PISINGER, David. "A unified heuristic for a large class of Vehicle Routing Problems with Backhauls". In: *European Journal of Operational Research* 171.3 (2006), pp. 750–775. ISSN: 0377-2217.

[100] SADYKOV, Ruslan, UCHOA, Eduardo, and PESSOA, Artur. "A Bucket Graph–Based Labeling Algorithm with Application to Vehicle Routing". In: *Transportation Science* 55.1 (2021), pp. 4–28. DOI: `10.1287/trsc.2020.0985`.

[101] SAMPSON, Samuel F. "A novitiate in a period of change: An experimental and case study of social relationships." PhD thesis. NY: Department of Sociology, Cornell University, 1968.

[102] SILVA, Marcos Melo, SUBRAMANIAN, Anand, and OCHI, Luiz Satoru. "An iterated local search heuristic for the split delivery vehicle routing problem". In: *Computers & Operations Research* 53 (2015), pp. 234–249. ISSN: 0305-0548.

[103] SILVA, Marcos Melo et al. "A simple and effective metaheuristic for the Minimum Latency Problem". In: *European Journal of Operational Research* 221.3 (2012), pp. 513–520. ISSN: 0377-2217.

[104] SUBRAMANIAN, Anand and FARIAS, Katyanne. "Efficient local search limitation strategy for single machine total weighted tardiness scheduling with sequence-dependent setup times". In: *Computers & Operations Research* 79 (2017), pp. 190–206. ISSN: 0305-0548.

[105] SUBRAMANIAN, Anand and QUEIROGA, Eduardo. "Solution strategies for the vehicle routing problem with backhauls". In: *Optimization Letters* (2020), pp. 1–13.

[106] SUBRAMANIAN, Anand, UCHOA, Eduardo, and OCHI, Luiz Satoru. "A hybrid algorithm for a class of vehicle routing problems". In: *Computers & Operations Research* 40.10 (2013), pp. 2519–2531. ISSN: 0305-0548. DOI: `https://doi.org/10.1016/j.cor.2013.01.013`.

[107] SUBRAMANIAN, Anand et al. "Branch-and-cut with lazy separation for the vehicle routing problem with simultaneous pickup and delivery". In: *Operations Research Letters* 39.5 (2011), pp. 338–341. ISSN: 0167-6377.

[108] SUBRAMANIAN, Anand et al. "Branch-cut-and-price for the vehicle routing problem with simultaneous pickup and delivery". In: *Optimization Letters* 7.7 (2013), pp. 1569–1581.

[109] TAILLARD, E. and HELSGAUN, K. "POPMUSIC for the travelling salesman problem". In: *European Journal of Operational Research* 272.2 (2019), pp. 420–429. ISSN: 0377-2217.

[110] TAILLARD, Éric D. and VOSS, Stefan. "Popmusic — Partial Optimization Metaheuristic under Special Intensification Conditions". In: *Essays and Surveys in Metaheuristics*. Boston, MA: Springer US, 2002, pp. 613–629. ISBN: 978-1-4615-1507-4.

[111] TARANTILIS, Christos D., ANAGNOSTOPOULOU, Afroditi K., and REPOUSSIS, Panagiotis P. "Adaptive Path Relinking for Vehicle Routing and Scheduling Problems with Product Returns". In: *Transportation Science* 47.3 (2013), pp. 356–379.

[112] TOTH, Paolo and VIGO, Daniele. "A Heuristic Algorithm for the Vehicle Routing Problem with Backhauls". In: *Advanced Methods in Transportation Analysis*. Ed. by Lucio BIANCO and Paolo TOTH. Berlin, Heidelberg: Springer Berlin Heidelberg, 1996, pp. 585–608. ISBN: 978-3-642-85256-5.

[113] TOTH, Paolo and VIGO, Daniele. "An Exact Algorithm for the Vehicle Routing Problem with Backhauls". In: *Transportation Science* 31.4 (1997), pp. 372–385.

[114] TÜTÜNCÜ, G. Yazgı. "An interactive GRAMPS algorithm for the heterogeneous fixed fleet vehicle routing problem with and without backhauls". In: *European Journal of Operational Research* 201.2 (2010), pp. 593–600. ISSN: 0377-2217.

[115] UCHOA, Eduardo et al. "New benchmark instances for the Capacitated Vehicle Routing Problem". In: *European Journal of Operational Research* 257.3 (2017), pp. 845–858. ISSN: 0377-2217.

[116] VAN GAEL, Jurgen and ZHU, Xiaojin. "Correlation Clustering for Crosslingual Link Detection". In: *Proceedings of the 20th International Joint Conference on Artifical Intelligence*. IJCAI'07. Hyderabad, India: Morgan Kaufmann Publishers Inc., 2007, pp. 1744–1749.

[117] VANDERBECK, F., SADYKOV, R., and TAHIRI, I. *BaPCod – a generic Branch-And-Price Code*. Available at `https://realopt.bordeaux.inria.fr/?page_id=2`. 2018.

[118] VASANTHI, B. et al. “Applications of Signed Graphs to Portfolio Turnover Analysis”. In: *Procedia - Social and Behavioral Sciences* 211 (2015). 2nd Global Conference on Business and Social Sciences (GCBSS-2015) on “Multidisciplinary Perspectives on Management and Society”, 17- 18 September, 2015, Bali, Indonesia, pp. 1203–1209. ISSN: 1877-0428.

[119] VIDAL, Thibaut. *Hybrid Genetic Search for the CVRP: Open-Source Implementation and SWAP* Neighborhood*. Tech. rep. 2020.

[120] VIDAL, Thibaut. *Personal communication*. 2019.

[121] VIDAL, Thibaut et al. “A Hybrid Genetic Algorithm for Multidepot and Periodic Vehicle Routing Problems”. In: *Operations Research* 60.3 (2012), pp. 611–624. DOI: 10.1287/opre.1120.1048.

[122] VIDAL, Thibaut et al. “A unified solution framework for multi-attribute vehicle routing problems”. In: *European Journal of Operational Research* 234.3 (2014), pp. 658–673. ISSN: 0377-2217.

[123] WANG, Ning and LI, Jie. “Restoring: A Greedy Heuristic Approach Based on Neighborhood for Correlation Clustering”. In: *Advanced Data Mining and Applications*. Ed. by Hiroshi MOTODA et al. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 348–359.

[124] WASSAN, N. “Reactive tabu adaptive memory programming search for the vehicle routing problem with backhauls”. In: *Journal of the Operational Research Society* 58.12 (2007), pp. 1630–1641.

[125] YANG, Bo, CHEUNG, William, and LIU, Jiming. “Community Mining from Signed Social Networks”. In: *IEEE Transactions on Knowledge and Data Engineering* 19 (2007), pp. 1333–1348.

[126] YANO, Candace Arai et al. “Vehicle Routing at Quality Stores”. In: *INFORMS Journal on Applied Analytics* 17.2 (1987), pp. 52–63.

[127] ZACHARIADIS, Emmanouil E. and KIRANOUDIS, Chris T. “An effective local search approach for the Vehicle Routing Problem with Backhauls”. In: *Expert Systems with Applications* 39.3 (2012), pp. 3174–3184. ISSN: 0957-4174.

[128] ZASLAVSKY, Thomas. “A mathematical bibliography of signed and gain graphs and allied areas”. In: *Electronic Journal of Combinatorics DS8* (1998).

[129] ZASLAVSKY, Thomas. “Signed graphs”. In: *Discrete Applied Mathematics* 4 (1982), pp. 47–74.

APPENDIX A – Updating the ADSs after an insertion move

The following pseudocodes describe in detail how to update the ADSs after performing an insertion move.

Algorithm `UpdateADSsAfterInsert`$(\boldsymbol{P}, \boldsymbol{S_p}, \boldsymbol{i}, \boldsymbol{S_q})$
- ▷ Note that, after the move, $i \in S_q$
- `CreateADSs`(P, S_q, i, S_p) ▷ Create the new ADSs indexed by i
- `DestroyADSsInsert`(P, S_p, i) ▷ Destroy old ADSs indexed by i
- `UpdateADSsSp`(P, S_p, i, S_q) ▷ Update ADSs related to S_p
- `UpdateADSsSq`(P, S_p, i, S_q) ▷ Update ADSs related to S_q
- `UpdateADSsOtherClusters`(P, S_p, i, S_q) ▷ Update ADSs related to other clusters

---

Algorithm `CreateADSs`$(\boldsymbol{P}, \boldsymbol{S_q}, \boldsymbol{i}, \boldsymbol{S_p})$
- $\texttt{SumIntra}^{+}[S_q][i][\leftarrow] = 0.0$
- $\texttt{SumIntra}^{-}[S_q][i][\leftarrow] = 0.0$
- $\texttt{SumIntra}^{+}[S_q][i][\rightarrow] = 0.0$
- $\texttt{SumIntra}^{-}[S_q][i][\rightarrow] = 0.0$
- ▷ transfer old values to the new ADSs
- **for** $S_r \in P \setminus \{S_p, S_q\}$ **do**
  - $\texttt{SumInter}^{+}[S_q][i][S_r][\leftarrow] = \texttt{SumInter}^{+}[S_p][i][S_r][\leftarrow]$
  - $\texttt{SumInter}^{-}[S_q][i][S_r][\leftarrow] = \texttt{SumInter}^{-}[S_p][i][S_r][\leftarrow]$
  - $\texttt{SumInter}^{+}[S_q][i][S_r][\rightarrow] = \texttt{SumInter}^{+}[S_p][i][S_r][\rightarrow]$
  - $\texttt{SumInter}^{-}[S_q][i][S_r][\rightarrow] = \texttt{SumInter}^{-}[S_p][i][S_r][\rightarrow]$
- $\texttt{SumInter}^{+}[S_q][i][S_p][\leftarrow] = 0.0$
- $\texttt{SumInter}^{-}[S_q][i][S_p][\leftarrow] = 0.0$
- $\texttt{SumInter}^{+}[S_q][i][S_p][\rightarrow] = 0.0$
- $\texttt{SumInter}^{-}[S_q][i][S_p][\rightarrow] = 0.0$

Algorithm `DestroyADSs`$(\boldsymbol{P}, \boldsymbol{S_p}, \boldsymbol{i})$
- Destroy $\texttt{SumIntra}^{+}[S_p][i][\leftarrow], \texttt{SumIntra}^{-}[S_p][i][\leftarrow], \texttt{SumIntra}^{+}[S_p][i][\rightarrow]$ and $\texttt{SumIntra}^{-}[S_p][i][\rightarrow]$
- **for** $S_r \in P \setminus \{S_p\}$ **do**
  - Destroy $\texttt{SumInter}^{+}[S_p][i][S_r][\leftarrow], \texttt{SumInter}^{-}[S_p][i][S_r][\leftarrow], \texttt{SumInter}^{+}[S_p][i][S_r][\rightarrow]$ and $\texttt{SumInter}^{-}[S_p][i][S_r][\rightarrow]$

Algorithm $\texttt{UpdateADSsSp}(\boldsymbol{P}, \boldsymbol{S_p}, \boldsymbol{i}, \boldsymbol{S_q})$

- **for** $j \in S_p$ **do**
  - **if** $(j,i) \in A^+$ **then**
    - $\texttt{SumIntra}^+[S_p][j][\rightarrow] = \texttt{SumIntra}^+[S_p][j][\rightarrow] - w_{ji}$
    - $\texttt{SumInter}^+[S_p][j][S_q][\rightarrow] = \texttt{SumInter}^+[S_p][j][S_q][\rightarrow] + w_{ji}$
    - $\texttt{SumInter}^+[S_q][i][S_p][\leftarrow] = \texttt{SumInter}^+[S_q][i][S_p][\leftarrow] + w_{ji}$
    - $\texttt{SumInter}^+[S_p][S_q] = \texttt{SumInter}^+[S_p][S_q] + w_{ji}$
    - $\texttt{SumIntra}^+[S_p] = \texttt{SumIntra}^+[S_p] - w_{ji}$
  - **else if** $(j,i) \in A^-$ **then**
    - $\texttt{SumIntra}^-[S_p][j][\rightarrow] = \texttt{SumIntra}^-[S_p][j][\rightarrow] - w_{ji}$
    - $\texttt{SumInter}^-[S_p][j][S_q][\rightarrow] = \texttt{SumInter}^-[S_p][j][S_q][\rightarrow] + w_{ji}$
    - $\texttt{SumInter}^-[S_q][i][S_p][\leftarrow] = \texttt{SumInter}^-[S_q][i][S_p][\leftarrow] + w_{ji}$
    - $\texttt{SumInter}^-[S_p][S_q] = \texttt{SumInter}^-[S_p][S_q] + w_{ji}$
    - $\texttt{SumIntra}^-[S_p] = \texttt{SumIntra}^-[S_p] - w_{ji}$
  - **if** $(i,j) \in A^+$ **then**
    - $\texttt{SumIntra}^+[S_p][j][\leftarrow] = \texttt{SumIntra}^+[S_p][j][\leftarrow] - w_{ij}$
    - $\texttt{SumInter}^+[S_p][j][S_q][\leftarrow] = \texttt{SumInter}^+[S_p][j][S_q][\leftarrow] + w_{ij}$
    - $\texttt{SumInter}^+[S_q][i][S_p][\rightarrow] = \texttt{SumInter}^+[S_q][i][S_p][\rightarrow] + w_{ij}$
    - $\texttt{SumInter}^+[S_q][S_p] = \texttt{SumInter}^+[S_q][S_p] + w_{ij}$
    - $\texttt{SumIntra}^+[S_p] = \texttt{SumIntra}^+[S_p] - w_{ij}$
  - **else if** $(i,j) \in A^-$ **then**
    - $\texttt{SumIntra}^-[S_p][j][\leftarrow] = \texttt{SumIntra}^-[S_p][j][\leftarrow] - w_{ij}$
    - $\texttt{SumInter}^-[S_p][j][S_q][\leftarrow] = \texttt{SumInter}^-[S_p][j][S_q][\leftarrow] + w_{ij}$
    - $\texttt{SumInter}^-[S_q][i][S_p][\rightarrow] = \texttt{SumInter}^-[S_q][i][S_p][\rightarrow] + w_{ij}$
    - $\texttt{SumInter}^-[S_q][S_p] = \texttt{SumInter}^-[S_q][S_p] + w_{ij}$
    - $\texttt{SumIntra}^-[S_p] = \texttt{SumIntra}^-[S_p] - w_{ij}$

Algorithm $\texttt{UpdateADSsSq}(\boldsymbol{P}, \boldsymbol{S_p}, \boldsymbol{i}, \boldsymbol{S_q})$

- **for** $j \in S_q$ **do**
  - **if** $(j,i) \in A^+$ **then**
    - $\texttt{SumIntra}^+[S_q][j][\rightarrow] = \texttt{SumIntra}^+[S_q][j][\rightarrow] + w_{ji}$
    - $\texttt{SumIntra}^+[S_q][i][\leftarrow] = \texttt{SumIntra}^+[S_q][i][\leftarrow] + w_{ji}$
    - $\texttt{SumInter}^+[S_q][j][S_p][\rightarrow] = \texttt{SumInter}^+[S_q][j][S_p][\rightarrow] - w_{ji}$
    - $\texttt{SumInter}^+[S_q][S_p] = \texttt{SumInter}^+[S_q][S_p] - w_{ji}$
    - $\texttt{SumIntra}^+[S_q] = \texttt{SumIntra}^+[S_q] + w_{ji}$
  - **else if** $(j,i) \in A^-$ **then**
    - $\texttt{SumIntra}^-[S_q][j][\rightarrow] = \texttt{SumIntra}^-[S_q][j][\rightarrow] + w_{ji}$
    - $\texttt{SumIntra}^-[S_q][i][\leftarrow] = \texttt{SumIntra}^-[S_q][i][\leftarrow] + w_{ji}$
    - $\texttt{SumInter}^-[S_q][j][S_p][\rightarrow] = \texttt{SumInter}^-[S_q][j][S_p][\rightarrow] - w_{ji}$
    - $\texttt{SumInter}^-[S_q][S_p] = \texttt{SumInter}^-[S_q][S_p] - w_{ji}$
    - $\texttt{SumIntra}^-[S_q] = \texttt{SumIntra}^-[S_q] + w_{ji}$
  - **if** $(i,j) \in A^+$ **then**
    - $\texttt{SumIntra}^+[S_q][j][\leftarrow] = \texttt{SumIntra}^+[S_q][j][\leftarrow] + w_{ij}$
    - $\texttt{SumIntra}^+[S_q][i][\rightarrow] = \texttt{SumIntra}^+[S_q][i][\rightarrow] + w_{ij}$
    - $\texttt{SumInter}^+[S_q][j][S_p][\leftarrow] = \texttt{SumInter}^+[S_q][j][S_p][\leftarrow] - w_{ij}$
    - $\texttt{SumInter}^+[S_p][S_q] = \texttt{SumInter}^+[S_p][S_q] - w_{ij}$
    - $\texttt{SumIntra}^+[S_q] = \texttt{SumIntra}^+[S_q] + w_{ij}$
  - **else if** $(i,j) \in A^-$ **then**
    - $\texttt{SumIntra}^-[S_q][j][\leftarrow] = \texttt{SumIntra}^-[S_q][j][\leftarrow] + w_{ij}$
    - $\texttt{SumIntra}^-[S_q][i][\rightarrow] = \texttt{SumIntra}^-[S_q][i][\rightarrow] + w_{ij}$
    - $\texttt{SumInter}^-[S_q][j][S_p][\leftarrow] = \texttt{SumInter}^-[S_q][j][S_p][\leftarrow] - w_{ij}$
    - $\texttt{SumInter}^-[S_p][S_q] = \texttt{SumInter}^-[S_p][S_q] - w_{ij}$
    - $\texttt{SumIntra}^-[S_q] = \texttt{SumIntra}^-[S_q] + w_{ij}$

Algorithm `UpdateADSsOtherClusters`$(\boldsymbol{P}, \boldsymbol{S_p}, \boldsymbol{i}, \boldsymbol{S_q})$

  for $S_r \in P \setminus \{S_p, S_q\}$ **do**

    for $j \in S_r$ **do**

      if $(j, i) \in A^+$ **then**

        $\texttt{SumInter}^+[S_r][j][S_p][\rightarrow] = \texttt{SumInter}^+[S_r][j][S_p][\rightarrow] - w_{ji}$

        $\texttt{SumInter}^+[S_r][j][S_q][\rightarrow] = \texttt{SumInter}^+[S_r][j][S_q][\rightarrow] + w_{ji}$

        $\texttt{SumInter}^+[S_r][S_p] = \texttt{SumInter}^+[S_r][S_p] - w_{ji}$

        $\texttt{SumInter}^+[S_r][S_q] = \texttt{SumInter}^+[S_r][S_q] + w_{ji}$

      else if $(j, i) \in A^-$ **then**

        $\texttt{SumInter}^-[S_r][j][S_p][\rightarrow] = \texttt{SumInter}^-[S_r][j][S_p][\rightarrow] - w_{ji}$

        $\texttt{SumInter}^-[S_r][j][S_q][\rightarrow] = \texttt{SumInter}^-[S_r][j][S_q][\rightarrow] + w_{ji}$

        $\texttt{SumInter}^-[S_r][S_p] = \texttt{SumInter}^-[S_r][S_p] - w_{ji}$

        $\texttt{SumInter}^-[S_r][S_q] = \texttt{SumInter}^-[S_r][S_q] + w_{ji}$

      if $(i, j) \in A^+$ **then**

        $\texttt{SumInter}^+[S_r][j][S_p][\leftarrow] = \texttt{SumInter}^+[S_r][j][S_p][\leftarrow] - w_{ij}$

        $\texttt{SumInter}^+[S_r][j][S_q][\leftarrow] = \texttt{SumInter}^+[S_r][j][S_q][\leftarrow] + w_{ij}$

        $\texttt{SumInter}^+[S_p][S_r] = \texttt{SumInter}^+[S_r][S_p] - w_{ij}$

        $\texttt{SumInter}^+[S_q][S_r] = \texttt{SumInter}^+[S_r][S_q] + w_{ij}$

      else if $(i, j) \in A^-$ **then**

        $\texttt{SumInter}^-[S_r][j][S_p][\leftarrow] = \texttt{SumInter}^-[S_r][j][S_p][\leftarrow] - w_{ij}$

        $\texttt{SumInter}^-[S_r][j][S_q][\leftarrow] = \texttt{SumInter}^-[S_r][j][S_q][\leftarrow] + w_{ij}$

        $\texttt{SumInter}^-[S_p][S_r] = \texttt{SumInter}^-[S_r][S_p] - w_{ij}$

        $\texttt{SumInter}^-[S_q][S_r] = \texttt{SumInter}^-[S_r][S_q] + w_{ij}$

APPENDIX B – Computing the cost for a swap move and updating the ADSs

The following pseudocodes describe in detail how to compute the cost of a swap move and how to update the ADSs after performing a move.

Algorithm $\texttt{CompCostSwap}(\boldsymbol{P}, \boldsymbol{RI_P}, \boldsymbol{S_p}, \boldsymbol{i}, \boldsymbol{S_q}, \boldsymbol{j})$

$\texttt{RemoveWeightsfromADSs}(S_p, i, S_q, j)$ ▷ Remove w_{ij} and w_{ji} from ADSs

$sum^+_{S_p} = \texttt{SumIntra}^+[S_p] - \texttt{SumIntra}^+[S_p][i][\leftarrow] - \texttt{SumIntra}^+[S_p][i][\rightarrow]$

$sum^-_{S_p} = \texttt{SumIntra}^-[S_p] - \texttt{SumIntra}^-[S_p][i][\leftarrow] - \texttt{SumIntra}^-[S_p][i][\rightarrow]$

$sum^+_{S_p} = sum^+_{S_p} + \texttt{SumInter}^+[S_q][j][S_p][\leftarrow] + \texttt{SumInter}^+[S_q][j][S_p][\rightarrow]$

$sum^-_{S_p} = sum^-_{S_p} + \texttt{SumInter}^-[S_q][j][S_p][\leftarrow] + \texttt{SumInter}^-[S_q][j][S_p][\rightarrow]$

$sum^+_{S_q} = \texttt{SumIntra}^+[S_q] - \texttt{SumIntra}^+[S_q][j][\leftarrow] - \texttt{SumIntra}^+[S_q][j][\rightarrow]$

$sum^-_{S_q} = \texttt{SumIntra}^-[S_q] - \texttt{SumIntra}^-[S_q][j][\leftarrow] - \texttt{SumIntra}^-[S_q][j][\rightarrow]$

$sum^+_{S_q} = sum^+_{S_q} + \texttt{SumInter}^+[S_p][i][S_q][\leftarrow] + \texttt{SumInter}^+[S_p][i][S_q][\rightarrow]$

$sum^-_{S_q} = sum^-_{S_q} + \texttt{SumInter}^-[S_p][i][S_q][\leftarrow] + \texttt{SumInter}^-[S_p][i][S_q][\rightarrow]$

$sum^+_{S_p,S_q} = \texttt{SumInter}^+[S_p][S_q] - \texttt{SumInter}^+[S_p][i][S_q][\rightarrow]$

$sum^+_{S_p,S_q} = sum^+_{S_p,S_q} - \texttt{SumInter}^+[S_q][j][S_p][\leftarrow]$

if $(i,j) \in A^+$ **then** $sum^+_{S_p,S_q} = sum^+_{S_p,S_q} - w_{ij}$

$sum^-_{S_p,S_q} = \texttt{SumInter}^-[S_p][S_q] - \texttt{SumInter}^-[S_p][i][S_q][\rightarrow]$

$sum^-_{S_p,S_q} = sum^-_{S_p,S_q} - \texttt{SumInter}^-[S_q][j][S_p][\leftarrow]$

if $(i,j) \in A^-$ **then** $sum^-_{S_p,S_q} = sum^-_{S_p,S_q} - w_{ij}$

$sum^+_{S_p,S_q} = sum^+_{S_p,S_q} + \texttt{SumIntra}^+[S_p][i][\leftarrow]$

$sum^-_{S_p,S_q} = sum^-_{S_p,S_q} + \texttt{SumIntra}^-[S_p][i][\leftarrow]$

$sum^+_{S_p,S_q} = sum^+_{S_p,S_q} + \texttt{SumIntra}^+[S_q][j][\rightarrow]$

$sum^-_{S_p,S_q} = sum^-_{S_p,S_q} + \texttt{SumIntra}^-[S_q][j][\rightarrow]$

$sum^+_{S_q,S_p} = \texttt{SumInter}^+[S_q][S_p] - \texttt{SumInter}^+[S_p][i][S_q][\leftarrow]$

$sum^+_{S_q,S_p} = sum^+_{S_q,S_p} - \texttt{SumInter}^+[S_q][j][S_p][\rightarrow]$

if $(j,i) \in A^+$ **then** $sum^+_{S_q,S_p} = sum^+_{S_q,S_p} - w_{ji}$

$sum^-_{S_q,S_p} = \texttt{SumInter}^-[S_q][S_p] - \texttt{SumInter}^-[S_p][i][S_q][\leftarrow]$

$sum^-_{S_q,S_p} = sum^-_{S_q,S_p} - \texttt{SumInter}^-[S_q][j][S_p][\rightarrow]$

if $(j,i) \in A^-$ **then** $sum^-_{S_q,S_p} = sum^-_{S_q,S_p} - w_{ji}$

$sum^+_{S_q,S_p} = sum^+_{S_q,S_p} + \texttt{SumIntra}^+[S_p][i][\rightarrow]$

$sum^-_{S_q,S_p} = sum^-_{S_q,S_p} + \texttt{SumIntra}^-[S_p][i][\rightarrow]$

$sum^+_{S_q,S_p} = sum^+_{S_q,S_p} + \texttt{SumIntra}^+[S_q][j][\leftarrow]$

$sum^-_{S_q,S_p} = sum^-_{S_q,S_p} + \texttt{SumIntra}^-[S_q][j][\leftarrow]$

$cost = \texttt{UpdateCost}(RI_P, S_p, S_p, sum^+_{S_p}, sum^-_{S_p})$

$cost = \texttt{UpdateCost}(cost, S_q, S_q, sum^+_{S_q}, sum^-_{S_q})$

$cost = \texttt{UpdateCost}(cost, S_p, S_q, sum^+_{S_p,S_q}, sum^-_{S_p,S_q})$

$cost = \texttt{UpdateCost}(cost, S_q, S_p, sum^+_{S_q,S_p}, sum^-_{S_q,S_p})$

$cost = \texttt{UpdateCostOtherClustersSwap}(cost, P, S_p, i, S_q, j)$

$\texttt{AddWeightsToADSs}(S_p, i, S_q, j)$ ▷ Add w_{ij} and w_{ji} to ADSs

return $cost$

Algorithm `RemoveWeightsfromADSs`$(\boldsymbol{S_p, i, S_q, j})$

  if $ij \in A^+$ **then**

    $\texttt{SumInter}^+[S_p][S_q] = \texttt{SumInter}^+[S_p][S_q] - w_{ij}$

    $\texttt{SumInter}^+[S_p][i][S_q][\rightarrow] = \texttt{SumInter}^+[S_p][i][S_q][\rightarrow] - w_{ij}$

    $\texttt{SumInter}^+[S_q][j][S_p][\leftarrow] = \texttt{SumInter}^+[S_q][j][S_p][\leftarrow] - w_{ij}$

  else if $ij \in A^-$ **then**

    $\texttt{SumInter}^-[S_p][S_q] = \texttt{SumInter}^-[S_p][S_q] - w_{ij}$

    $\texttt{SumInter}^-[S_p][i][S_q][\rightarrow] = \texttt{SumInter}^-[S_p][i][S_q][\rightarrow] - w_{ij}$

    $\texttt{SumInter}^-[S_q][j][S_p][\leftarrow] = \texttt{SumInter}^-[S_q][j][S_p][\leftarrow] - w_{ij}$

  if $ji \in A^+$ **then**

    $\texttt{SumInter}^+[S_q][S_p] = \texttt{SumInter}^+[S_q][S_p] - w_{ji}$

    $\texttt{SumInter}^+[S_q][j][S_p][\rightarrow] = \texttt{SumInter}^+[S_q][j][S_p][\rightarrow] - w_{ji}$

    $\texttt{SumInter}^+[S_p][i][S_q][\leftarrow] = \texttt{SumInter}^+[S_p][i][S_q][\leftarrow] - w_{ji}$

  else if $ji \in A^-$ **then**

    $\texttt{SumInter}^-[S_q][S_p] = \texttt{SumInter}^-[S_q][S_p] - w_{ji}$

    $\texttt{SumInter}^-[S_q][j][S_p][\rightarrow] = \texttt{SumInter}^-[S_q][j][S_p][\rightarrow] - w_{ji}$

    $\texttt{SumInter}^-[S_p][i][S_q][\leftarrow] = \texttt{SumInter}^-[S_p][i][S_q][\leftarrow] - w_{ji}$

Algorithm `UpdateCostOtherClustersSwap`$(\boldsymbol{cost, P, S_p, i, S_q, j})$

  for $S_r \in P \setminus \{S_p, S_q\}$ **do**

    $sum^+_{S_r,S_p} = \texttt{SumInter}^+[S_r][S_p] - \texttt{SumInter}^+[S_p][i][S_r][\leftarrow]$

    $sum^-_{S_r,S_p} = \texttt{SumInter}^-[S_r][S_p] - \texttt{SumInter}^-[S_p][i][S_r][\leftarrow]$

    $sum^+_{S_r,S_q} = \texttt{SumInter}^+[S_r][S_q] + \texttt{SumInter}^+[S_p][i][S_r][\leftarrow]$

    $sum^-_{S_r,S_q} = \texttt{SumInter}^-[S_r][S_q] + \texttt{SumInter}^-[S_p][i][S_r][\leftarrow]$

    $sum^+_{S_r,S_q} = sum^+_{S_r,S_q} - \texttt{SumInter}^+[S_q][j][S_r][\leftarrow]$

    $sum^-_{S_r,S_q} = sum^-_{S_r,S_q} - \texttt{SumInter}^-[S_q][j][S_r][\leftarrow]$

    $sum^+_{S_r,S_p} = sum^+_{S_r,S_p} + \texttt{SumInter}^+[S_q][j][S_r][\leftarrow]$

    $sum^-_{S_r,S_p} = sum^-_{S_r,S_p} + \texttt{SumInter}^-[S_q][j][S_r][\leftarrow]$

    $sum^+_{S_p,S_r} = \texttt{SumInter}^+[S_p][S_r] - \texttt{SumInter}^+[S_p][i][S_r][\rightarrow]$

    $sum^-_{S_p,S_r} = \texttt{SumInter}^-[S_p][S_r] - \texttt{SumInter}^-[S_p][i][S_r][\rightarrow]$

    $sum^+_{S_q,S_r} = \texttt{SumInter}^+[S_q][S_r] + \texttt{SumInter}^+[S_p][i][S_r][\rightarrow]$

    $sum^-_{S_q,S_r} = \texttt{SumInter}^-[S_q][S_r] + \texttt{SumInter}^-[S_p][i][S_r][\rightarrow]$

    $sum^+_{S_q,S_r} = sum^+_{S_q,S_r} - \texttt{SumInter}^+[S_q][j][S_r][\rightarrow]$

    $sum^-_{S_q,S_r} = sum^-_{S_q,S_r} - \texttt{SumInter}^-[S_q][j][S_r][\rightarrow]$

    $sum^+_{S_p,S_r} = sum^+_{S_p,S_r} + \texttt{SumInter}^+[S_q][j][S_r][\rightarrow]$

    $sum^-_{S_p,S_r} = sum^-_{S_q,S_r} + \texttt{SumInter}^-[S_q][j][S_r][\rightarrow]$

    $cost = \texttt{UpdateCost}(cost, S_r, S_p, sum^+_{S_r,S_p}, sum^-_{S_r,S_p})$

    $cost = \texttt{UpdateCost}(cost, S_r, S_q, sum^+_{S_r,S_q}, sum^-_{S_r,S_q})$

    $cost = \texttt{UpdateCost}(cost, S_p, S_r, sum^+_{S_p,S_r}, sum^-_{S_p,S_r})$

    $cost = \texttt{UpdateCost}(cost, S_q, S_r, sum^+_{S_q,S_r}, sum^-_{S_q,S_r})$

  return $cost$

Algorithm `AddWeightsToADSs`$(\boldsymbol{S_p, i, S_q, j})$

  if $(i,j) \in A^+$ **then**
    $\texttt{SumInter}^+[S_p][S_q] = \texttt{SumInter}^+[S_p][S_q] + w_{ij}$
    $\texttt{SumInter}^+[S_p][i][S_q][\rightarrow] = \texttt{SumInter}^+[S_p][i][S_q][\rightarrow] + w_{ij}$
    $\texttt{SumInter}^+[S_q][j][S_p][\leftarrow] = \texttt{SumInter}^+[S_q][j][S_p][\leftarrow] + w_{ij}$
  else if $(i,j) \in A^-$ **then**
    $\texttt{SumInter}^-[S_p][S_q] = \texttt{SumInter}^-[S_p][S_q] + w_{ij}$
    $\texttt{SumInter}^-[S_p][i][S_q][\rightarrow] = \texttt{SumInter}^-[S_p][i][S_q][\rightarrow] + w_{ij}$
    $\texttt{SumInter}^-[S_q][j][S_p][\leftarrow] = \texttt{SumInter}^-[S_q][j][S_p][\leftarrow] + w_{ij}$
  if $(j,i) \in A^+$ **then**
    $\texttt{SumInter}^+[S_q][S_p] = \texttt{SumInter}^+[S_q][S_p] + w_{ji}$
    $\texttt{SumInter}^+[S_q][j][S_p][\rightarrow] = \texttt{SumInter}^+[S_q][j][S_p][\rightarrow] + w_{ji}$
    $\texttt{SumInter}^+[S_p][i][S_q][\leftarrow] = \texttt{SumInter}^+[S_p][i][S_q][\leftarrow] + w_{ji}$
  else if $(j,i) \in A^-$ **then**
    $\texttt{SumInter}^-[S_q][S_p] = \texttt{SumInter}^-[S_q][S_p] + w_{ji}$
    $\texttt{SumInter}^-[S_q][j][S_p][\rightarrow] = \texttt{SumInter}^-[S_q][j][S_p][\rightarrow] + w_{ji}$
    $\texttt{SumInter}^-[S_p][i][S_q][\leftarrow] = \texttt{SumInter}^-[S_p][i][S_q][\leftarrow] + w_{ji}$

Algorithm `UpdateADSsAfterSwap`$(\boldsymbol{P, S_p, i, S_q, j})$

  ▷ Note that, after the move, $j \in S_p$ and $i \in S_q$
  `CreateADSs`(P, S_p, j, S_q) ▷ Create the new ADSs indexed by j
  `CreateADSs`(P, S_q, i, S_p) ▷ Create the new ADSs indexed by i
  `DestroyADSs`(P, S_p, i) ▷ Destroy old ADSs indexed by j
  `DestroyADSs`(P, S_q, j) ▷ Destroy old ADSs indexed by i
  `UpdateADSsSi`(P, S_p, i, S_q)
  ▷ Destroy old ADSs indexed by j
  `UpdateADSsSj`(P, S_p, i, S_q)
  `UpdateADSsSi`(P, S_q, j, S_p)
  `UpdateADSsSj`(P, S_q, j, S_p)
  ▷ Update ADSs due to arcs (i,j) and (j,i)
  if $(i,j) \in A^+$ **then**
    $\texttt{SumInter}^+[S_q][i][S_p][\rightarrow] = \texttt{SumInter}^+[S_q][i][S_p][\rightarrow] + w_{ij}$
    $\texttt{SumInter}^+[S_p][j][S_q][\leftarrow] = \texttt{SumInter}^+[S_p][j][S_q][\leftarrow] + w_{ij}$
    $\texttt{SumInter}^+[S_q][S_p] = \texttt{SumInter}^+[S_q][S_p] + w_{ij}$
    $\texttt{SumInter}^+[S_p][S_q] = \texttt{SumInter}^+[S_p][S_q] - w_{ij}$
  else if $(i,j) \in A^-$ **then**
    $\texttt{SumInter}^-[S_q][i][S_p][\rightarrow] = \texttt{SumInter}^-[S_q][i][S_p][\rightarrow] + w_{ij}$
    $\texttt{SumInter}^-[S_p][j][S_q][\leftarrow] = \texttt{SumInter}^-[S_p][j][S_q][\leftarrow] + w_{ij}$
    $\texttt{SumInter}^-[S_q][S_p] = \texttt{SumInter}^-[S_q][S_p] + w_{ij}$
    $\texttt{SumInter}^-[S_p][S_q] = \texttt{SumInter}^-[S_p][S_q] - w_{ij}$
  if $(j,i) \in A^+$ **then**
    $\texttt{SumInter}^+[S_p][j][S_q][\rightarrow] = \texttt{SumInter}^+[S_p][j][S_q][\rightarrow] + w_{ji}$
    $\texttt{SumInter}^+[S_q][i][S_p][\leftarrow] = \texttt{SumInter}^+[S_q][i][S_p][\leftarrow] + w_{ji}$
    $\texttt{SumInter}^+[S_p][S_q] = \texttt{SumInter}^+[S_p][S_q] + w_{ji}$
    $\texttt{SumInter}^+[S_q][S_p] = \texttt{SumInter}^+[S_q][S_p] - w_{ji}$
  else if $(j,i) \in A^-$ **then**
    $\texttt{SumInter}^-[S_p][j][S_q][\rightarrow] = \texttt{SumInter}^-[S_p][j][S_q][\rightarrow] + w_{ji}$
    $\texttt{SumInter}^-[S_q][i][S_p][\leftarrow] = \texttt{SumInter}^-[S_q][i][S_p][\leftarrow] + w_{ji}$
    $\texttt{SumInter}^-[S_p][S_q] = \texttt{SumInter}^-[S_p][S_q] + w_{ji}$
    $\texttt{SumInter}^-[S_q][S_p] = \texttt{SumInter}^-[S_q][S_p] - w_{ji}$
  `UpdateADSsOtherClusters`(P, S_p, i, S_q)
  `UpdateADSsOtherClusters`(P, S_q, j, S_p)

APPENDIX C – Best improvement algorithm for the split neighborhood

The following pseudocodes describe an efficient implementation of the *Split* neighborhood. The best improvement strategy makes use of local data structures to speedup the subsequent evaluations by taking advantage of the information obtained in the previous iteration. The efficient move evaluation is also presented using local and global ADSs.

Algorithm `BestSplitMove`$(\boldsymbol{P}, \boldsymbol{RI_P}, \boldsymbol{k})$
- **if** $|P| = k$ **then return** P
- Let S_b and c_b be a cluster and an index that represent the best split move
- $best_{imp} = 0$ ▷ `Improvement value obtained by applying the best split move`
- ▷ `For each cluster, evaluate all possible splits`
- **for** $S_p \in P$ **do**
  - **if** $|S_p| > 1$ **then**
    - ▷ `Creating local structures over` S'_p `and` S''_p `(resulting clusters) to speed up subsequent evaluations`
    - `LocalSumIntra`$^+[S'_p] = 0.0$ ▷ S'_p `is initially empty`
    - `LocalSumIntra`$^-[S'_p] = 0.0$
    - `LocalSumIntra`$^+[S''_p] =$ `SumIntra`$^+[S_p]$ ▷ S''_p `is initially` S_p
    - `LocalSumIntra`$^-[S''_p] =$ `SumIntra`$^-[S_p]$
    - ▷ `Sum of weights between clusters to the new ones`
    - **for** $S_r \in P \setminus S_p$ **do**
      - `LocalSumInter`$^+[S_r][S'_p] = 0.0$; `LocalSumInter`$^+[S'_p][S_r] = 0.0$
      - `LocalSumInter`$^-[S_r][S'_p] = 0.0$; `LocalSumInter`$^-[S'_p][S_r] = 0.0$
      - `LocalSumInter`$^+[S_r][S''_p] =$ `SumInter`$^+[S_r][S_p]$
      - `LocalSumInter`$^+[S''_p][S_r] =$ `SumInter`$^+[S_p][S_r]$
      - `LocalSumInter`$^-[S_r][S''_p] =$ `SumInter`$^-[S_r][S_p]$
      - `LocalSumInter`$^-[S''_p][S_r] =$ `SumInter`$^-[S_p][S_r]$
    - `LocalSumInter`$^+[S'_p][S''_p] = 0.0$
    - `LocalSumInter`$^-[S'_p][S''_p] = 0.0$
    - `LocalSumInter`$^+[S''_p][S'_p] = 0.0$
    - `LocalSumInter`$^-[S''_p][S'_p] = 0.0$
    - ▷ `Evaluate move and adjusts the local structures for the next one`
    - **for** $c \in \{0, 1, \ldots, |S_p| - 1\}$ **do**
      - $imp =$ `CompCostSplit`$(P, RI_P, S_p, c,$ `LocalSumIntra`, `LocalSumInter`$)$
      - **if** $imp > best_{imp}$ **then**
        - $S_b = S_p$
        - $c_b = c$
        - $best_{imp} = imp$
- Perform the split move (S_b, c_b) over P
- **return** P

Algorithm $\texttt{CompCostSplit}(\boldsymbol{P}, \boldsymbol{RI_P}, \boldsymbol{S_p}, \boldsymbol{c}, \texttt{LocalSumIntra}, \texttt{LocalSumInter})$

▷ Let $S_p = \{v_1, v_2, \ldots, v_{|S_p|}\}$, such as the resulting clusters are $S'_p = \{v_1, v_2, \ldots, v_c\}$ and $S''_p = \{v_{c+1}, \ldots, v_{|S_p|}\}$

$sum^+_{v_c,\leftarrow} = 0.0,\ sum^-_{v_c,\leftarrow} = 0.0, sum^+_{v_c,\rightarrow} = 0.0, sum^-_{v_c,\rightarrow} = 0.0$

▷ Sweep arcs between v_c and the S'_p of the previous move evaluation

for $v_i \in \{v_1, v_2, \ldots, v_{c-1}\}$ **do**

  if $(v_i, v_c) \in A^+$ **then**

    $\texttt{LocalSumIntra}^+[S'_p] = \texttt{LocalSumIntra}^+[S'_p] + w_{v_i,v_c}$

    $sum^+_{v_c,\leftarrow} = sum^+_{v_c,\leftarrow} + w_{v_i,v_c}$

  else if $(v_i, v_c) \in A^-$ **then**

    $\texttt{LocalSumIntra}^-[S'_p] = \texttt{LocalSumIntra}^-[S'_p] + w_{v_i,v_c}$

    $sum^-_{v_c,\leftarrow} = sum^-_{v_c,\leftarrow} + w_{v_i,v_c}$

  if $(v_c, v_i) \in A^+$ **then**

    $\texttt{LocalSumIntra}^+[S'_p] = \texttt{LocalSumIntra}^+[S'_p] + w_{v_c,v_i}$

    $sum^+_{v_c,\rightarrow} = sum^+_{v_c,\rightarrow} + w_{v_c,v_i}$

  else if $(v_c, v_i) \in A^-$ **then**

    $\texttt{LocalSumIntra}^-[S'_p] = \texttt{LocalSumIntra}^-[S'_p] + w_{v_c,v_i}$

    $sum^-_{v_c,\rightarrow} = sum^-_{v_c,\rightarrow} + w_{v_c,v_i}$

▷ Update LocalSumIntra for S''_p using global ADSs and auxiliary variables

$\texttt{LocalSumIntra}^+[S''_p] = \texttt{SumIntra}^+[S_p][\leftarrow] + \texttt{SumIntra}^+[S_p][\rightarrow] - sum^+_{v_c,\leftarrow} - sum^+_{v_c,\rightarrow}$

$\texttt{LocalSumIntra}^-[S''_p] = \texttt{SumIntra}^-[S_p][\leftarrow] + \texttt{SumIntra}^-[S_p][\rightarrow] - sum^-_{v_c,\leftarrow} - sum^-_{v_c,\rightarrow}$

▷ Update LocalSumInter using global ADSs and auxiliary variables

$sum^+_{S'_p,S''_p} = (\texttt{SumIntra}^+[S_p][v_c][\rightarrow] - sum^+_{v_c,\rightarrow}) - sum^+_{v_c,\leftarrow}$

$\texttt{LocalSumInter}^+[S'_p][S''_p] = \texttt{LocalSumInter}^+[S'_p][S''_p] + sum^+_{S'_p,S''_p}$

$sum^-_{S'_p,S''_p} = (\texttt{SumIntra}^-[S_p][v_c][\rightarrow] - sum^-_{v_c,\rightarrow}) - sum^-_{v_c,\leftarrow}$

$\texttt{LocalSumInter}^-[S'_p][S''_p] = \texttt{LocalSumInter}^-[S'_p][S''_p] + sum^-_{S'_p,S''_p}$

$sum^+_{S''_p,S'_p} = (\texttt{SumIntra}^+[S_p][v_c][\leftarrow] - sum^+_{v_c,\leftarrow}) - sum^+_{v_c,\rightarrow}$

$\texttt{LocalSumInter}^+[S''_p][S'_p] = \texttt{LocalSumInter}^+[S''_p][S'_p] + sum^+_{S''_p,S'_p}$

$sum^-_{S''_p,S'_p} = (\texttt{SumIntra}^-[S_p][v_c][\leftarrow] - sum^-_{v_c,\leftarrow}) - sum^-_{v_c,\rightarrow}$

$\texttt{LocalSumInter}^-[S''_p][S'_p] = \texttt{LocalSumInter}^-[S''_p][S'_p] + sum^-_{S''_p,S'_p}$

▷ Get relaxed imbalance and update the cost

$RI_{S_p} = \min\{\texttt{SumIntra}^+[S_p], \texttt{SumIntra}^-[S_p]\}$

$RI_{S'_p} = \min\{\texttt{LocalSumIntra}^+[S'_p], \texttt{LocalSumIntra}^-[S'_p]\}$

$RI_{S''_p} = \min\{\texttt{LocalSumIntra}^+[S''_p], \texttt{LocalSumIntra}^-[S''_p]\}$

$RI_{S'_p,S''_p} = \min\{\texttt{LocalSumInter}^+[S'_p][S''_p], \texttt{LocalSumInter}^-[S'_p][S''_p]\}$

$RI_{S''_p,S'_p} = \min\{\texttt{LocalSumInter}^+[S''_p][S'_p], \texttt{LocalSumInter}^-[S''_p][S'_p]\}$

$cost = RI_P - (RI_{S_p} - (RI_{S'_p} + RI_{S''_p} + RI_{S'_p,S''_p} + RI_{S''_p,S'_p}))$

$cost = \texttt{UpdateCostOtherClustersSplit}(cost, P, S_p, v_c, \texttt{LocalSumIntra}, \texttt{LocalSumInter})$

return $cost$

Algorithm `UpdateCostOtherClustersSplit(`$\boldsymbol{cost, P, S_p, v_c}$ `,LocalSumIntra, LocalSumInter)`

for $S_r \in P \setminus S_p$ **do**

$\texttt{LocalSumInter}^+[S_r][S'_p] = \texttt{LocalSumInter}^+[S_r][S'_p] + \texttt{SumInter}^+[S_p][v_c][S_r][\leftarrow]$

$\texttt{LocalSumInter}^-[S_r][S'_p] = \texttt{LocalSumInter}^-[S_r][S'_p] + \texttt{SumInter}^-[S_p][v_c][S_r][\leftarrow]$

$\texttt{LocalSumInter}^+[S'_p][S_r] = \texttt{LocalSumInter}^+[S'_p][S_r] + \texttt{SumInter}^+[S_p][v_c][S_r][\rightarrow]$

$\texttt{LocalSumInter}^-[S'_p][S_r] = \texttt{LocalSumInter}^-[S'_p][S_r] + \texttt{SumInter}^-[S_p][v_c][S_r][\rightarrow]$

$\texttt{LocalSumInter}^+[S_r][S''_p] = \texttt{LocalSumInter}^+[S_r][S''_p] - \texttt{SumInter}^+[S_p][v_c][S_r][\leftarrow]$

$\texttt{LocalSumInter}^-[S_r][S''_p] = \texttt{LocalSumInter}^-[S_r][S''_p] - \texttt{SumInter}^-[S_p][v_c][S_r][\leftarrow]$

$\texttt{LocalSumInter}^+[S''_p][S_r] = \texttt{LocalSumInter}^+[S''_p][S_r] - \texttt{SumInter}^+[S_p][v_c][S_r][\rightarrow]$

$\texttt{LocalSumInter}^-[S''_p][S_r] = \texttt{LocalSumInter}^-[S''_p][S_r] - \texttt{SumInter}^-[S_p][v_c][S_r][\rightarrow]$

$RI_{S_r,S_p} = \min\{\texttt{SumInter}^+[S_r][S_p], \texttt{SumInter}^-[S_r][S_p]\}$

$RI_{S_p,S_r} = \min\{\texttt{SumInter}^+[S_p][S_r], \texttt{SumInter}^-[S_p][S_r]\}$

$RI_{S_r,S'_p} = \min\{\texttt{LocalSumInter}^+[S_r][S'_p], \texttt{LocalSumInter}^-[S_r][S'_p]\}$

$RI_{S'_p,S_r} = \min\{\texttt{LocalSumInter}^+[S'_p][S_r], \texttt{LocalSumInter}^-[S'_p][S_r]\}$

$RI_{S_r,S''_p} = \min\{\texttt{LocalSumInter}^+[S_r][S''_p], \texttt{LocalSumInter}^-[S_r][S''_p]\}$

$RI_{S''_p,S_r} = \min\{\texttt{LocalSumInter}^+[S''_p][S_r], \texttt{LocalSumInter}^-[S''_p][S_r]\}$

$cost = cost - (RI_{S_r,S_p} + RI_{S_p,S_r} - (RI_{S_r,S'_p} + RI_{S'_p,S_r} + RI_{S_r,S''_p} + RI_{S''_p,S_r}))$

return $cost$

APPENDIX D – Detailed results for the random RCC instances

Table D.1: Relaxed imbalance obtained by ILS_{RCC} and $\text{ILS}_{\text{adapt}}$.

$\lvert V\rvert$	d	d^-	k	ILS_{RCC} min	avg	max	$\text{ILS}_{\text{adapt}}$ min	avg	max	t_{avg}
100	0.1	0.1	3	**131**	132.10	134	133	135.40	139	2.82
100	0.1	0.1	5	**94**	98.20	101	103	108.50	112	6.71
100	0.1	0.1	7	**68**	70.90	74	78	84.40	89	11.80
100	0.1	0.1	9	**45**	50.70	54	66	69.00	73	14.65
100	0.1	0.3	3	**408**	408.10	409	409	412.90	416	5.73
100	0.1	0.3	5	**323**	327.50	333	335	342.60	348	11.05
100	0.1	0.3	7	**270**	276.70	284	280	294.00	305	15.84
100	0.1	0.3	9	**233**	239.00	247	252	257.60	263	17.78
100	0.1	0.5	3	533	533.10	534	533	537.10	546	6.93
100	0.1	0.5	5	**427**	430.70	435	435	446.60	457	11.98
100	0.1	0.5	7	**352**	365.30	371	380	387.10	397	15.10
100	0.1	0.5	9	**298**	312.50	324	325	334.90	341	18.99
100	0.2	0.1	3	327	329.30	333	327	332.40	336	2.83
100	0.2	0.1	5	**284**	288.80	291	293	297.00	302	7.22
100	0.2	0.1	7	**253**	256.70	261	263	269.90	275	13.10
100	0.2	0.1	9	**227**	230.70	233	234	245.40	252	19.12
100	0.2	0.3	3	1032	1032.00	1032	1032	1033.00	1035	5.88
100	0.2	0.3	5	**913**	914.90	917	927	937.10	948	14.35
100	0.2	0.3	7	**833**	843.70	854	847	866.80	876	19.09
100	0.2	0.3	9	**775**	785.70	799	797	812.10	821	23.22
100	0.2	0.5	3	**1341**	1342.50	1344	1346	1349.40	1358	7.87
100	0.2	0.5	5	**1162**	1171.40	1177	1182	1194.90	1204	13.43
100	0.2	0.5	7	**1057**	1067.60	1075	1070	1092.30	1105	17.56
100	0.2	0.5	9	**955**	981.30	998	994	1013.10	1023	21.90
100	0.5	0.1	3	940	941.10	943	940	943.30	947	3.81
100	0.5	0.1	5	**899**	901.00	903	902	906.50	910	8.74
100	0.5	0.1	7	**864**	867.30	873	871	878.20	883	14.85
100	0.5	0.1	9	**833**	838.80	843	844	854.70	871	21.54
100	0.5	0.3	3	2741	2741.00	2741	2741	2741.10	2742	7.57
100	0.5	0.3	5	**2629**	2631.20	2634	2632	2636.70	2642	19.69
100	0.5	0.3	7	**2532**	2537.90	2556	2555	2568.80	2581	28.54
100	0.5	0.3	9	**2459**	2472.10	2487	2497	2508.60	2524	31.01
100	0.5	0.5	3	3939	3944.90	3958	3939	3949.80	3966	12.35
100	0.5	0.5	5	**3659**	3677.70	3693	3697	3713.10	3729	19.50
100	0.5	0.5	7	**3507**	3518.90	3529	3537	3548.70	3559	25.56

Table D.1 – *Continued from previous page*

$\lvert V \rvert$	d	d^-	k	ILS$_{\text{RCC}}$ min	avg	max	ILS$_{\text{adapt}}$ min	avg	max	t$_{\text{avg}}$
100	0.5	0.5	9	**3343**	3367.90	3382	3399	3417.10	3435	26.51
100	0.8	0.1	3	1537	1538.00	1541	1537	1540.40	1546	2.82
100	0.8	0.1	5	**1502**	1502.00	1502	1503	1509.50	1515	6.08
100	0.8	0.1	7	**1470**	1471.30	1474	1473	1484.00	1493	10.41
100	0.8	0.1	9	**1439**	1443.10	1445	1448	1456.40	1465	16.35
100	0.8	0.3	3	4599	4601.50	4606	4599	4600.60	4604	7.17
100	0.8	0.3	5	**4462**	4468.90	4476	4468	4478.60	4488	18.14
100	0.8	0.3	7	**4364**	4377.30	4385	4369	4391.50	4403	30.49
100	0.8	0.3	9	**4270**	4286.10	4308	4303	4316.60	4332	36.91
100	0.8	0.5	3	6676	6676.00	6676	6676	6695.60	6741	16.14
100	0.8	0.5	5	**6319**	6330.80	6347	6343	6376.80	6401	23.40
100	0.8	0.5	7	**6095**	6115.60	6141	6136	6166.90	6195	28.12
100	0.8	0.5	9	**5921**	5939.60	5959	5973	5989.00	6008	31.76
200	0.1	0.1	3	**691**	694.60	700	695	699.60	703	16.03
200	0.1	0.1	5	**602**	606.70	611	615	624.30	634	42.44
200	0.1	0.1	7	**539**	545.20	550	565	576.10	582	85.12
200	0.1	0.1	9	**499**	504.30	511	524	535.90	544	121.75
200	0.1	0.3	3	**1993**	1993.50	1995	2002	2011.90	2024	32.91
200	0.1	0.3	5	**1782**	1799.30	1812	1823	1834.90	1848	90.46
200	0.1	0.3	7	**1659**	1680.70	1695	1712	1719.70	1730	123.25
200	0.1	0.3	9	**1574**	1589.70	1607	1619	1633.60	1645	142.54
200	0.1	0.5	3	**2716**	2726.60	2744	2732	2744.20	2754	56.06
200	0.1	0.5	5	**2403**	2414.90	2435	2435	2465.90	2484	81.47
200	0.1	0.5	7	**2226**	2240.50	2260	2274	2299.50	2327	108.14
200	0.1	0.5	9	**2104**	2122.70	2135	2151	2173.70	2189	119.36
200	0.2	0.1	3	**1515**	1517.30	1526	1516	1520.70	1524	15.82
200	0.2	0.1	5	**1422**	1433.80	1445	1437	1452.50	1461	43.79
200	0.2	0.1	7	**1357**	1364.40	1371	1384	1394.30	1410	88.66
200	0.2	0.1	9	**1293**	1311.50	1326	1328	1350.10	1357	145.51
200	0.2	0.3	3	**4376**	4383.10	4387	4377	4384.10	4392	32.41
200	0.2	0.3	5	**4136**	4144.30	4154	4161	4176.80	4194	102.09
200	0.2	0.3	7	**3957**	3971.30	3988	4027	4042.70	4054	163.24
200	0.2	0.3	9	**3856**	3871.90	3885	3916	3938.90	3960	168.21
200	0.2	0.5	3	**6142**	6153.80	6178	6172	6187.30	6206	66.70
200	0.2	0.5	5	**5689**	5715.00	5729	5740	5772.30	5818	92.31
200	0.2	0.5	7	**5436**	5460.90	5501	5449	5528.80	5558	114.75
200	0.2	0.5	9	**5229**	5264.30	5283	5298	5336.40	5362	138.09
200	0.5	0.1	3	**3950**	3951.50	3957	3952	3958.90	3963	20.22
200	0.5	0.1	5	**3876**	3887.10	3899	3893	3903.90	3912	43.74
200	0.5	0.1	7	**3814**	3826.40	3835	3835	3855.60	3869	77.27
200	0.5	0.1	9	**3769**	3774.70	3782	3802	3811.20	3824	125.47
200	0.5	0.3	3	11624	11629.30	11639	11624	11632.30	11643	43.78
200	0.5	0.3	5	**11368**	11384.70	11398	11398	11416.20	11436	112.73
200	0.5	0.3	7	**11169**	11193.00	11217	11227	11254.90	11273	213.97
200	0.5	0.3	9	**11052**	11066.70	11089	11122	11142.10	11175	251.60
200	0.5	0.5	3	**17090**	17098.40	17124	17121	17147.80	17192	104.95
200	0.5	0.5	5	**16367**	16401.30	16462	16434	16496.40	16540	136.04
200	0.5	0.5	7	**15969**	16006.10	16038	16054	16099.80	16137	157.54
200	0.5	0.5	9	**15621**	15673.30	15710	15734	15787.00	15821	169.82
200	0.8	0.1	3	6348	6348.80	6350	6348	6355.50	6366	14.59
200	0.8	0.1	5	**6284**	6291.10	6295	6297	6310.60	6322	29.45

Table D.1 – *Continued from previous page*

$\lvert V\rvert$	d	d^-	k	$\mathrm{ILS}_{\mathrm{RCC}}$ min	$\mathrm{ILS}_{\mathrm{RCC}}$ avg	$\mathrm{ILS}_{\mathrm{RCC}}$ max	$\mathrm{ILS}_{\mathrm{adapt}}$ min	$\mathrm{ILS}_{\mathrm{adapt}}$ avg	$\mathrm{ILS}_{\mathrm{adapt}}$ max	t_{avg}
200	0.8	0.1	7	**6226**	6237.90	6247	6243	6263.90	6279	52.70
200	0.8	0.1	9	**6181**	6189.70	6202	6199	6214.10	6234	80.28
200	0.8	0.3	3	**18810**	18811.80	18816	18817	18823.60	18831	41.21
200	0.8	0.3	5	**18546**	18556.70	18569	18573	18588.90	18612	116.73
200	0.8	0.3	7	**18339**	18374.70	18400	18381	18415.50	18450	197.81
200	0.8	0.3	9	**18148**	18211.30	18241	18261	18291.50	18319	291.05
200	0.8	0.5	3	**28291**	28301.70	28311	28305	28364.70	28410	129.39
200	0.8	0.5	5	**27342**	27400.10	27458	27497	27544.10	27589	157.90
200	0.8	0.5	7	**26793**	26867.10	26911	26921	27000.30	27078	180.17
200	0.8	0.5	9	**26416**	26474.00	26534	26521	26591.40	26676	193.61
400	0.1	0.1	3	2989	2996.30	3006	**2986**	2996.10	3003	104.42
400	0.1	0.1	5	**2812**	2826.60	2839	2813	2833.70	2853	284.54
400	0.1	0.1	7	**2669**	2688.60	2711	2687	2708.50	2756	625.85
400	0.1	0.1	9	**2569**	2586.10	2605	2578	2608.00	2633	1020.60
400	0.1	0.3	3	8843	8851.50	8862	**8837**	8845.30	8854	233.79
400	0.1	0.3	5	**8360**	8374.40	8383	8389	8413.10	8475	811.53
400	0.1	0.3	7	**8022**	8047.40	8074	8040	8119.80	8206	1280.75
400	0.1	0.3	9	**7835**	7880.20	7934	7893	7950.30	7990	1153.92
400	0.1	0.5	3	12523	12532.80	12544	**12522**	12562.00	12610	508.94
400	0.1	0.5	5	**11611**	11648.60	11679	11622	11734.00	11783	660.74
400	0.1	0.5	7	**11099**	11178.80	11209	11179	11243.30	11309	815.20
400	0.1	0.5	9	**10807**	10868.90	10928	10869	10929.00	10985	812.93
400	0.2	0.1	3	6296	6300.10	6306	6296	6303.30	6315	91.57
400	0.2	0.1	5	**6147**	6159.20	6170	6157	6167.90	6186	247.20
400	0.2	0.1	7	**6028**	6039.90	6071	6035	6054.00	6085	462.77
400	0.2	0.1	9	**5912**	5923.90	5934	5927	5950.60	5975	775.45
400	0.2	0.3	3	18278	18284.70	18298	**18277**	18291.60	18305	235.19
400	0.2	0.3	5	**17735**	17758.10	17779	17764	17802.00	17849	786.29
400	0.2	0.3	7	**17385**	17431.00	17467	17509	17555.30	17597	1370.41
400	0.2	0.3	9	**17163**	17207.10	17236	17244	17321.80	17373	1560.29
400	0.2	0.5	3	26898	26921.80	26941	**26892**	26960.90	27041	567.47
400	0.2	0.5	5	**25577**	25655.80	25711	25707	25774.00	25805	749.25
400	0.2	0.5	7	**24947**	24994.40	25055	25016	25108.90	25199	846.28
400	0.2	0.5	9	**24472**	24550.60	24590	24530	24639.40	24727	950.62
400	0.5	0.1	3	**15853**	15860.50	15867	15865	15875.60	15885	100.59
400	0.5	0.1	5	**15758**	15767.10	15777	15775	15786.70	15811	208.04
400	0.5	0.1	7	**15642**	15668.50	15684	15665	15689.90	15717	361.93
400	0.5	0.1	9	**15561**	15580.80	15610	15602	15625.00	15641	580.47
400	0.5	0.3	3	**47486**	47500.60	47521	47488	47516.40	47536	228.52
400	0.5	0.3	5	47004	47020.20	47038	**46996**	47035.40	47074	666.00
400	0.5	0.3	7	**46604**	46636.50	46677	46632	46682.40	46759	1295.69
400	0.5	0.3	9	**46319**	46352.50	46381	46353	46429.70	46474	1722.73
400	0.5	0.5	3	**71795**	71852.80	71898	71858	71969.60	72029	955.35
400	0.5	0.5	5	**69825**	69888.40	69958	69973	70148.90	70274	1105.49
400	0.5	0.5	7	**68747**	68873.20	68939	68945	69082.80	69205	1116.31
400	0.5	0.5	9	**68093**	68167.30	68262	68103	68239.80	68522	1195.26
400	0.8	0.1	3	**25285**	25289.20	25293	25295	25302.50	25310	68.99
400	0.8	0.1	5	**25192**	25203.60	25210	25205	25220.40	25242	134.38
400	0.8	0.1	7	**25086**	25106.80	25124	25095	25135.10	25170	209.22
400	0.8	0.1	9	**25027**	25038.30	25051	25054	25064.40	25081	321.54
400	0.8	0.3	3	**76029**	76044.80	76075	76039	76084.70	76157	214.40

Table D.1 – *Continued from previous page*

$\|V\|$	d	d^-	k	ILS$_{\text{RCC}}$ min	ILS$_{\text{RCC}}$ avg	ILS$_{\text{RCC}}$ max	ILS$_{\text{adapt}}$ min	ILS$_{\text{adapt}}$ avg	ILS$_{\text{adapt}}$ max	t$_{\text{avg}}$
400	0.8	0.3	5	**75601**	75627.60	75674	75605	75673.40	75748	547.72
400	0.8	0.3	7	**75228**	75280.40	75332	75237	75315.60	75396	1099.86
400	0.8	0.3	9	**74910**	74954.10	74982	74960	75032.90	75096	1662.96
400	0.8	0.5	3	**117476**	117543.50	117626	117527	117631.00	117773	1109.08
400	0.8	0.5	5	**114831**	114990.90	115126	115077	115261.10	115489	1294.47
400	0.8	0.5	7	**113596**	113681.80	113790	113660	113938.50	114126	1407.46
400	0.8	0.5	9	**112546**	112725.80	112904	112836	112996.20	113288	1406.61
600	0.1	0.1	3	6842	6854.10	6861	**6833**	6847.00	6858	317.03
600	0.1	0.1	5	6621	6632.10	6643	**6614**	6642.60	6678	736.46
600	0.1	0.1	7	**6405**	6438.80	6474	6438	6464.60	6501	1656.70
600	0.1	0.1	9	**6251**	6283.30	6298	6291	6316.40	6359	2645.73
600	0.1	0.3	3	**20349**	20382.00	20398	20356	20385.00	20412	783.90
600	0.1	0.3	5	**19655**	19682.20	19706	19712	19755.70	19788	2518.19
600	0.1	0.3	7	**19168**	19201.50	19228	19238	19324.00	19372	4519.86
600	0.1	0.3	9	**18775**	18840.30	18875	18913	19016.70	19078	4870.33
600	0.1	0.5	3	**29305**	29349.80	29379	29379	29451.00	29516	2034.70
600	0.1	0.5	5	**27696**	27765.20	27866	27923	27959.40	28017	2364.64
600	0.1	0.5	7	**26880**	26956.00	27029	27007	27129.90	27196	2621.14
600	0.1	0.5	9	**26244**	26374.90	26442	26352	26501.40	26600	2860.93
600	0.2	0.1	3	14166	14171.30	14180	**14154**	14172.20	14178	262.04
600	0.2	0.1	5	13976	13996.50	14030	**13969**	14007.40	14053	626.76
600	0.2	0.1	7	**13812**	13830.10	13862	13830	13866.80	13894	1196.27
600	0.2	0.1	9	**13662**	13683.90	13707	13674	13711.30	13758	2054.11
600	0.2	0.3	3	**42106**	42129.40	42160	42115	42149.60	42189	751.15
600	0.2	0.3	5	**41340**	41395.50	41442	41425	41464.50	41531	2294.29
600	0.2	0.3	7	**40801**	40856.90	40912	40951	41016.00	41092	4377.53
600	0.2	0.3	9	**40406**	40450.90	40490	40584	40716.60	40825	5831.76
600	0.2	0.5	3	**62367**	62455.70	62532	62487	62603.90	62704	2301.21
600	0.2	0.5	5	**60041**	60203.50	60337	60316	60483.90	60597	2740.62
600	0.2	0.5	7	**58939**	59089.90	59212	59165	59296.90	59363	3095.99
600	0.2	0.5	9	**58124**	58245.80	58399	58270	58433.80	58576	3159.25
600	0.5	0.1	3	**35888**	35897.40	35910	35893	35911.10	35920	265.25
600	0.5	0.1	5	**35748**	35764.60	35779	35784	35799.60	35814	574.68
600	0.5	0.1	7	**35614**	35649.50	35677	35659	35678.20	35701	874.83
600	0.5	0.1	9	**35493**	35527.30	35551	35543	35580.60	35608	1415.54
600	0.5	0.3	3	106944	106962.10	106982	**106940**	106989.50	107042	664.15
600	0.5	0.3	5	**106282**	106328.70	106368	106297	106375.70	106493	1943.99
600	0.5	0.3	7	**105701**	105777.80	105879	105767	105879.30	106002	3598.85
600	0.5	0.3	9	**105256**	105341.80	105417	105298	105424.20	105562	5084.31
600	0.5	0.5	3	**165124**	165197.50	165260	165337	165464.60	165612	3580.68
600	0.5	0.5	5	**161478**	161564.70	161669	161907	162056.50	162160	3874.33
600	0.5	0.5	7	**159537**	159707.10	159876	159824	160116.70	160347	3929.42
600	0.5	0.5	9	**158301**	158384.50	158506	158513	158763.00	159063	4083.18
600	0.8	0.1	3	57334	57346.90	57357	**57333**	57352.90	57369	166.32
600	0.8	0.1	5	**57208**	57225.60	57248	57231	57242.50	57251	352.95
600	0.8	0.1	7	57044	57075.50	57106	57044	57099.50	57179	524.58
600	0.8	0.1	9	**56968**	56990.30	57006	56971	57003.20	57061	852.47
600	0.8	0.3	3	**172024**	172049.90	172084	172064	172094.00	172142	575.18
600	0.8	0.3	5	171515	171559.10	171605	**171459**	171590.80	171695	1375.56
600	0.8	0.3	7	**170996**	171088.70	171165	171044	171165.60	171259	2608.47
600	0.8	0.3	9	**170559**	170648.70	170738	170651	170742.30	170905	4124.47

Table D.1 – *Continued from previous page*

$\lvert V \rvert$	d	d^{-}	k	$\mathrm{ILS}_{\mathrm{RCC}}$			$\mathrm{ILS}_{\mathrm{adapt}}$			$\mathrm{t}_{\mathrm{avg}}$
				min	avg	max	min	avg	max	
600	0.8	0.5	3	**269158**	269304.00	269465	269353	269646.40	269850	4247.83
600	0.8	0.5	5	**264210**	264560.20	264822	265003	265276.20	265560	4722.79
600	0.8	0.5	7	**261978**	262158.00	262388	262582	262903.30	263149	4998.81
600	0.8	0.5	9	**260310**	260584.20	260937	260654	260983.80	261283	4755.96

APPENDIX E – Detailed results for the SRCC instances

Table E.1: Symmetric relaxed imbalance obtained by ILS_{RCC} and ILS Levorato et al. [76].

Instance	$\|V\|$	d	d^-	k	ILS_{RCC}			ILS Levorato et al. [76]			t_{avg}
					min	avg	max	min	avg	max	
UNGA-1946	54	0.484	0.27	2	9.338	9.338	9.338	9.338	9.338	9.338	0.5
UNGA-1947	57	0.490	0.42	3	18.698	18.697	18.698	18.698	18.697	18.698	0.7
UNGA-1948	59	0.494	0.34	4	4.399	4.399	4.399	4.399	4.399	4.399	1.2
UNGA-1949	59	0.496	0.28	2	37.748	37.748	37.748	37.748	37.748	37.748	0.6
UNGA-1950	60	0.496	0.25	2	25.028	25.028	25.028	25.028	25.028	25.028	0.6
UNGA-1951	60	0.490	0.37	2	58.960	58.960	58.960	58.960	58.960	58.960	0.8
UNGA-1952	60	0.495	0.26	2	46.099	46.099	46.099	46.099	46.099	46.099	0.6
UNGA-1953	60	0.488	0.34	2	31.288	31.288	31.288	31.288	31.288	31.288	0.8
UNGA-1954	60	0.492	0.30	2	32.823	32.823	32.823	32.823	32.823	32.823	0.8
UNGA-1955	65	0.464	0.11	4	5.377	5.377	5.377	5.377	5.377	5.377	1.5
UNGA-1956	81	0.480	0.30	4	17.181	17.181	17.181	17.181	17.181	17.181	2.3
UNGA-1957	82	0.495	0.32	3	37.512	37.512	37.512	37.512	37.512	37.512	2.0
UNGA-1958	82	0.489	0.25	2	122.536	122.536	122.536	122.536	122.536	122.536	1.2
UNGA-1959	82	0.497	0.35	2	102.881	102.881	102.881	102.881	102.881	102.881	2.2
UNGA-1960	100	0.488	0.39	3	45.464	45.464	45.464	45.464	45.464	45.464	3.4
UNGA-1961	106	0.467	0.35	3	37.395	37.395	37.395	37.395	37.395	37.395	3.9
UNGA-1962	110	0.468	0.33	2	154.412	154.412	154.412	154.412	154.412	154.412	4.4
UNGA-1963	113	0.490	0.18	4	20.639	20.639	20.639	20.639	20.639	20.639	3.7
UNGA-1964[1]	115	0.500	0.28	3	0.000	0.000	0.000	0.000	31.200	39.000	1.0
UNGA-1965	117	0.495	0.21	4	29.482	29.482	29.482	29.482	29.482	29.482	6.3
UNGA-1966	122	0.484	0.23	2	213.680	213.680	213.680	213.680	213.680	213.680	2.3
UNGA-1967	124	0.490	0.29	4	42.298	42.298	42.298	42.298	42.298	42.298	7.4
UNGA-1968	126	0.490	0.25	3	86.239	86.239	86.239	86.239	86.239	86.239	6.1
UNGA-1969	126	0.495	0.21	3	66.277	66.277	66.277	66.277	66.277	66.277	4.9
UNGA-1970	127	0.497	0.21	3	69.316	69.316	69.316	69.316	69.316	69.316	4.9
UNGA-1971	133	0.493	0.09	4	19.306	19.306	19.306	19.306	19.306	19.306	4.8
UNGA-1972	132	0.499	0.04	2	16.294	16.294	16.294	16.294	16.294	16.294	1.4
UNGA-1973	135	0.499	0.09	3	14.142	14.142	14.142	14.142	14.142	14.142	2.5
UNGA-1974	138	0.499	0.10	3	18.608	18.608	18.608	18.608	18.608	18.608	3.1
UNGA-1975	143	0.472	0.21	4	53.707	53.707	53.707	53.707	53.707	53.707	6.6
UNGA-1976	144	0.460	0.16	4	34.606	34.606	34.606	34.606	34.606	34.606	5.6
UNGA-1977	146	0.465	0.09	6	15.548	15.548	15.548	15.548	15.548	15.548	12.7
UNGA-1978	148	0.483	0.14	3	**74.755**	74.755	74.755	75.445	76.629	77.812	3.1

[1] In the 19th session, voting occurred on only one resolution which explains the signed digraph with very low relaxed imbalance.

Table E.1 – *Continued from previous page*

Instance	$\|V\|$	d	d^-	k	ILS$_{\text{RCC}}$ min	ILS$_{\text{RCC}}$ avg	ILS$_{\text{RCC}}$ max	ILS Levorato et al. [76] min	ILS Levorato et al. [76] avg	ILS Levorato et al. [76] max	t$_{\text{avg}}$
UNGA-1979	150	0.470	0.16	5	21.520	21.520	21.520	21.520	21.520	21.520	9.6
UNGA-1980	151	0.474	0.18	6	29.303	29.303	29.303	29.303	30.054	31.807	11.7
UNGA-1981	155	0.477	0.18	5	29.121	29.121	29.121	29.121	29.121	29.121	11.3
UNGA-1982	156	0.432	0.15	4	31.378	31.378	31.378	31.378	31.378	31.378	9.8
UNGA-1983	157	0.474	0.22	4	29.523	29.523	29.523	29.523	36.168	62.750	7.4
UNGA-1984	158	0.439	0.20	4	14.033	14.033	14.033	14.033	17.197	45.679	6.9
UNGA-1985	158	0.431	0.12	2	53.630	53.630	53.630	53.630	53.630	53.630	2.9
UNGA-1986	158	0.499	0.08	2	44.673	44.673	44.673	44.673	44.673	44.673	2.4
UNGA-1987	158	0.499	0.05	3	9.119	9.119	9.119	9.119	9.119	9.119	2.8
UNGA-1988	158	0.499	0.08	2	33.905	33.905	33.905	33.905	33.905	33.905	2.5
UNGA-1989	158	0.500	0.05	2	17.302	17.302	17.302	17.302	17.302	17.302	2.3
UNGA-1990	158	0.498	0.10	3	15.024	15.024	15.024	15.024	15.024	15.024	3.5
UNGA-1991	178	0.467	0.10	3	15.480	15.480	15.480	15.480	15.569	15.692	3.6
UNGA-1992	180	0.493	0.08	4	12.201	12.201	12.201	12.201	12.770	17.889	8.0
UNGA-1993	184	0.496	0.09	3	24.689	24.972	25.003	24.689	24.689	24.689	4.4
UNGA-1994	185	0.497	0.13	3	23.573	23.573	23.573	23.573	23.573	23.573	5.5
UNGA-1995	185	0.489	0.11	3	27.624	27.624	27.624	27.624	28.248	28.663	5.1
UNGA-1996	185	0.499	0.07	3	9.541	9.541	9.541	9.541	9.541	9.541	5.6
UNGA-1997	176	0.471	0.16	5	27.018	27.018	27.018	27.018	27.018	27.018	13.0
UNGA-1998	177	0.492	0.14	4	39.300	39.300	39.300	39.300	39.300	39.300	11.2
UNGA-1999	182	0.487	0.10	4	14.375	14.375	14.375	14.375	14.386	14.412	9.0
UNGA-2000	189	0.495	0.13	4	25.099	25.100	25.099	25.099	25.100	25.099	8.6
UNGA-2001	191	0.495	0.16	2	33.531	33.531	33.531	33.531	33.531	33.531	6.8
UNGA-2002	192	0.495	0.10	3	12.687	12.687	12.687	12.687	12.687	12.687	4.1
UNGA-2003	191	0.489	0.06	2	7.466	7.466	7.466	7.466	7.466	7.466	4.2
UNGA-2004	191	0.498	0.05	2	20.638	20.638	20.638	20.638	20.638	20.638	3.6
UNGA-2005	192	0.482	0.06	3	25.516	25.516	25.516	25.516	25.516	25.516	6.0
UNGA-2006	192	0.498	0.05	2	28.954	28.955	28.954	28.954	28.955	28.954	3.6
UNGA-2007	192	0.498	0.06	2	45.570	45.570	45.570	45.570	45.570	45.570	3.7
UNGA-2008	192	0.495	0.06	2	36.889	36.889	36.889	36.889	36.889	36.889	3.8
Slashdot1	200	0.022	0.07	5	**11.0**	11.700	12.0	16.0	17.30	19.0	16.9
Slashdot2	300	0.012	0.08	8	**4.0**	4.600	5.0	8.0	11.10	12.0	61.1
Slashdot3	400	0.008	0.07	4	**15.0**	16.100	18.0	20.0	22.20	24.0	67.7
Slashdot4	600	0.005	0.08	9	**8.0**	10.700	13.0	16.0	19.20	21.0	438.5
Slashdot5	800	0.005	0.11	20	**14.0**	16.300	20.0	32.0	35.60	41.0	2161.5
Slashdot6	1000	0.006	0.14	11	**182.0**	187.200	194.0	206.0	211.10	218.0	6107.6
Slashdot7	2000	0.005	0.15	43	**626.0**	650.700	686.0	712.0	751.90	777.0	7200.0
BR-2010-v1	545	0.490	0.01	4	**309.692**	309.692	309.692	316.091	375.860	596.137	135.3
BR-2010-v2	545	0.490	0.02	4	**332.493**	332.493	332.493	338.836	343.849	345.102	160.9
BR-2011-v1	553	0.488	0.20	4	562.540	562.541	562.541	562.540	584.250	589.677	534.2
BR-2011-v2	553	0.486	0.21	4	566.950	566.950	566.950	566.950	594.462	601.993	517.5
BR-2012-v1	555	0.489	0.04	4	**658.613**	658.613	658.613	672.986	755.450	1072.030	202.8
BR-2012-v2	555	0.488	0.04	4	**681.168**	681.168	681.168	697.567	703.277	704.705	202.2
BR-2013-v1	540	0.489	0.04	4	483.511	503.611	514.011	483.511	757.230	1266.900	117.1
BR-2013-v2	540	0.489	0.04	4	**504.918**	522.200	539.912	631.341	692.390	732.912	115.6
BR-2014-v1	556	0.496	0.01	4	**130.020**	131.001	131.968	131.973	136.706	168.493	61.3
BR-2014-v2	556	0.495	0.01	4	**137.165**	137.629	138.326	140.349	148.078	182.796	65.4
BR-2015-v1	552	0.486	0.12	4	536.724	536.724	536.724	536.724	676.785	1147.860	271.7

Table E.1 – *Continued from previous page*

Instance	$\|V\|$	d	d^-	k	ILS_{RCC}			ILS Levorato et al. [76]			t_{avg}
					min	avg	max	min	avg	max	
BR-2015-v2	552	0.484	0.18	4	585.038	585.038	585.038	585.038	707.996	830.955	294.7
BR-2016-v1	544	0.484	0.10	4	1241.520	1241.520	1241.520	1241.520	1411.645	1610.620	291.9
BR-2016-v2	544	0.482	0.11	4	**1285.950**	1285.950	1285.950	1377.020	1511.614	1677.130	287.7

APPENDIX F – Detailed results of the BCP algorithms for the X instances

Table F.1: Results for X instances by the BCP$_{\mathcal{F}2}$ with a time limit of 60 hours. The results which were already reported in Table 3.5 were omitted. The final lower bound is denoted by LB_f.

Instance	UB	$time_{ub}$ (h)	LB_f	$z(IP)$	LB^f_{root}	$time$ (s)	$time_{prc}$ (s)	$nodes$
X-n172-50-k27	30,634	4.1	30,634	**30,634**	30,535	270	41	19
X-n172-66-k31	31,864	3.6	31,864	**31,864**	31,808	161	20	7
X-n172-80-k39	36,803	3.2	36,803	**36,803**	36,746	560	37	15
X-n176-50-k23	45,239	4.4	45,239	**45,239**	45,162	437	60	37
X-n176-66-k24	46,416	4.3	46,416	**46,416**	46,337	736	122	65
X-n176-80-k25	47,033	6.1	47,033	**47,033**	46,987	341	61	11
X-n181-50-k12	16,549	5.6	16,549	**16,549**	16,549	57	15	1
X-n181-66-k15	18,832	4.9	18,832	**18,832**	18,832	41	14	1
X-n181-80-k18	21,241	4.4	21,241	**21,241**	21,241	54	15	1
X-n186-50-k8	17,978	4.3	17,978	**17,978**	17,868	9,088	1,721	41
X-n186-66-k10	19,751	4.1	19,751	**19,751**	19,751	302	118	1
X-n186-80-k12	21,754	5.3	21,754	**21,754**	21,631	21,953	9,826	113
X-n190-50-k4	11,552	7.6	11,552	**11,552**	11,493	9,096	4,850	29
X-n190-66-k5	12,784	14.1	12,738	–	12,719	–	177,686	231
X-n190-80-k6	14,410	11.4	14,345	–	14,340	–	191,111	237
X-n195-50-k27	29,470	4.0	29,470	**29,470**	29,376	698	46	25
X-n195-66-k34	33,137	4.4	33,137	**33,137**	33,077	166	20	7
X-n195-80-k42	38,629	5.2	38,629	**38,629**	38,629	186	27	1
X-n200-50-k18	34,416	11.8	34,408	**34,408**	34,291	81,125	4,131	1,269
X-n200-66-k24	40,474	12.6	40,342	–	40,321	–	21,542	4,435
X-n200-80-k29	47,741	8.9	47,741	**47,741**	47,714	585	76	5
X-n204-50-k10	15,877	5.0	15,877	**15,877**	15,841	1,349	269	7
X-n204-66-k12	16,703	5.1	16,703	**16,703**	16,564	24,018	4,998	179

(*Continues on the next page*)

Instance	UB	$time_{ub}$ (h)	LB_f	$z(IP)$	LB^f_{root}	$time$ (s)	$time_{prc}$ (s)	$nodes$
X-n204-80-k15	17,832	5.3	17,832	**17,832**	17,791	1,394	411	5
X-n209-50-k8	21,837	7.3	21,837	**21,837**	21,649	52,058	6,891	205
X-n209-66-k11	24,378	8.8	24,378	**24,378**	24,209	75,385	38,246	327
X-n209-80-k13	27,177	11.2	27,001	–	26,983	–	115,022	661
X-n214-50-k6	9,574	7.1	9,574	**9,574**	9,550	3,422	1,118	9
X-n214-66-k8	10,001	7.4	10,001	**10,001**	9,964	2,356	1,490	9
X-n214-80-k9	10,457	6.8	10,457	**10,457**	10,406	18,255	10,668	63
X-n219-50-k37	64,691	10.8	64,691	**64,691**	64,620	50	10	3
X-n219-66-k48	80,405	7.9	80,405	**80,405**	80,405	40	14	1
X-n219-80-k59	95,845	7.5	95,845	**95,845**	95,845	36	16	1
X-n223-50-k18	27,449	10.0	27,442	**27,442**	27,327	11,835	1,069	139
X-n223-66-k23	30,717	8.4	30,717	**30,717**	30,568	35,662	3,976	407
X-n223-80-k27	34,440	8.1	34,440	**34,440**	34,336	15,291	1,752	161
X-n228-50-k19	23,128	7.1	23,128	**23,128**	23,079	1,022	327	23
X-n228-66-k20	24,114	10.2	24,113	**24,113**	24,052	5,511	2,057	39
X-n228-80-k21	24,592	6.4	24,592	**24,592**	24,592	724	319	1
X-n233-50-k10	17,186	7.1	17,186	**17,186**	17,053	93,406	71,427	159
X-n233-66-k12	18,026	5.7	18,026	**18,026**	17,966	2,328	1,394	9
X-n233-80-k14	18,885	6.4	18,662	–	18,642	–	174,564	793
X-n237-50-k7	20,745	10.1	20,745	**20,745**	20,676	23,322	13,874	33
X-n237-66-k9	22,471	13.0	22,471	**22,471**	22,380	83,074	66,372	105
X-n237-80-k11	24,357	12.0	24,357	**24,357**	24,308	2,593	1,530	7
X-n242-50-k25	47,949	18.3	47,722	–	47,672	–	19,577	2,899
X-n242-66-k32	57,197	14.0	57,197	**57,197**	57,044	84,515	9,241	1,559
X-n242-80-k39	68,969	19.4	68,965	**68,965**	68,828	63,924	8,640	895
X-n247-50-k42	36,701	10.0	36,701	**36,701**	36,701	76	42	1
X-n247-66-k43	36,994	10.6	36,994	**36,994**	36,994	84	41	1
X-n247-80-k45	37,220	11.7	37,205	**37,205**	37,200	293	130	3
X-n251-50-k14	24,968	20.3	24,968	**24,968**	24,876	25,557	3,127	127
X-n251-66-k18	27,817	13.6	27,817	**27,817**	27,713	72,575	8,566	355
X-n251-80-k22	32,170	15.5	32,027	–	32,007	–	50,325	965
X-n256-50-k8	15,922	6.9	15,922	**15,922**	15,922	404	189	1
X-n256-66-k11	17,250	7.4	17,250	**17,250**	17,250	521	290	1
X-n256-80-k13	18,189	8.7	18,073	–	18,041	–	142,662	819
X-n261-50-k7	21,555	13.0	21,555	**21,555**	21,457	63,665	40,112	37
X-n261-66-k9	23,065	12.2	22,885	–	22,856	–	187,116	61
X-n261-80-k11	25,128	16.5	24,913	–	24,867	–	180,864	281
X-n266-50-k30	47,815	35.1	47,783	**47,783**	47,649	101,191	13,933	1,461
X-n266-66-k39	55,962	24.4	55,794	55,945	55,782	–	34,684	3,563
X-n266-80-k47	63,880	25.5	63,780	–	63,731	–	24,218	3,147

(*Continues on the next page*)

Instance	UB	$time_{ub}$ (h)	LB_f	$z(IP)$	LB^f_{root}	$time$ (s)	$time_{prc}$ (s)	$nodes$
X-n270-50-k18	24,776	14.0	24,751	**24,751**	24,654	45,333	6,267	237
X-n270-66-k24	26,377	7.5	26,377	**26,377**	26,329	1,941	272	21
X-n270-80-k29	29,789	7.9	29,692	–	29,659	–	30,397	1,377
X-n275-50-k14	15,561	13.6	15,561	**15,561**	15,515	5,294	832	21
X-n275-66-k19	16,944	10.7	16,944	**16,944**	16,930	513	113	3
X-n275-80-k22	18,690	13.0	18,688	**18,688**	18,659	3,719	681	29
X-n280-50-k13	29,132	11.4	29,132	**29,132**	29,005	128,359	93,994	71
X-n280-66-k15	31,315	19.7	31,138	–	31,111	–	158,023	407
X-n280-80-k16	32,332	15.3	32,030	–	32,013	–	192,361	147
X-n284-50-k8	15,944	20.6	15,944	**15,944**	15,834	93,999	43,893	209
X-n284-66-k10	17,277	16.3	17,226	–	17,196	–	190,019	21
X-n284-80-k12	18,830	19.5	18,693	–	18,676	–	184,435	127
X-n289-50-k34	57,957	34.0	57,573	–	57,530	–	33,748	1,775
X-n289-66-k38	63,446	26.9	63,207	–	63,187	–	42,435	2,423
X-n289-80-k47	75,963	33.8	75,645	–	75,628	–	40,762	1,739
X-n294-50-k26	30,859	9.1	30,859	**30,859**	30,747	4,905	468	45
X-n294-66-k33	34,636	11.5	34,636	**34,636**	34,543	12,903	1,056	143
X-n294-80-k40	39,269	11.5	39,096	–	39,077	–	22,714	1,951
X-n298-50-k16	25,081	8.5	25,081	**25,081**	24,959	35,865	3,878	173
X-n298-66-k21	27,643	15.8	27,521	–	27,471	–	54,914	899
X-n298-80-k25	30,222	16.2	30,222	**30,222**	30,108	85,792	22,231	305
X-n303-50-k11	17,763	14.5	17,669	–	17,647	–	128,405	101
X-n303-66-k13	18,120	8.9	18,120	**18,120**	18,048	54,891	34,827	91
X-n303-80-k16	19,603	12.0	19,480	–	19,457	–	172,877	249
X-n308-50-k9	22,544	15.6	22,319	–	22,305	–	194,203	5
X-n308-66-k11	24,154	20.5	24,001	–	23,991	–	194,915	29
X-n308-80-k12	25,164	22.4	24,860	–	24,845	–	209,695	13
X-n313-50-k39	57,762	30.7	57,476	–	57,445	–	41,082	2,765
X-n313-66-k44	60,089	31.8	59,936	60,069	59,915	–	28,091	2,901
X-n313-80-k56	73,834	28.0	73,672	–	73,655	–	37,558	2,661
X-n317-50-k27	43,396	60.8	43,391	**43,391**	43,368	1,290	236	17
X-n317-66-k35	54,502	40.9	54,502	**54,502**	54,486	1,018	233	15
X-n317-80-k43	63,683	24.6	63,683	**63,683**	63,666	626	250	9
X-n322-50-k14	23,309	11.9	23,309	**23,309**	23,140	132,954	23,529	349
X-n322-66-k19	25,034	10.7	25,034	**25,034**	24,952	6,100	1,703	21
X-n322-80-k23	27,500	14.7	27,500	**27,500**	27,376	184,907	63,083	453
X-n327-50-k10	21,610	25.2	21,379	–	21,347	–	107,246	145
X-n327-66-k13	23,322	17.7	23,197	–	23,186	–	159,555	69
X-n327-80-k16	24,990	21.4	24,751	–	24,729	–	186,367	221
X-n331-50-k8	24,152	21.7	23,905	–	23,855	–	156,722	103

(*Continues on the next page*)

Instance	UB	$time_{ub}$ (h)	LB_f	$z(IP)$	LB^f_{root}	$time$ (s)	$time_{prc}$ (s)	$nodes$
X-n331-66-k10	26,247	26.8	26,082	–	26,057	–	162,636	81
X-n331-80-k12	28,265	29.9	28,052	–	28,039	–	185,486	35
X-n336-50-k45	81,760	48.7	81,349	–	81,324	–	34,124	2,331
X-n336-66-k57	99,226	46.1	98,879	–	98,862	–	40,186	2,861
X-n336-80-k68	116,185	48.8	115,893	–	115,871	–	36,471	2,555
X-n344-50-k22	28,527	19.8	28,527	**28,527**	28,409	65,642	10,809	245
X-n344-66-k29	31,845	23.8	31,701	–	31,676	–	33,287	1,215
X-n344-80-k35	35,743	27.0	35,648	–	35,633	–	40,381	921
X-n351-50-k21	18,584	25.1	18,481	–	18,444	–	51,270	937
X-n351-66-k26	19,758	23.8	19,699	–	19,682	–	71,277	859
X-n351-80-k32	22,158	26.5	22,065	–	22,054	–	99,086	703
X-n359-50-k15	33,255	49.4	33,000	–	32,958	–	102,315	183
X-n359-66-k19	37,695	47.7	37,440	–	37,419	–	160,781	161
X-n359-80-k23	43,412	49.3	43,274	–	43,261	–	143,353	193
X-n367-50-k12	20,526	36.9	20,361	–	20,345	–	184,200	9
X-n367-66-k14	21,479	26.8	21,398	–	21,398	–	192,713	3
X-n367-80-k15	22,386	26.8	22,202	–	22,180	–	201,875	17
X-n376-50-k47	80,736	45.6	80,736	**80,736**	80,685	705	139	11
X-n376-66-k62	100,613	44.8	100,613	**100,613**	100,574	2,125	510	33
X-n376-80-k75	119,581	32.5	119,581	**119,581**	119,581	363	213	1
X-n384-50-k27	41,206	54.4	40,828	–	40,803	–	43,174	1,045
X-n384-66-k35	47,373	38.6	47,150	–	47,103	–	37,375	1,199
X-n384-80-k42	55,386	47.5	55,102	–	55,086	–	50,940	871
X-n393-50-k19	30,005	58.6	29,859	–	29,848	–	72,360	235
X-n393-66-k25	29,340	37.1	29,167	–	29,144	–	84,039	307
X-n393-80-k31	32,619	42.5	32,492	–	32,486	–	106,245	289
X-n401-50-k15	39,746	67.3	39,298	–	39,264	–	199,911	47
X-n401-66-k20	47,658	90.9	47,268	–	47,254	–	202,207	43
X-n401-80-k23	54,270	69.3	53,935	–	53,920	–	205,753	41
X-n411-50-k14	17,959	30.4	17,871	–	17,871	–	99,554	3
X-n411-66-k15	18,785	31.5	18,645	–	18,630	–	135,572	7
X-n411-80-k17	19,496	35.3	19,158	–	19,151	–	206,181	19
X-n420-50-k67	75,527	185.8	75,351	–	75,328	–	57,441	2,107
X-n420-66-k86	76,079	69.2	75,898	–	75,880	–	57,456	1,719
X-n420-80-k105	89,381	59.8	89,356	**89,356**	89,269	72,223	10,370	707
X-n429-50-k31	41,284	39.3	40,989	–	40,967	–	37,302	953
X-n429-66-k40	47,793	51.1	47,516	–	47,493	–	42,041	987
X-n429-80-k48	54,835	50.2	54,522	–	54,505	–	51,458	749
X-n439-50-k19	27,011	36.1	26,969	–	26,943	–	107,470	25
X-n439-66-k25	28,883	32.0	28,826	–	28,804	–	103,337	235

(*Continues on the next page*)

Instance	UB	$time_{ub}$ (h)	LB_f	$z(IP)$	LB^f_{root}	$time$ (s)	$time_{prc}$ (s)	$nodes$
X-n439-80-k30	32,074	38.3	31,999	–	31,986	–	135,395	165
X-n449-50-k15	36,929	55.1	36,498	–	36,469	–	137,845	125
X-n449-66-k20	41,846	75.8	41,323	–	41,313	–	192,684	93
X-n449-80-k23	46,738	65.7	46,082	–	46,061	–	203,178	49
X-n459-50-k14	18,891	39.7	18,724	–	18,691	–	181,996	11
X-n459-66-k18	20,561	39.6	20,159	–	20,132	–	182,592	49
X-n459-80-k21	22,047	52.8	21,733	–	21,719	–	197,129	55
X-n469-50-k70	123,817	154.8	122,783	123,773	122,730	–	45,176	2,703
X-n469-66-k90	148,455	96.2	148,185	–	148,164	–	27,860	2,595
X-n469-80-k109	178,511	93.8	178,067	–	178,048	–	30,321	2,083
X-n480-50-k36	52,309	95.7	51,933	–	51,897	–	51,958	507
X-n480-66-k47	63,577	80.1	63,314	–	63,297	–	76,292	563
X-n480-80-k56	73,993	100.9	73,650	–	73,632	–	105,667	313
X-n491-50-k30	43,952	117.2	43,512	–	43,489	–	132,730	121
X-n491-66-k39	49,627	87.7	49,299	–	49,151	–	91,111	339
X-n491-80-k47	56,141	80.6	55,610	–	55,566	–	137,699	303
X-n502-50-k20	40,591	130.6	40,454	–	40,444	–	122,851	11
X-n502-66-k26	49,285	113.2	49,204	–	49,194	–	107,574	27
X-n502-80-k31	56,997	96.5	56,937	–	56,922	–	177,770	11
X-n513-50-k11	21,675	44.2	21,417	–	21,417	–	132,874	3
X-n513-66-k14	22,426	35.7	22,133	–	22,133	–	145,374	3
X-n513-80-k17	23,448	35.3	23,081	–	23,081	–	206,960	3
X-n524-50-k125	154,137	80.5	154,137	**154,137**	154,080	1,829	778	39
X-n524-66-k129	154,416	99.9	154,416	**154,416**	154,360	12,384	4,118	255
X-n524-80-k132	154,497	90.1	154,446	**154,446**	154,413	3,549	1,782	45
X-n536-50-k49	54,658	116.3	54,249	–	54,193	–	104,940	303
X-n536-66-k64	66,032	114.7	65,688	–	65,668	–	114,173	385
X-n536-80-k77	77,811	140.8	77,513	–	77,495	–	149,699	297
X-n548-50-k25	53,049	147.7	52,681	–	52,649	–	122,148	73
X-n548-66-k33	61,421	134.8	61,259	–	61,243	–	142,637	89
X-n548-80-k40	71,867	161.2	71,761	–	71,749	–	155,396	139
X-n561-50-k22	31,826	64.4	31,336	–	31,307	–	190,355	43
X-n561-66-k28	34,370	59.8	34,129	–	34,100	–	190,425	43
X-n561-80-k34	38,053	71.1	37,637	–	37,588	–	187,865	87
X-n573-50-k22	40,239	110.4	40,003	–	40,003	–	117,244	1
X-n573-66-k25	44,151	166.1	43,765	–	43,765	–	181,946	1
X-n573-80-k27	47,054	189.2	46,580	–	46,580	–	200,243	3
X-n586-50-k80	122,632	515.9	121,890	–	121,857	–	128,653	585
X-n586-66-k105	140,396	204.3	140,017	–	139,991	–	71,338	821
X-n586-80-k127	160,390	178.2	160,006	–	159,982	–	79,723	613

(*Continues on the next page*)

Instance	UB	$time_{ub}$ (h)	LB_f	$z(IP)$	LB^f_{root}	$time$ (s)	$time_{prc}$ (s)	$nodes$
X-n599-50-k47	65,292	151.3	64,384	–	64,334	–	137,453	197
X-n599-66-k61	76,472	115.8	76,040	–	76,018	–	94,481	339
X-n599-80-k74	89,844	145.7	89,234	–	89,220	–	135,692	243
X-n613-50-k32	40,838	89.2	40,375	–	40,356	–	141,694	75
X-n613-66-k41	46,074	94.9	45,445	–	45,358	–	188,735	95
X-n613-80-k50	52,096	125.0	51,385	–	51,376	–	192,914	151
X-n627-50-k22	38,096	211.2	37,734	–	37,717	–	154,012	71
X-n627-66-k29	44,782	296.3	44,201	–	44,186	–	194,269	61
X-n627-80-k35	52,429	323.5	51,775	–	51,768	–	198,419	53
X-n641-50-k18	42,333	160.8	41,579	–	41,565	–	183,134	39
X-n641-66-k23	47,501	151.1	46,866	–	46,858	–	199,661	19
X-n641-80-k28	54,116	171.0	53,398	–	53,390	–	204,268	13
X-n655-50-k66	59,442	200.6	59,233	–	59,217	–	124,641	515
X-n655-66-k87	72,456	160.8	72,424	–	72,418	–	91,208	787
X-n655-80-k105	86,564	158.6	86,564	**86,564**	86,542	50,660	23,748	165
X-n670-50-k112	144,707	165.6	144,637	–	144,627	–	122,379	45
X-n670-66-k117	144,990	165.0	144,846	–	144,819	–	136,613	27
X-n670-80-k120	145,275	194.2	145,054	–	145,036	–	193,090	17
X-n685-50-k43	48,023	150.5	47,498	–	47,479	–	170,949	73
X-n685-66-k54	53,240	146.2	52,595	–	52,580	–	196,578	55
X-n685-80-k62	59,301	130.1	58,697	–	58,692	–	202,085	51
X-n701-50-k23	51,390	332.9	50,714	–	50,657	–	187,874	39
X-n701-66-k30	58,844	277.9	58,042	–	58,033	–	200,435	9
X-n701-80-k36	68,618	278.6	67,735	–	67,722	–	207,724	7
X-n716-50-k18	29,757	217.6	29,195	–	29,189	–	162,933	7
X-n716-66-k23	32,527	269.4	31,905	–	31,905	–	200,782	1
X-n716-80-k28	37,976	264.4	37,338	–	37,338	–	191,495	1
X-n733-50-k83	80,585	202.9	79,856	–	79,821	–	116,949	267
X-n733-66-k102	92,156	200.8	91,757	–	91,723	–	74,442	455
X-n733-80-k125	110,659	225.0	110,238	–	110,223	–	128,547	197
X-n749-50-k49	47,740	295.5	47,110	–	47,082	–	150,513	91
X-n749-66-k63	55,560	251.0	54,765	–	54,754	–	196,196	75
X-n749-80-k78	63,991	265.7	63,189	–	63,182	–	200,628	95
X-n766-50-k58	95,674	341.3	94,819	–	94,819	–	165,150	1
X-n766-66-k62	101,566	390.6	100,651	–	100,633	–	196,948	5
X-n766-80-k65	106,758	406.3	105,675	–	105,665	–	205,395	15
X-n783-50-k24	49,027	263.3	47,777	–	47,758	–	198,844	19
X-n783-66-k31	53,429	243.3	52,496	–	52,496	–	177,559	3
X-n783-80-k38	60,937	302.2	59,880	–	59,873	–	196,223	3
X-n801-50-k20	48,459	369.0	48,024	–	48,015	–	180,780	5

(*Continues on the next page*)

Instance	UB	$time_{ub}$ (h)	LB_f	$z(IP)$	LB^f_{root}	$time$ (s)	$time_{prc}$ (s)	$nodes$
X-n801-66-k27	54,929	425.8	54,193	–	54,193	–	184,590	3
X-n801-80-k32	62,698	398.0	62,000	–	62,000	–	180,045	3
X-n819-50-k86	89,296	332.4	88,170	–	88,144	–	128,254	189
X-n819-66-k112	108,431	338.2	107,726	–	107,688	–	132,300	187
X-n819-80-k136	128,617	354.4	127,891	–	127,874	–	144,472	151
X-n837-50-k71	116,553	639.6	115,016	–	114,975	–	124,162	111
X-n837-66-k94	129,183	463.3	128,236	–	128,203	–	157,542	137
X-n837-80-k114	154,966	476.1	154,098	–	154,084	–	173,868	177
X-n856-50-k48	57,777	368.6	57,543	–	57,529	–	128,258	53
X-n856-66-k63	63,542	296.5	63,241	–	63,222	–	157,856	37
X-n856-80-k76	73,802	279.1	73,549	–	73,534	–	168,100	57
X-n876-50-k30	58,780	460.2	57,903	–	57,891	–	186,246	35
X-n876-66-k38	69,617	447.8	68,489	–	68,480	–	203,526	11
X-n876-80-k46	80,983	548.0	80,204	–	80,204	–	207,511	3
X-n895-50-k19	40,668	252.7	39,807	–	39,807	–	161,965	3
X-n895-66-k25	44,059	277.5	43,052	–	43,052	–	132,947	1
X-n895-80-k30	48,451	338.5	47,375	–	47,375	–	188,132	1
X-n916-50-k105	190,108	644.1	187,568	–	187,527	–	107,633	165
X-n916-66-k136	222,807	597.7	221,570	–	221,517	–	122,939	143
X-n916-80-k165	263,885	596.4	262,762	–	262,720	–	138,277	123
X-n936-50-k132	127,497	240.2	127,347	–	127,323	–	164,181	25
X-n936-66-k138	128,871	320.1	128,475	–	128,444	–	199,818	33
X-n936-80-k143	130,808	396.8	129,839	–	129,818	–	204,051	61
X-n957-50-k44	57,019	620.3	56,340	–	56,298	–	162,896	23
X-n957-66-k58	62,593	459.0	62,087	–	62,070	–	190,143	29
X-n957-80-k70	71,855	470.8	71,277	–	71,240	–	203,048	29
X-n979-50-k30	69,739	671.7	68,032	–	68,011	–	183,342	5
X-n979-66-k39	84,499	657.6	83,100	–	83,100	–	212,646	1
X-n979-80-k47	99,605	681.1	98,338	–	98,338	–	195,325	1
X-n1001-50-k22	49,978	505.6	48,927	–	48,927	–	152,333	1
X-n1001-66-k28	56,126	503.8	55,093	–	55,093	–	174,370	1
X-n1001-80-k34	63,278	600.7	62,097	–	62,097	–	161,585	1

APPENDIX G – Results of BCP algorithms when no upper bound is given as input

Table G.1: Results obtained for the GJB instances when no upper bound is given as input

Problem data					$z(IP)$	$\text{BCP}_{\mathcal{F}1}$			$\text{BCP}_{\mathcal{F}2}$		
Instance	$n+m$	n	m	K		LB^f_{root}	*time* (s)	*nodes*	LB^f_{root}	*time* (s)	*nodes*
A1	25	20	5	8	229,885.65	229,885.65	2	1	229,885.65	2	1
A2	25	20	5	5	180,119.21	180,119.21	5	1	179,954.01	7	3
A3	25	20	5	4	163,405.38	163,405.38	2	1	163,405.38	2	1
A4	25	20	5	3	155,796.41	155,796.41	3	1	155,796.41	2	1
B1	30	20	10	7	239,080.16	239,080.16	2	1	239,080.16	2	1
B2	30	20	10	5	198,047.77	198,047.77	10	1	197,763.04	5	3
B3	30	20	10	3	169,372.29	169,372.29	3	1	169,372.29	2	1
C1	40	20	20	7	250,556.77	250,556.77	7	1	250,556.77	3	1
C2	40	20	20	5	215,020.23	215,020.23	12	1	215,020.23	4	1
C3	40	20	20	5	199,345.96	199,345.96	2	1	199,345.96	3	1
C4	40	20	20	4	195,366.63	195,366.63	2	1	195,366.63	2	1
D1	38	30	8	12	322,530.13	322,530.13	2	1	322,530.13	3	1
D2	38	30	8	11	316,708.86	316,708.86	3	1	316,708.86	3	1
D3	38	30	8	7	239,478.63	239,478.63	6	1	238,777.54	12	3
D4	38	30	8	5	205,831.94	204,542.88	14	3	204,497.82	18	3
E1	45	30	15	7	238,879.58	238,879.58	3	1	238,879.58	2	1
E2	45	30	15	4	212,263.11	212,263.11	3	1	212,263.11	3	1
E3	45	30	15	4	206,659.17	206,659.17	6	1	206,659.17	4	1
F1	60	30	30	6	263,173.96	263,173.96	5	1	262,966.91	17	3
F2	60	30	30	7	265,214.16	265,214.16	3	1	265,214.16	3	1
F3	60	30	30	5	241,120.78	241,120.78	3	1	241,120.78	4	1
F4	60	30	30	4	233,861.85	233,861.85	11	1	233,861.85	7	1
G1	57	45	12	10	306,305.40	306,305.40	7	1	305,866.21	15	3

(Continues on the next page)

Problem data					$z(IP)$	BCP$_{\mathcal{F}1}$			BCP$_{\mathcal{F}2}$		
Instance	$n+m$	n	m	K		LB^f_{root}	*time* (s)	*nodes*	LB^f_{root}	*time* (s)	*nodes*
G2	57	45	12	6	245,440.99	245,440.99	5	1	245,440.99	6	1
G3	57	45	12	5	229,507.48	229,507.48	13	1	229,043.53	19	3
G4	57	45	12	6	232,521.25	232,521.25	8	1	232,347.93	24	3
G5	57	45	12	5	221,730.35	221,440.66	33	3	221,726.90	86	3
G6	57	45	12	4	213,457.45	213,457.45	7	1	213,457.45	6	1
H1	68	45	23	6	268,933.06	268,933.06	70	1	268,933.06	17	1
H2	68	45	23	5	253,365.50	253,365.50	14	1	253,365.50	7	1
H3	68	45	23	4	247,449.04	247,449.04	21	1	247,449.04	9	1
H4	68	45	23	5	250,220.77	250,220.77	41	1	250,220.77	6	1
H5	68	45	23	4	246,121.31	246,121.31	20	1	246,121.31	7	1
H6	68	45	23	5	249,135.32	249,135.32	16	1	249,135.32	6	1
I1	90	45	45	10	350,245.28	349,824.74	51	3	349,993.59	36	3
I2	90	45	45	7	309,943.84	309,943.84	22	1	309,943.84	12	1
I3	90	45	45	5	294,507.38	294,407.06	105	3	294,433.79	49	3
I4	90	45	45	6	295,988.45	295,256.19	154	3	293,883.49	38	5
I5	90	45	45	7	301,236.01	301,236.01	46	1	300,603.94	24	5
J1	94	75	19	10	335,006.68	335,006.68	28	1	334,384.98	70	3
J2	94	75	19	8	310,417.21	309,987.63	284	3	308,968.01	185	3
J3	94	75	19	6	279,219.21	279,219.21	130	1	279,068.01	191	3
J4	94	75	19	7	296,533.16	293,734.75	2,789	19	293,776.18	8,041	59
K1	113	75	38	10	394,071.17	394,071.17	41	1	392,638.99	76	5
K2	113	75	38	8	362,130.00	362,130.00	47	1	362,130.00	27	1
K3	113	75	38	9	365,694.08	365,694.08	53	1	365,694.08	36	1
K4	113	75	38	7	348,949.39	348,949.39	68	1	347,968.14	165	5
L1	150	75	75	10	417,896.71	417,896.71	322	1	417,332.11	220	3
L2	150	75	75	8	401,228.80	401,220.82	591	3	400,645.23	288	3
L3	150	75	75	9	402,677.72	402,677.72	250	1	402,677.72	71	1
L4	150	75	75	7	384,636.33	384,636.33	139	1	384,636.33	60	1
L5	150	75	75	8	387,564.55	387,564.55	126	1	387,564.55	59	1
M1	125	100	25	11	398,593.19	398,430.30	773	3	397,620.41	434	3
M2	125	100	25	10	396,916.97	396,466.48	347	3	395,706.60	506	3
M3	125	100	25	9	375,695.42	372,231.71	51,574	299	371,764.59	48,057	311
M4	125	100	25	7	348,140.16	347,865.37	423	3	346,956.35	610	3
N1	150	100	50	11	408,100.62	408,060.05	191	3	406,628.97	328	7
N2	150	100	50	10	408,065.44	407,596.22	362	3	406,213.20	411	9
N3	150	100	50	9	394,337.86	394,155.99	235	3	393,510.41	336	3
N4	150	100	50	10	394,788.36	394,678.28	312	3	394,006.39	614	5
N5	150	100	50	7	373,476.30	373,300.81	329	3	372,444.74	289	5
N6	150	100	50	8	373,758.65	373,629.92	281	3	372,753.43	237	5

(*Continues on the next page*)

Problem data					$z(IP)$	BCP$_{\mathcal{F}1}$			BCP$_{\mathcal{F}2}$		
Instance	$n+m$	n	m	K		LB^f_{root}	*time (s)*	*nodes*	LB^f_{root}	*time (s)*	*nodes*
O1	200	100	100	10	478,126.75	474,563.78	133,107	201	475,203.15	7,189	27
O2	200	100	100	11	477,256.15	476,353.83	1,833	9	476,890.24	292	3
O3	200	100	100	9	457,294.48	457,108.58	725	3	457,294.48	288	1
O4	200	100	100	10	458,874.87	458,874.87	235	1	458,874.87	101	1
O5	200	100	100	7	436,974.20	435,987.58	6,361	19	436,531.84	11,314	9
O6	200	100	100	8	438,004.69	437,496.80	899	3	437,827.10	698	3
Mean							2,994.1			1,201.0	
Geometric mean							45.6			36.1	

Table G.2: Results obtained for the TV instances when no upper bound is given as input

Problem data					$z(IP)$	BCP$_{\mathcal{F}1}$			BCP$_{\mathcal{F}2}$		
Instance	$n+m$	n	m	K		LB^f_{root}	*time (s)*	*nodes*	LB^f_{root}	*time (s)*	*nodes*
E-n22-50	21	10	11	3	371	371	7	1	371	2	1
E-n22-66	21	14	7	3	366	366	6	1	366	2	1
E-n22-80	21	17	4	3	375	375	7	1	375	2	1
E-n23-50	22	11	11	2	682	682	7	1	682	2	1
E-n23-66	22	15	7	2	649	646	17	3	649	5	1
E-n23-80	22	18	4	2	623	623	11	1	623	3	1
E-n30-50	29	14	15	2	501	501	8	1	501	4	1
E-n30-66	29	19	10	3	537	537	17	1	537	2	1
E-n30-80	29	23	6	3	514	514	8	1	514	3	1
E-n33-50	32	16	16	3	738	738	8	1	738	6	1
E-n33-66	32	21	11	3	750	750	8	1	750	2	1
E-n33-80	32	26	6	3	736	736	8	1	736	2	1
E-n51-50	50	25	25	3	559	559	8	1	559	5	1
E-n51-66	50	33	17	4	548	548	9	1	548	4	1
E-n51-80	50	40	10	4	565	565	17	1	565	5	1
E-n76-A-50	75	38	37	6	739	738	39	3	739	12	1
E-n76-A-66	75	50	25	7	768	768	17	1	768	10	1
E-n76-A-80	75	60	15	8	781	781	20	1	781	8	1
E-n76-B-50	75	38	37	8	801	801	17	1	801	7	1
E-n76-B-66	75	50	25	10	873	872	26	3	873	20	3
E-n76-B-80	75	60	15	12	919	919	18	1	919	6	1
E-n76-C-50	75	38	37	5	713	710	30	3	713	30	1
E-n76-C-66	75	50	25	6	734	734	33	1	734	19	1
E-n76-C-80	75	60	15	7	733	731	628	15	731	528	9
E-n76-D-50	75	38	37	4	690	690	17	1	690	6	1

(*Continues on the next page*)

Problem data					$z(IP)$	BCP$_{\mathcal{F}1}$			BCP$_{\mathcal{F}2}$		
Instance	$n+m$	n	m	K		LB^f_{root}	$time$ (s)	$nodes$	LB^f_{root}	$time$ (s)	$nodes$
E-n76-D-66	75	50	25	5	715	713	55	3	715	50	1
E-n76-D-80	75	60	15	6	694	694	177	3	694	147	1
E-n101-A-50	100	50	50	4	831	831	52	1	831	61	1
E-n101-A-66	100	66	34	6	846	846	25	1	846	22	1
E-n101-A-80	100	80	20	6	856	853	2,364	21	853	1,191	9
E-n101-B-50	100	50	50	7	923	921	201	5	923	134	3
E-n101-B-66	100	66	34	9	982	973	3,621	65	974	4,106	37
E-n101-B-80	100	80	20	11	1,008	1,004	850	17	1,006	222	5
Average							252.6			200.8	
Geometric mean							29.9			13.4	

Table G.3: Results obtained for the FTV instances when no upper bound is given as input.

Problem data					$z(IP)$	BCP$_{\mathcal{F}1}$			BCP$_{\mathcal{F}2}$		
Instance	$n+m$	n	m	K		LB^f_{root}	$time$	$nodes$	LB^f_{root}	$time$	$nodes$
FTV33_50	33	17	16	2	1,841	1,841	10	1	1,841	2	1
FTV33_66	33	22	11	2	1,899	1,899	11	1	1,899	3	1
FTV33_80	33	27	6	2	1,704	1,704	10	1	1,704	3	1
FTV35_50	35	18	17	2	2,077	2,077	21	1	2,077	12	1
FTV35_66	35	24	11	2	2,150	2,150	18	1	2,141	12	3
FTV35_80	35	28	7	2	1,996	1,996	17	1	1,996	6	1
FTV38_50	38	19	19	2	2,162	2,162	17	1	2,162	3	1
FTV38_66	38	26	12	2	2,132	2,132	20	1	2,122	14	3
FTV38_80	38	31	7	3	1,982	1,982	11	1	1,982	4	1
FTV44_50	44	22	22	2	2,348	2,336	140	3	2,336	62	3
FTV44_66	44	30	14	2	2,225	2,213	181	3	2,210	222	5
FTV44_80	44	36	8	3	2,184	2,184	68	1	2,178	66	3
FTV47_50	47	24	23	2	2,343	2,342	88	3	2,343	16	1
FTV47_66	47	32	15	2	2,427	2,427	20	1	2,427	4	1
FTV47_80	47	38	9	2	2,312	2,312	20	1	2,312	9	1
FTV55_50	55	28	27	2	2,425	2,423	190	3	2,425	18	1
FTV55_66	55	37	18	2	2,246	2,240	395	3	2,238	186	3
FTV55_80	55	44	11	2	2,264	2,264	104	1	2,264	21	1
FTV64_50	64	32	32	2	2,728	2,728	142	1	2,728	36	1
FTV64_66	64	43	21	2	2,673	2,670	297	3	2,671	142	3
FTV64_80	64	52	12	3	2,659	2,724	91	1	2,659	55	1
FTV70_50	70	35	35	2	2,934	2,914	207	3	2,934	58	1
FTV70_66	70	47	23	2	2,808	2,808	69	1	2,808	24	1
FTV70_80	70	56	14	2	2,684	2,684	168	1	2,684	81	1
Average							96.4			44.1	
Geometric mean							51.1			18.4	

Table G.4: Comparison between the two BCP algorithms for the X instances when no upper bound is given as input. Only the first 45 instances of X were considered. The value *time* is not given for executions which stopped by the time limit.

Instance	BCP$_{\mathcal{F}1}$						BCP$_{\mathcal{F}2}$					
	$z(IP)$	LB^f	LB^f_{root}	$time$ (s)	$time_{prc}$ (s)	$nodes$	$z(IP)$	LB^f	LB^f_{root}	$time$ (s)	$time_{prc}$ (s)	$nodes$
X-n101-50-k13	19,033	19,033	18,940	538	83	21	19,033	19,033	18,913	378	55	29
X-n101-66-k17	20,490	20,490	20,347	1,822	333	119	20,490	20,490	20,343	818	138	53
X-n101-80-k21	23,305	23,305	23,227	142	39	15	23,305	23,305	23,225	323	47	17
X-n106-50-k7	15,413	15,413	15,413	52	22	1	15,413	15,413	15,413	31	12	1
X-n106-66-k9	18,984	18,984	18,972	300	84	3	18,984	18,984	18,972	174	49	3
X-n106-80-k11	22,131	22,131	22,084	9,317	2,875	75	22,131	22,131	22,090	3,816	910	43
X-n110-50-k7	13,103	13,103	13,103	36	12	1	13,103	13,103	13,103	28	9	1
X-n110-66-k9	13,598	13,598	13,598	31	14	1	13,598	13,598	13,598	27	9	1
X-n110-80-k11	14,302	14,302	14,200	2,377	631	33	14,302	14,302	14,201	3,229	720	45
X-n115-50-k8	13,927	13,927	13,927	62	25	1	13,927	13,927	13,927	37	11	1
X-n115-66-k8	14,032	14,032	14,032	75	30	1	14,032	14,032	14,031	122	32	3
X-n115-80-k9	13,536	13,536	13,536	101	51	1	13,536	13,536	13,536	50	29	1
X-n120-50-k3	12,416	12,416	12,392	1,092	659	3	12,416	12,416	12,398	291	97	5
X-n120-66-k4	13,145	13,145	13,097	4,297	2,641	13	13,145	13,145	13,106	776	438	5
X-n120-80-k5	13,528	13,528	13,457	15,336	10,713	57	13,528	13,528	13,464	3,855	2,325	15
X-n125-50-k16	32,224	32,224	32,075	11,558	1,875	249	32,224	32,224	32,061	2,312	305	81
X-n125-66-k19	36,400	36,400	36,330	3,743	2,185	33	36,400	36,400	36,326	1,160	532	13
X-n125-80-k23	43,960	43,960	43,787	211,149	71,657	2,249	44,795	43,825	43,779	–	74,880	2,659
X-n129-50-k10	19,468	19,468	19,402	6,611	1,220	25	19,468	19,468	19,395	929	174	17
X-n129-66-k12	22,606	22,606	22,532	7,687	1,512	99	22,606	22,606	22,541	2,115	383	45
X-n129-80-k14	24,575	24,575	24,550	2,835	898	25	24,575	24,575	24,546	606	178	9
X-n134-50-k7	8,369	8,369	8,310	37,341	23,991	121	8,369	8,369	8,324	2,664	1,700	19
X-n134-66-k9	9,132	8,918	8,909	–	114,867	1,515	8,974	8,974	8,900	7,248	5,564	81
X-n134-80-k11	9,699	9,699	9,634	168,426	111,455	703	9,945	9,640	9,610	–	143,661	2,539
X-n139-50-k5	13,281	13,281	13,199	21,818	11,624	67	13,281	13,281	13,198	1,192	608	9
X-n139-66-k7	13,512	13,512	13,505	292	141	3	13,512	13,512	13,486	332	129	7
X-n139-80-k8	13,662	13,662	13,662	133	78	1	13,662	13,662	13,662	116	68	1
X-n143-50-k4	14,539	14,539	14,484	13,463	9,486	29	14,539	14,539	14,502	1,365	705	5
X-n143-66-k4	14,310	14,310	14,310	498	324	1	14,310	14,310	14,310	262	192	1
X-n143-80-k5	14,506	14,143	14,127	–	151,022	597	–	14,149	14,130	–	201,798	139
X-n148-50-k25	28,210	28,210	28,165	1,334	249	67	28,210	28,210	28,142	290	54	17
X-n148-66-k29	30,482	30,482	30,376	563	124	23	30,482	30,482	30,367	2,136	408	81
X-n148-80-k36	35,430	35,430	35,327	766	174	39	35,430	35,430	35,316	2,947	484	81
X-n153-50-k19	20,536	20,536	20,536	62	34	1	20,536	20,536	20,536	44	17	3
X-n153-66-k20	20,613	20,613	20,609	181	81	5	20,613	20,613	20,610	128	40	5
X-n153-80-k21	20,819	20,819	20,811	99	61	3	20,819	20,819	20,813	120	55	3
X-n157-50-k7	11,727	11,727	11,727	500	94	1	11,727	11,727	11,727	82	30	1
X-n157-66-k9	13,651	13,651	13,651	94	45	1	13,651	13,651	13,651	80	31	1
X-n157-80-k11	15,264	15,264	15,240	941	412	5	15,264	15,264	15,237	2,896	1,035	15
X-n162-50-k6	12,812	12,812	12,772	23,249	15,383	61	12,812	12,812	12,780	1,939	1,300	5
X-n162-66-k8	13,668	13,315	13,286	–	128,290	953	13,417	13,417	13,311	77,117	52,893	315
X-n162-80-k9	13,854	13,854	13,808	12,292	8,266	47	13,854	13,854	13,815	28,871	19,791	89
X-n167-50-k5	16,489	16,489	16,433	4,877	2,848	9	16,489	16,489	16,436	1,457	524	7
X-n167-66-k7	17,827	17,827	17,738	50,869	30,683	115	–	17,763	17,703	–	181,044	69
X-n167-80-k8	19,415	19,415	19,367	5,216	3,665	17	19,415	19,415	19,367	10,573	7,390	57
Average				28,226.1	16,588.6	134.0				22,820.9	15,574.5	146.6

(*Continues on the next page*)

Instance	BCP$_{\mathcal{F}1}$						BCP$_{\mathcal{F}2}$					
	$z(IP)$	LB^f	LB^f_{root}	$time$ (s)	$time_{prc}$ (s)	$nodes$	$z(IP)$	LB^f	LB^f_{root}	$time$ (s)	$time_{prc}$ (s)	$nodes$
Geometric mean				1,940.0	798.7	15.7				1,028.4	379.8	13.1

Table G.5: Results for X instances by the BCP$_{\mathcal{F}2}$ with a time limit of 60 hours and no upper bound given as input. The results which were already reported in Table 12 were omitted. The value *time* is not given for executions which stopped by the time limit.

Instance	LB_f	$z(IP)$	LB^f_{root}	$time$	$time_{prc}$	$nodes$
X-n172-66-k31	31,864	31,864	31,799	4,927	711	137
X-n172-80-k39	36,803	36,803	36,738	18,997	3,568	357
X-n176-50-k23	45,239	45,239	45,151	1,656	357	57
X-n176-66-k24	46,416	46,416	46,324	2,218	604	61
X-n176-80-k25	47,033	47,033	46,966	1,798	797	15
X-n181-50-k12	16,549	16,549	16,540	250	56	3
X-n181-66-k15	18,832	18,832	18,832	118	42	1
X-n181-80-k18	21,241	21,241	21,236	152	53	3
X-n186-50-k8	17,978	17,978	17,829	28,946	15,245	71
X-n186-66-k10	19,751	19,751	19,706	20,320	16,431	45
X-n186-80-k12	21,630	–	21,609	–	192,394	179
X-n190-50-k4	11,552	11,552	11,467	13,607	9,852	27
X-n190-66-k5	12,727	–	12,717	–	160,121	53
X-n190-80-k6	14,336	–	14,290	–	199,173	81
X-n195-50-k27	29,470	29,470	29,359	2,238	476	49
X-n195-66-k34	33,137	33,137	33,062	1,639	330	35
X-n195-80-k42	38,629	38,629	38,554	1,828	410	29
X-n200-50-k18	34,316	34,416	34,284	–	67,417	1,967
X-n200-66-k24	40,335	40,525	40,310	–	51,793	2,373
X-n200-80-k29	47,741	47,741	47,697	5,073	1,104	59
X-n204-50-k10	15,858	–	15,804	–	159,889	49
X-n204-66-k12	16,573	–	16,540	–	178,723	229
X-n204-80-k15	17,832	17,832	17,779	7,159	4,377	31
X-n209-50-k8	21,728	–	21,623	–	162,003	87
X-n209-66-k11	24,264	–	24,196	–	180,544	67
X-n209-80-k13	26,981	–	26,956	–	199,446	155
X-n214-50-k6	9,574	9,574	9,536	5,760	3,371	9
X-n214-66-k8	9,986	–	9,966	–	135,415	37
X-n214-80-k9	10,374	–	10,321	–	198,525	49
X-n219-50-k37	64,691	64,691	64,619	554	136	19
X-n219-66-k48	80,405	80,405	80,315	602	119	23
X-n219-80-k59	95,845	95,845	95,743	987	169	21
X-n223-50-k18	27,442	27,442	27,304	24,716	5,521	153
X-n223-66-k23	30,717	30,717	30,537	130,095	20,136	691
X-n223-80-k27	34,341	–	34,314	–	156,173	365
X-n228-50-k19	23,128	23,128	23,067	7,585	4,263	63
X-n228-66-k20	24,113	24,113	24,048	26,733	10,889	129
X-n228-80-k21	24,592	24,592	24,561	4,811	3,233	29
X-n233-50-k10	17,120	–	17,065	–	198,842	29
X-n233-66-k12	18,026	18,026	17,921	56,478	46,735	119
X-n233-80-k14	18,634	–	18,509	–	202,953	119
X-n237-50-k7	20,745	20,745	20,626	21,627	10,739	35

(*Continues on the next page*)

Instance	LB_f	$z(IP)$	LB^f_{root}	*time*	$time_{prc}$	*nodes*
X-n237-66-k9	22,409	22,474	22,379	–	143,449	35
X-n237-80-k11	24,309	–	24,285	–	184,129	63
X-n242-50-k25	47,719	48,755	47,652	–	116,477	1,133
X-n242-66-k32	57,044	57,424	57,018	–	56,575	1,295
X-n242-80-k39	68,849	–	68,806	–	157,604	409
X-n247-50-k42	36,701	36,701	36,701	102	59	1
X-n247-66-k43	36,994	36,994	36,991	244	96	3
X-n247-80-k45	37,205	37,205	37,196	672	314	5
X-n251-50-k14	24,895	25,049	24,836	–	100,502	507
X-n251-66-k18	27,710	–	27,663	–	158,894	177
X-n251-80-k22	32,015	32,684	31,964	–	170,608	263
X-n256-50-k8	15,922	15,922	15,905	1,490	822	3
X-n256-66-k11	17,250	17,250	17,226	3,036	2,088	7
X-n256-80-k13	18,065	–	18,031	–	200,350	159
X-n261-50-k7	21,467	–	21,418	–	178,362	27
X-n261-66-k9	22,836	–	22,657	–	198,780	49
X-n261-80-k11	24,707	–	24,675	–	202,201	87
X-n266-50-k30	47,677	48,003	47,642	–	51,584	1,629
X-n266-66-k39	55,776	56,213	55,746	–	45,164	1,367
X-n266-80-k47	63,708	64,282	63,668	–	61,812	1,211
X-n270-50-k18	24,751	24,751	24,639	100,563	35,402	253
X-n270-66-k24	26,377	26,377	26,307	14,158	4,865	77
X-n270-80-k29	29,677	30,031	29,627	–	124,368	411
X-n275-50-k14	15,561	15,561	15,491	84,543	39,864	135
X-n275-66-k19	16,944	16,944	16,918	1,456	519	11
X-n275-80-k22	18,688	18,688	18,643	45,886	26,882	65
X-n280-50-k13	29,054	–	28,979	–	187,073	51
X-n280-66-k15	31,082	–	31,045	–	196,817	85
X-n280-80-k16	31,756	–	31,736	–	201,582	75
X-n284-50-k8	15,860	–	15,822	–	181,874	87
X-n284-66-k10	17,220	–	17,185	–	182,809	59
X-n284-80-k12	18,676	–	18,656	–	200,106	75
X-n289-50-k34	57,558	–	57,514	–	141,794	373
X-n289-66-k38	63,191	64,136	63,160	–	176,127	357
X-n289-80-k47	75,619	–	75,603	–	163,495	391
X-n294-50-k26	30,859	30,859	30,711	16,360	2,660	109
X-n294-66-k33	34,544	34,976	34,508	–	166,014	215
X-n294-80-k40	39,070	39,364	39,032	–	46,358	1,039
X-n298-50-k16	24,956	25,119	24,897	–	149,917	199
X-n298-66-k21	27,502	–	27,435	–	186,681	125
X-n298-80-k25	30,113	30,223	30,062	–	112,642	355
X-n303-50-k11	17,664	–	17,614	–	187,344	43
X-n303-66-k13	18,049	–	18,016	–	192,353	45
X-n303-80-k16	19,416	–	19,374	–	200,219	155
X-n308-50-k9	22,302	–	22,288	–	184,265	9
X-n308-66-k11	23,623	–	23,607	–	191,971	19
X-n308-80-k12	24,404	–	24,371	–	206,654	29
X-n313-50-k39	57,489	–	57,450	–	169,743	453
X-n313-66-k44	59,928	60,193	59,897	–	53,724	1,505
X-n313-80-k56	73,658	75,274	73,633	–	151,610	457
X-n317-50-k27	43,391	43,391	43,363	3,696	914	31
X-n317-66-k35	54,502	54,502	54,470	3,929	1,118	33

(Continues on the next page)

Instance	LB_f	$z(IP)$	LB^f_{root}	$time$	$time_{prc}$	$nodes$
X-n317-80-k43	63,683	63,683	63,656	1,721	679	11
X-n322-50-k14	23,159	23,314	23,070	–	137,205	103
X-n322-66-k19	25,034	25,034	24,906	153,564	125,925	131
X-n322-80-k23	27,353	–	27,315	–	191,199	279
X-n327-50-k10	21,378	–	21,334	–	179,999	75
X-n327-66-k13	23,148	–	23,131	–	189,678	49
X-n327-80-k16	24,590	–	24,537	–	200,928	99
X-n331-50-k8	23,961	–	23,910	–	181,875	37
X-n331-66-k10	26,108	–	26,072	–	177,596	41
X-n331-80-k12	28,007	–	27,965	–	195,745	39
X-n336-50-k45	81,346	82,173	81,302	–	150,804	377
X-n336-66-k57	98,880	99,755	98,842	–	179,016	587
X-n336-80-k68	115,881	–	115,840	–	164,639	451
X-n344-50-k22	28,424	28,542	28,358	–	74,562	379
X-n344-66-k29	31,679	–	31,631	–	162,020	155
X-n344-80-k35	35,639	–	35,590	–	179,234	285
X-n351-50-k21	18,475	–	18,429	–	161,461	135
X-n351-66-k26	19,688	–	19,669	–	197,745	121
X-n351-80-k32	22,045	–	22,031	–	191,584	243
X-n359-50-k15	32,992	–	32,911	–	166,992	73
X-n359-66-k19	37,418	–	37,399	–	189,679	127
X-n359-80-k23	43,219	–	43,195	–	200,330	75
X-n367-50-k12	20,095	–	20,021	–	188,261	11
X-n367-66-k14	21,147	–	21,134	–	191,684	23
X-n367-80-k15	21,979	–	21,969	–	200,625	33
X-n376-50-k47	80,736	80,736	80,672	13,337	2,721	125
X-n376-66-k62	100,613	100,613	100,553	7,391	1,713	57
X-n376-80-k75	119,581	119,581	119,525	2,054	557	15
X-n384-50-k27	40,836	–	40,756	–	127,902	257
X-n384-66-k35	47,133	–	47,051	–	162,590	175
X-n384-80-k42	55,078	–	55,029	–	179,261	243
X-n393-50-k19	29,849	–	29,803	–	146,486	159
X-n393-66-k25	29,139	–	29,092	–	175,430	183
X-n393-80-k31	32,412	–	32,390	–	188,887	159
X-n401-50-k15	39,286	–	39,237	–	201,035	31
X-n401-66-k20	47,112	–	47,096	–	196,960	85
X-n401-80-k23	53,793	–	53,781	–	204,671	61
X-n411-50-k14	17,889	–	17,889	–	168,742	1
X-n411-66-k15	18,548	–	18,524	–	165,919	9
X-n411-80-k17	19,165	–	19,159	–	192,544	31
X-n420-50-k67	75,270	–	75,234	–	170,116	255
X-n420-66-k86	75,886	–	75,851	–	146,454	415
X-n420-80-k105	89,256	89,361	89,211	–	21,841	817
X-n429-50-k31	40,967	–	40,910	–	143,608	163
X-n429-66-k40	47,490	–	47,442	–	157,309	173
X-n429-80-k48	54,500	–	54,444	–	155,847	307
X-n439-50-k19	26,939	–	26,896	–	155,140	23
X-n439-66-k25	28,792	–	28,748	–	174,003	63
X-n439-80-k30	31,987	–	31,954	–	194,597	125
X-n449-50-k15	36,469	–	36,425	–	180,475	87
X-n449-66-k20	41,184	–	41,175	–	194,528	75
X-n449-80-k23	46,118	–	46,106	–	204,928	39

(*Continues on the next page*)

Instance	LB_f	$z(IP)$	LB^f_{root}	$time$	$time_{prc}$	$nodes$
X-n459-50-k14	18,716	–	18,663	–	188,181	15
X-n459-66-k18	20,160	–	20,134	–	199,722	27
X-n459-80-k21	21,729	–	21,715	–	202,383	43
X-n469-50-k70	122,782	–	122,739	–	124,900	715
X-n469-66-k90	148,155	148,644	148,118	–	30,066	1,111
X-n469-80-k109	178,031	–	177,972	–	96,065	625
X-n480-50-k36	51,912	–	51,856	–	100,624	207
X-n480-66-k47	63,301	–	63,264	–	126,718	177
X-n480-80-k56	73,637	–	73,591	–	139,247	181
X-n491-50-k30	43,505	–	43,452	–	161,965	113
X-n491-66-k39	49,243	–	49,202	–	183,503	187
X-n491-80-k47	55,546	–	55,430	–	196,367	145
X-n502-50-k20	40,451	–	40,412	–	148,349	21
X-n502-66-k26	49,197	–	49,179	–	173,587	41
X-n502-80-k31	56,923	–	56,902	–	197,501	47
X-n513-50-k11	21,304	–	21,304	–	152,033	1
X-n513-66-k14	22,072	–	22,059	–	155,223	7
X-n513-80-k17	22,990	–	22,975	–	185,413	5
X-n524-50-k125	154,137	154,137	154,071	4,683	1,417	49
X-n524-66-k129	154,416	154,416	154,348	13,052	2,413	153
X-n524-80-k132	154,446	154,446	154,403	5,161	1,282	43
X-n536-50-k49	54,244	–	54,164	–	176,542	235
X-n536-66-k64	65,675	–	65,637	–	172,730	235
X-n536-80-k77	77,495	–	77,479	–	189,184	243
X-n548-50-k25	52,675	–	52,589	–	145,190	59
X-n548-66-k33	61,233	–	61,190	–	177,050	53
X-n548-80-k40	71,734	–	71,692	–	195,468	59
X-n561-50-k22	31,331	–	31,288	–	182,596	71
X-n561-66-k28	34,002	–	33,927	–	197,346	63
X-n561-80-k34	37,609	–	37,595	–	198,839	61
X-n573-50-k22	39,993	–	39,993	–	163,363	1
X-n573-66-k25	43,893	–	43,893	–	159,842	1
X-n573-80-k27	46,597	–	46,591	–	202,900	9
X-n586-50-k80	121,875	–	121,829	–	118,335	323
X-n586-66-k105	139,982	–	139,934	–	97,273	357
X-n586-80-k127	159,944	–	159,887	–	116,518	251
X-n599-50-k47	64,358	–	64,292	–	135,566	121
X-n599-66-k61	75,998	–	75,916	–	158,683	221
X-n599-80-k74	89,160	–	89,136	–	167,541	213
X-n613-50-k32	40,352	–	40,284	–	164,930	97
X-n613-66-k41	45,516	–	45,456	–	193,379	81
X-n613-80-k50	51,445	–	51,435	–	195,546	123
X-n627-50-k22	37,671	–	37,642	–	175,239	59
X-n627-66-k29	44,028	–	43,700	–	194,965	37
X-n627-80-k35	51,303	–	51,295	–	200,357	41
X-n641-50-k18	41,646	–	41,632	–	183,553	37
X-n641-66-k23	46,725	–	46,714	–	202,733	17
X-n641-80-k28	53,271	–	53,261	–	203,571	17
X-n655-50-k66	59,234	–	59,216	–	155,546	163
X-n655-66-k87	72,406	–	72,348	–	174,320	75
X-n655-80-k105	86,527	–	86,486	–	183,137	75
X-n670-50-k112	144,621	144,688	144,602	–	140,047	89

(*Continues on the next page*)

Instance	LB_f	$z(IP)$	LB^f_{root}	$time$	$time_{prc}$	$nodes$
X-n670-66-k117	144,839	144,882	144,813	–	165,461	183
X-n670-80-k120	145,035	–	145,010	–	202,974	21
X-n685-50-k43	47,407	–	47,342	–	170,246	53
X-n685-66-k54	52,587	–	52,579	–	204,370	25
X-n685-80-k62	58,596	–	58,585	–	203,097	39
X-n701-50-k23	50,681	–	50,213	–	192,061	27
X-n701-66-k30	58,062	–	58,052	–	198,252	11
X-n701-80-k36	67,740	–	67,729	–	199,734	7
X-n716-50-k18	29,173	–	29,163	–	191,038	5
X-n716-66-k23	31,916	–	31,916	–	195,583	1
X-n716-80-k28	37,349	–	37,345	–	208,194	3
X-n733-50-k83	79,844	–	79,781	–	112,174	183
X-n733-66-k102	91,742	–	91,693	–	131,939	119
X-n733-80-k125	110,192	–	110,066	–	144,296	131
X-n749-50-k49	47,096	–	47,049	–	164,098	105
X-n749-66-k63	54,774	–	54,757	–	196,229	55
X-n749-80-k78	63,201	–	63,184	–	201,468	49
X-n766-50-k58	94,883	–	94,883	–	179,499	1
X-n766-66-k62	100,629	–	100,609	–	195,474	11
X-n766-80-k65	105,601	–	105,599	–	201,160	15
X-n783-50-k24	47,790	–	47,767	–	194,547	19
X-n783-66-k31	52,525	–	52,525	–	191,591	1
X-n783-80-k38	59,840	–	59,840	–	212,236	1
X-n801-50-k20	47,967	–	47,946	–	159,575	7
X-n801-66-k27	54,024	–	54,024	–	173,590	5
X-n801-80-k32	61,987	–	61,987	–	172,834	3
X-n819-50-k86	88,189	–	88,113	–	126,840	95
X-n819-66-k112	107,696	–	107,642	–	112,531	107
X-n819-80-k136	127,889	–	127,837	–	111,957	165
X-n837-50-k71	115,003	–	114,946	–	112,370	87
X-n837-66-k94	128,204	–	128,137	–	170,597	71
X-n837-80-k114	154,027	–	154,010	–	170,493	133
X-n856-50-k48	57,518	–	57,447	–	142,210	27
X-n856-66-k63	63,213	–	63,170	–	170,993	61
X-n856-80-k76	73,510	–	73,477	–	190,204	71
X-n876-50-k30	57,908	–	57,894	–	187,424	25
X-n876-66-k38	68,494	–	68,484	–	198,796	11
X-n876-80-k46	80,098	–	80,093	–	209,728	3
X-n895-50-k19	39,754	–	39,738	–	176,623	3
X-n895-66-k25	42,952	–	42,952	–	211,581	1
X-n895-80-k30	47,389	–	47,389	–	167,800	1
X-n916-50-k105	187,565	–	187,501	–	92,046	119
X-n916-66-k136	221,521	–	221,445	–	89,710	99
X-n916-80-k165	262,715	–	262,631	–	101,187	71
X-n936-50-k132	127,340	–	127,287	–	174,810	5
X-n936-66-k138	128,306	–	128,138	–	205,809	15
X-n936-80-k143	129,904	–	129,895	–	204,086	39
X-n957-50-k44	56,298	–	56,213	–	156,715	23
X-n957-66-k58	62,060	–	62,049	–	194,368	29
X-n957-80-k70	71,312	–	71,282	–	203,391	19
X-n979-50-k30	68,030	–	68,016	–	182,505	2
X-n979-66-k39	74,118	–	74,118	–	209,674	1

(*Continues on the next page*)

Instance	LB_f	$z(IP)$	LB^f_{root}	$time$	$time_{prc}$	$nodes$
X-n979-80-k47[a]	–	–	–	–	–	1
X-n1001-50-k22	48,896	–	48,896	–	174,534	1
X-n1001-66-k28	55,122	–	55,122	–	162,503	1
X-n1001-80-k34	62,131	–	62,131	–	180,291	1

[a]Not even the first lower bound of the root node has been solved

APPENDIX H – VRPSolver models

VRPSolver is a framework for building BCP algorithms for VRP and related problems, available at `vrpsolver.math.u-bordeaux.fr`. It was used to implement our proposed VRPB algorithms, BCP$_{\mathcal{F}1}$ and BCP$_{\mathcal{F}2}$. We present here the VRPSolver models used. This appendix is not self-contained, important concepts used in these models, such as main resources, packing sets, mapping between variables and arcs, and Rounded Capacity Cuts (RCC) separators, are discussed in Pessoa et al. [92]. The parameterization of the solver for all problems and formulations is the same: $\tau^{\text{soft}} = 5$ sec., $\tau^{\text{hard}} = 10$ sec., $\phi^{\text{bidir}} = 1$, $\omega^{\text{labels}} = 2 \cdot 10^5$, $\omega^{\text{routes}} = 2 \cdot 10^6$, $\eta^{\max} = 20$, $\delta^{\text{gap}} = 1.5\%$, $\zeta_1^{\text{num}} = 50$, $\zeta_1^{\text{estim}} = 1.0$. The meaning of these parameters is also explained in Pessoa et al. [92].

H.1 Formulation $\mathcal{F}1$

We first give the model corresponding to formulation $\mathcal{F}1$ for the VRPB. The RCSP graph is exactly the graph $\mathcal{G} = (\mathcal{V}, \mathcal{A}) = (V, A) = G$ together with the consumption and intervals defined in Section 4.1.1 ; $v_{source} = v_{sink} = 0$. The capacity resource is defined as a main resource. Define an integer variable x_a for each $a \in A$ (exactly the same arc variables defined in formulation $\mathcal{F}0$). The formulation is:

$$\text{Min} \sum_{a \in A} c_a x_a \tag{H.1}$$

$$\text{S.t.} \sum_{a \in \delta^-(i)} x_a = 1, \quad i \in \bar{V}. \tag{H.2}$$

The number of paths in the solution is fixed to K ($L = U = K$). Each variable x_a is mapped to arc a ($M(x_a) = \{a\}$, $a \in A$). Packing sets are defined on vertices $\mathcal{B}^{\mathcal{V}} = \cup_{i \in \bar{V}} \{\{i\}\}$. There are two RCC separators, the first is defined on $(\cup_{i \in L} \{(\{i\}, d_i)\}, Q)$, and the second is defined on $(\cup_{i \in B} \{(\{i\}, d_i)\}, Q)$. Branching is performed on the aggregation of x variables corresponding to opposite arcs. Enumeration is activated.

To adapt the model to the VRPBTW, we add a second main resource corresponding to the time, so $R = R_M = \{1, 2\}$. The arc resource consumption for the second resource equals the travelling time plus the service time: $q_{a=(i,j),2} = c_{ij} + s_i$, $a \in A$, where $s_0 = 0$. The resource consumption intervals for the second resource are equal to customer time windows (or to the time horizon for the depot). Otherwise, the model is the same as for the VRPB.

The model for the HFFVRPB, considering a set T of vehicle types, is the following. We define graphs $\mathcal{G}^k = (\mathcal{V}^k, \mathcal{A}^k)$, $k \in T$, all of them isomorphic to G. However, the intervals are defined using the capacity Q^k of each type. We denote vertex i in graph $\mathcal{G}^k$ as i^k. Define $\delta^-(i^k)$ as the set of arcs in $\mathcal{A}^k$ entering i^k. Define an integer variable x_a^k per vehicle type $k \in T$ and per arc $a \in A^k$. The formulation is:

$$\text{Min} \sum_{k \in T} \sum_{a \in A^k} c_a^k x_a^k \tag{H.3}$$

$$\text{S.t.} \sum_{k \in T} \sum_{e \in \delta^-(i^k)} x_a^k = 1, \quad i \in \bar{V}. \tag{H.4}$$

where c_a^k are the type dependent costs. The bounds for the number of paths from graph $\mathcal{G}^k$, $k \in T$, in the solution are $[0, U^k]$. Each variable $x^k_{a=(i^k,j^k)}$ is mapped to arc (i^k, j^k). Packing sets are defined on vertices $\mathcal{B}^{\mathcal{V}} = \cup_{i \in \bar{V}} \{\{i^k : k \in T\}\}$. We have two RCC separators, the first is defined on $(\cup_{i \in L} \{(\{i^k : k \in T\}, d_i)\}, \max_{k \in T} Q^k)$, and the second is defined on $(\cup_{i \in B} \{(\{i^k : k \in T\}, d_i)\}, \max_{k \in T} Q^k)$. Branching is performed on variable expressions: i) on the number of used vehicles of each type $\sum_{i \in L} x^k_{(0,i)}$, $k \in T$; ii) on assignment of customers to vehicle types $\sum_{a \in \delta^-(i^k)} x_a^k$, $k \in T$, $i \in L \cup B$; iii) on aggregated edges $\sum_{k \in T} x^k_{(i^k,j^k)} + x^k_{(j^k,i^k)}$, $i, j \in V$, $i < j$.

H.2 Formulation $\mathcal{F}2$

We now give the model corresponding to formulation $\mathcal{F}2$ for the VRPB. It is less direct than the model for $\mathcal{F}1$, some tricks are needed to apply VRPSolver to this case.

There are two RCSP graphs. The backhaul graph $\mathcal{G}_B = (\mathcal{V}_B, \mathcal{A}_B)$ is exactly the one defined in Section 4.1.2. However, the linehaul graph $\mathcal{G}'_L = (\mathcal{V}'_L, \mathcal{A}'_L)$ is a bit different: $\mathcal{V}'_L = L_0 \cup \{i' : i \in L_0\}$ and $\mathcal{A}'_L = A_L \cup \{(i, i') : i \in L\} \cup \{(i', 0') : i \in L\}$; $v_{source} = 0$ and $v_{sink} = 0'$. Each arc $a = (i, j) \in A_L$ has a capacity resource consumption given by $q_a = d_j$, the other arcs in $\mathcal{A}'_L$ have zero consumption. Each vertex $i \in \mathcal{V}'_L$ has resource interval $[0, Q]$. The additional copies of the linehaul vertices in $\mathcal{G}'_L$ are introduced in order to be able to use path enumeration in it. Without them, the necessary condition to use enumeration defined in Pessoa et al. [92] would not be satisfied.

Define an integer variable x_a for each $a \in A$ (again, exactly the same arc variables defined in formulation $\mathcal{F}0$). In addition, there are two integer variables z_i, w_i for every linehaul customer $i \in L$. The formulation is:

$$\text{Min} \sum_{a \in A} c_a x_a \tag{H.5}$$

$$\text{S.t.} \sum_{a \in \delta^-(i)} x_a = 1, \quad i \in \bar{V}, \tag{H.6}$$

$$z_i = w_i, \quad i \in L. \tag{H.7}$$

The number of paths from both $\mathcal{G}'_L$ and $\mathcal{G}_B$ in the solution is fixed to K. Each variable x_a, $a = (i,j) \in A_L$, is mapped to arc (i,j) in $\mathcal{A}'_L$. Each variable x_a, $a = (i,j) \in A_{LB}$, is mapped to arc (i',j) in $\mathcal{A}'_B$. Each variable x_a, $a = (i,j) \in A_B$, is mapped to arc (i,j) in $\mathcal{A}'_B$. A variable z_i, $i \in L$, is mapped to arc (i,i') in $\mathcal{A}'_L$. Finally, variables w_i, $i \in L$, is mapped to arc $(0',i')$ in $\mathcal{A}'_B$. Packing sets are defined on vertices, one packing set is defined for each vertex in the both graphs, except for the depot vertices. Branching is performed on the aggregation of x variables corresponding to opposite arcs and z variables. Enumeration is activated. Figure H.1 illustrates RCSP graphs $\mathcal{G}_L$ and $\mathcal{G}_B$, the consumptions and variables mapped to each arc are also depicted.

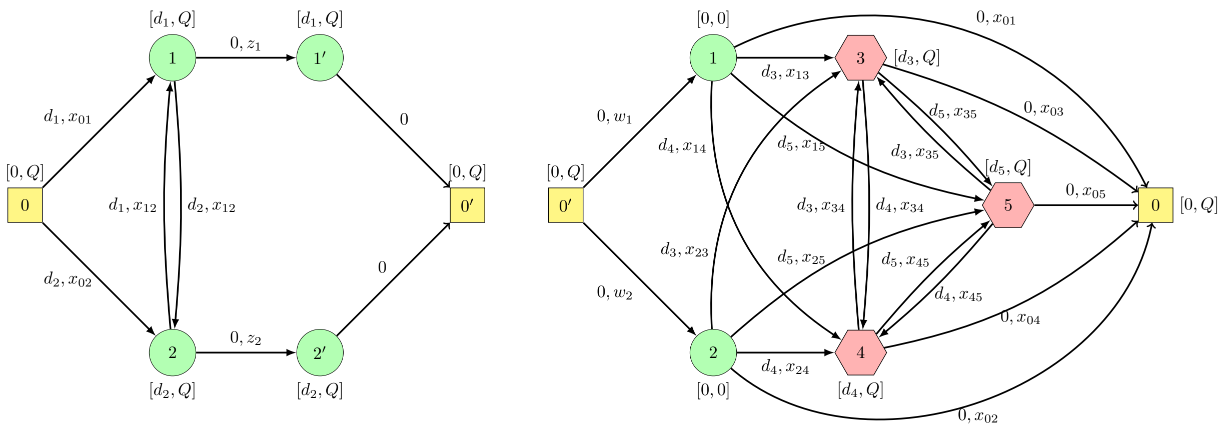


Figure H.1: RCSP graphs for the VRPSolver model of $\mathcal{F}2$

.

The Julia code corresponding to the above VRPB model is available on the VRP-Solver webpage `vrpsolver.math.u-bordeaux.fr`.

APPENDIX I – Comparing $\mathcal{F}1$, $\mathcal{F}2$ and Mingozzi, Giorgi, and Baldacci [82] formulations

We first present the SP formulation by Mingozzi, Giorgi, and Baldacci [82]. Although that formulation was originally defined with elementary routes, here we will compare all formulations as using ng-paths defined over the same ng-sets. This is more general, as elementary routes correspond to the case where ng-sets contain all vertices.

Let $G'_B = (B_0, A_B)$, Ω'_B be the set of all ng-paths over G'_B, and $\Omega'^i_B \subseteq \Omega'_B$ be the set of ng-paths in Ω'_B starting at $i \in B$. Let y_p be a binary variable that assumes 1 if $p \in \Omega'_L \cup \Omega'_B$ is chosen, 0 otherwise. Let ξ_{ij} be a binary variable that assumes 1 if the arc $(i, j) \in A_{LB}$ is in the optimal solution, 0 otherwise.

The SP formulation by Mingozzi, Giorgi, and Baldacci [82], which we will denote by $\mathcal{M}$, can be written as follows:

$$\min \sum_{p \in \Omega_L} \left(\sum_{a \in A} c_a h_a^p \right) y_p + \sum_{p \in \Omega'_B} \left(\sum_{a \in A} c_a h_a^p \right) y_p + \sum_{(i,j) \in A_{LB}} c_{ij} \xi_{ij} \tag{I.1}$$

$$S.t. \sum_{a\in\delta^-(i)} \sum_{p\in\Omega_L} h_a^p y_p = 1 \qquad i \in L, \tag{I.2}$$

$$\sum_{a\in\delta^-(j)} \sum_{p\in\Omega'_B} h_a^p y_p = 1 \qquad j \in B, \tag{I.3}$$

$$\sum_{a\in\delta^+(0)} \sum_{p\in\Omega_L} h_a^p y_p = K, \tag{I.4}$$

$$\sum_{p\in\Omega_L^i} y_p = \sum_{j\in B_0} \xi_{ij} \qquad i \in L, \tag{I.5}$$

$$\sum_{p\in\Omega_B'^j} y_p = \sum_{i\in L} \xi_{ij} \qquad j \in B, \tag{I.6}$$

$$y_p \in \{0,1\} \qquad p \in \Omega_L \cup \Omega'_B, \tag{I.7}$$

$$\xi_{ij} \in \{0,1\} \qquad (i,j) \in A_{LB}. \tag{I.8}$$

Constraints (I.2) require that each linehaul customer must be visited once by a linehaul route. Constraints (I.3) require that each backhaul customer must be visited once by a backhaul route. Constraint (I.4) forces the existence of K linehaul routes in the solution. Constraints (I.5)–(I.6) force linehaul and backhaul routes to be connected by an arc. Finally, the domain of the variables are defined in constraints (I.7)–(I.8).

In the following proof we consider $\mathcal{F}1$ and $\mathcal{F}2$ without rounded capacity cuts, (13-14) and (24-25) respectively, since $\mathcal{M}$ was proposed without such cuts.

Proposition 3. $\mathcal{F}1$, $\mathcal{F}2$ (both without rounded capacity cuts) and $\mathcal{M}$ formulations are equally strong.

Proof. It is sufficient to prove that $\mathcal{F}1$ and $\mathcal{M}$ are equally strong because the equivalence between $\mathcal{F}1$ and $\mathcal{F}2$ was already proved in Section 3.4.

Let P_1 and P_M be the polyhedra defined by the linear relaxations of $\mathcal{F}1$ and $\mathcal{M}$, respectively. We show that for any solution of P_1 there is a solution of P_M with the same objective value, and vice versa.

Given a solution $\bar{\lambda} \in P_1$, the function described in Algorithm 3 returns a solution $\mathtt{P}_M(\bar{\lambda}) = (\bar{y}, \bar{\xi})$ in P_M. It is clear from lines 7–10 that constraints (I.5) and (I.6) are satisfied by that solution. One can verify that $\mathtt{P}_M(\bar{\lambda})$ also satisfies the constraints (I.2)–(I.4) and has the same cost as $\bar{\lambda}$ (during the algorithm, all cost is "transferred" from $\bar{\lambda}$ to $\mathtt{P}_M(\bar{\lambda})$).

Given a solution $(\bar{y}, \bar{\xi}) \in P_M$, the function described in Algorithm 4 returns a solution $\mathtt{P}_1(\bar{\lambda}) = \bar{\lambda}$ in $\mathcal{F}1$ space. The procedure is very similar to Algorithm 2, where the

Algorithm 3: Obtains the solution $(\bar{y}, \bar{\xi}) \in P_M$ corresponding to $\bar{\lambda} \in P_1$

Function $\mathtt{P}_M(\bar{\boldsymbol{\lambda}})$
  Let $\gamma = \{(p, \bar{\lambda}_p) : p \in \Omega,\ \bar{\lambda}_p > 0\}$ be the set that maps the routes to their values
  Let $L(p) \in \Omega_L$ and $B(p) \in \Omega'_B$ be the ng-paths obtained by splitting route $p \in \Omega$ into linehaul and backhaul parts, respectively.
  Let $(\bar{y}, \bar{\xi})$ be the solution to be built for P_M, such that $\bar{y}_p$ is initially zero $\forall p \in \Omega_L \cup \Omega'_B$ and $\bar{\xi}_{ij}$ is initially zero $\forall (i, j) \in A_{LB}$
  while $\gamma \neq \emptyset$ **do**
    Let (p, ζ) be a pair in γ
    $\bar{y}_{L(p)} = \bar{y}_{L(p)} + \zeta$
    $\bar{y}_{B(p)} = \bar{y}_{B(p)} + \zeta$
    Let (i, j) be the arc connecting $L(p)$ with $B(p)$ in p
    $\bar{\xi}_{ij} = \bar{\xi}_{ij} + \zeta$
    $\gamma = \gamma \setminus \{(p, \zeta)\}$ `// Remove` p
  return $(\bar{\lambda}^L, \bar{\lambda}^B)$

minimum value variable is iteratively selected to build routes in Ω. Hence, one can verify that $\mathtt{P}_1(\bar{\lambda})$ satisfies all the constraints and has the same cost as $(\bar{y}, \bar{\xi})$.

□

Although formulations $\mathcal{F}2$ and $\mathcal{M}$ are quite similar and are equally strong, $\mathcal{F}2$ avoids a quadratic number of variables by incorporating the cost of arcs between linehaul and backhaul customers in the cost of ng-paths in Ω_B. Sets Ω_B has $|L|$ times more ng-paths than Ω'_B, but, since path variables are dynamically priced, this is not a significant drawback.

Algorithm 4: Obtains the solution $\bar{\lambda} \in P_1$ corresponding to $(\bar{y}, \bar{\xi}) \in P_M$

Function P₁ $(\bar{\boldsymbol{y}}, \bar{\boldsymbol{\xi}})$
  Let $\gamma = \{(p, \bar{y}_p) : p \in \Omega_L \cup \Omega'_B,\ \bar{y}_p > 0\} \cup \{((i,j), \bar{\xi}_{ij}) : (i,j) \in A_{LB},\ \bar{\xi}_{ij} > 0\}$, be the sets that maps the ng-paths (which can be just one arc) to their values
  Let $p_l \oplus p_b$ be the route in Ω obtained by concatenating the paths $p_l \in \Omega_L$ and $p_b \in \Omega'_B$
  Let $\bar{\lambda}$ be the solution to be built for P_1, such that $\bar{\lambda}_p$ is initially zero $\forall p \in \Omega$
  while $\gamma \neq \emptyset$ **do**
    Let (p^1, ζ^1) be a pair in γ whose ζ^1 is minimum
    if $p^1 \in \Omega_L$ **then**
      Let $l \in L$ be the last vertex in p^1
      Let (p^2, ζ^2) be any pair in γ such that $p^2 = (l,k) \in \{(l,j) : j \in B_0\}$
      if $k \in B$ **then**
        Let (p^3, ζ^3) be any pair in γ such that $p^3 \in \Omega'^k_B$
        $\bar{\lambda}_p = \zeta^1$, such that $p = p^1 \oplus p^3$
      else `//` ng`-path with only linehaul customers`
        $\bar{\lambda}_p = \zeta^1$, such that $p = p^1 \oplus 0$ `// just add the depot to` p^1
    else if $p^1 = (i,j) \in A_{LB}$ **then**
      Let (p^2, ζ^2) be any pair in γ such that $p^2 \in \Omega^i_L$
      if $j \in B$ **then**
        Let (p^3, ζ^3) be any pair in γ such that $p^3 \in \Omega'^j_B$
        $\bar{\lambda}_p = \zeta^1$, such that $p = p^2 \oplus p^3$
      else
        $\bar{\lambda}_p = \zeta^1$, such that $p = p^2 \oplus 0$
    else `//` $p^1 \in \Omega'_B$
      Let $b \in B$ be the first vertex in p^1
      Let (p^2, ζ^2) be any pair in γ such that $p^2 = (k,b) \in \{(i,b) : i \in L\}$
      Let (p^3, ζ^3) be any pair in γ such that $p^3 \in \Omega^k_L$
      $\bar{\lambda}_p = \zeta^1$, such that $p = p^3 \oplus p^1$
    $\gamma = \gamma \setminus \{(p^1, \zeta^1), (p^2, \zeta^2)\}$ `// Remove` p^1, p^2
    if $\zeta^2 - \zeta^1 > 0$ **then**
      $\gamma = \gamma \cup \{(p^2, \zeta^2 - \zeta^1)\}$ `// Reinsert` p^2 `with updated value`
    if (p^3, ζ^3) *was defined* **then**
      $\gamma = \gamma \setminus \{(p^3, \zeta^3)\}$ `// Remove` p^3
      if $\zeta^3 - \zeta^1 > 0$ **then**
        $\gamma = \gamma \cup \{(p^3, \zeta^3 - \zeta^1)\}$ `// Reinsert` p^3 `with updated value`
  return $\bar{\lambda}$

APPENDIX J – Detailed results of the heuristic approaches for VRPB

J.1 Comparison with the literature

Table J.1: Detailed information on works considered for comparison.

Method	Machine	#Runs	Benchmarks
MACS	Intel Xeon 2.4 GHz	8	TV, GJB (up to 150 vertices)
ILS-1000	Intel Core i7 2.93 GHz	10	TV, GJB
UHGS	Opteron 250 2.4 GHz	10	GJB (up to 150 vertices)
SISRs	Xeon E5-2650 v2 CPU 2.60 GHz	10	GJB (up to 150 vertices)
Our ILS-SP approaches	Intel Xeon E5-2680 2.50 GHz	50	TV, GJB

Table J.2: Detailed results of the best heuristics for the GJB instances. The costs were divided by 10^3 and the CPU times (in seconds) were scaled to the machine of Cuervo et al. [32] for a fair comparison. For ILS-1000, Avg. and CPU were not reported by the authors.

			MACS			ILS-1000	UHGS			SISRs			ILS-SP			ILS$_B$-SP			ILS$_B$-SP$_B$		
Instance	$n+m$	K	Best	Avg.	CPU	Best	Best	Avg.	CPU	Best	Avg.	CPU	Best	Avg.	CPU	Best	Avg.	CPU	Best	Avg.	CPU
A1	25	8	229.89	229.89	0.6	229.89	229.89	229.89	0	229.89	229.89	0.4	229.89	229.89	0.1	229.89	229.89	0.1	229.89	229.89	0.1
A2	25	5	180.12	180.12	1	180.12	180.12	180.12	5.8	180.12	180.12	0.4	180.12	180.12	0.1	180.12	180.12	0.1	180.12	180.12	0.1
A3	25	4	163.41	163.41	1.7	163.41	163.41	163.41	6.3	163.41	163.41	0.4	163.41	163.41	0.1	163.41	163.41	0.1	163.41	163.41	0.1
A4	25	3	155.80	155.80	1	155.80	155.80	155.80	7.8	155.80	155.80	0.4	155.80	155.80	0.1	155.80	155.80	0.1	155.80	155.80	0.1
B1	30	7	239.08	239.08	1.2	239.08	239.08	239.08	6.8	239.08	239.08	0.5	239.08	239.08	0.1	239.08	239.08	0.2	239.08	239.08	0.2
B2	30	5	198.05	198.05	1.4	198.05	198.05	198.05	7.3	198.05	198.05	0.5	198.05	198.05	0.2	198.05	198.05	0.1	198.05	198.05	0.1
B3	30	3	169.37	169.37	1.1	169.37	169.37	169.37	8.7	169.37	169.37	0.5	169.37	169.37	0.1	169.37	169.37	0.1	169.37	169.37	0.1
C1	40	7	250.56	250.56	2.1	250.56	250.56	250.56	10.7	250.56	250.56	0.7	250.56	250.56	0.3	250.56	250.56	0.3	250.56	250.56	0.3
C2	40	5	215.02	215.02	2.3	215.02	215.02	215.02	11.6	215.02	215.02	0.7	215.02	215.02	0.3	215.02	215.02	0.2	215.02	215.02	0.3
C3	40	5	199.35	199.35	2.7	199.35	199.35	199.35	11.1	199.35	199.35	0.8	199.35	199.35	0.4	199.35	199.35	0.3	199.35	199.35	0.3
C4	40	4	195.37	195.37	2.1	195.37	195.37	195.37	11.6	195.37	195.37	0.8	195.37	195.37	0.4	195.37	195.37	0.2	195.37	195.37	0.2
D1	38	12	322.53	322.53	3.4	322.53	322.53	322.53	8.7	322.53	322.53	0.7	322.53	322.53	0.3	322.53	322.53	0.4	322.53	322.53	0.4
D2	38	11	316.71	316.71	3.5	316.71	316.71	316.71	8.2	316.71	316.71	0.7	316.71	316.71	0.3	316.71	316.71	0.3	316.71	316.71	0.3
D3	38	7	239.48	239.48	3.1	239.48	239.48	239.48	9.2	239.48	239.48	0.6	239.48	239.48	0.3	239.48	239.48	0.3	239.48	239.48	0.3
D4	38	5	205.83	205.83	3.6	205.83	205.83	205.83	11.6	205.83	205.83	0.7	205.83	205.83	0.3	205.83	205.83	0.3	205.83	205.83	0.3
E1	45	7	238.88	238.88	3.7	238.88	238.88	238.88	13.1	238.88	238.88	0.8	238.88	238.88	0.3	238.88	238.88	0.3	238.88	238.88	0.3
E2	45	4	212.26	212.26	3.6	212.26	212.26	212.26	15.5	212.26	212.26	0.8	212.26	212.26	0.5	212.26	212.26	0.4	212.26	212.26	0.4
E3	45	4	206.66	206.66	5.7	206.66	206.66	206.66	17.5	206.66	206.66	0.8	206.66	206.66	0.6	206.66	206.66	0.4	206.66	206.66	0.5
F1	60	6	263.17	263.17	6.2	263.17	263.17	263.17	18.4	263.17	263.17	1.2	263.17	263.17	0.9	263.17	263.17	0.8	263.17	263.17	0.8
F2	60	7	265.21	265.21	5.1	265.21	265.21	265.21	18.9	265.21	265.21	1.3	265.21	265.21	1	265.21	265.21	0.8	265.21	265.21	0.8
F3	60	5	241.12	241.48	6.2	241.12	241.12	241.12	23.8	241.12	241.12	1.4	241.12	241.12	1.1	241.12	241.12	0.7	241.12	241.12	0.8
F4	60	4	233.86	233.86	8.3	233.86	233.86	233.86	26.2	233.86	233.86	1.4	233.86	233.86	1.2	233.86	233.86	0.8	233.86	233.86	0.8
G1	57	10	306.31	307.01	10	306.31	–	–	–	306.31	306.31	1.1	306.31	306.31	0.9	306.31	306.31	1	306.31	306.31	1.0
G2	57	6	245.44	245.44	5.7	245.44	245.44	245.44	18.4	245.44	245.44	1.1	245.44	245.44	0.8	245.44	245.44	0.7	245.44	245.44	0.7
G3	57	5	229.51	229.51	7.9	229.51	229.51	229.51	20.8	229.51	229.51	1.1	229.51	229.51	0.8	229.51	229.51	0.7	229.51	229.51	0.7
G4	57	6	232.52	232.52	12	232.52	232.52	232.52	21.8	232.52	232.52	1.2	232.52	232.52	0.8	232.52	232.52	0.8	232.52	232.52	0.8
G5	57	5	221.73	221.73	11.3	221.73	221.73	221.73	22.3	221.73	221.73	1.3	221.73	221.73	1	221.73	221.73	0.8	221.73	221.73	0.8
G6	57	4	213.46	213.46	11.4	213.46	213.46	213.46	26.2	213.46	213.46	1.4	213.46	213.46	1.1	213.46	213.46	0.9	213.46	213.46	0.9
H1	68	6	268.93	269.00	13.6	268.93	268.93	268.93	30.1	268.93	268.93	1.4	268.93	268.93	1.7	268.93	268.93	1.4	268.93	268.93	1.4
H2	68	5	253.37	253.37	11.9	253.37	253.37	253.37	27.6	253.37	253.37	1.5	253.37	253.37	1.8	253.37	253.37	1.3	253.37	253.37	1.3

(*Continues on the next page*)

			MACS			ILS-1000	UHGS			SISRs			ILS-SP			ILS_B-SP			ILS_B-SP_B		
Instance	$n+m$	K	Best	Avg.	CPU	Best	Best	Avg.	CPU	Best	Avg.	CPU	Best	Avg.	CPU	Best	Avg.	CPU	Best	Avg.	CPU
H3	68	4	247.45	247.45	11.2	247.45	247.45	247.45	31	247.45	247.45	1.7	247.45	247.45	2	247.45	247.45	1.3	247.45	247.45	1.4
H4	68	5	250.22	250.22	15	250.22	250.22	250.22	28.6	250.22	250.22	1.7	250.22	250.22	1.9	250.22	250.22	1.4	250.22	250.22	1.5
H5	68	4	246.12	246.12	14.4	246.12	246.12	246.12	30.1	246.12	246.12	1.7	246.12	246.12	2.2	246.12	246.12	1.4	246.12	246.12	1.4
H6	68	5	249.14	249.14	16.7	249.14	249.14	249.14	28.6	249.14	249.14	1.7	249.14	249.14	2.3	249.14	249.14	1.5	249.14	249.14	1.5
I1	90	10	350.25	350.40	23	350.25	350.25	350.37	43.1	350.25	350.25	2.3	350.25	350.25	3.7	350.25	350.25	3.3	350.25	350.25	3.8
I2	90	7	309.94	310.32	20.7	309.94	309.94	309.94	41.2	309.94	309.94	2.4	309.94	309.94	3.4	309.94	309.94	2.5	309.94	309.94	2.6
I3	90	5	294.51	294.84	24.3	294.51	294.51	294.51	48	294.51	294.51	2.5	294.51	294.51	4.2	294.51	294.51	2.8	294.51	294.51	2.9
I4	90	6	295.99	296.13	25.8	295.99	295.99	295.99	44.6	295.99	295.99	2.8	295.99	295.99	4.3	295.99	295.99	2.6	295.99	295.99	2.7
I5	90	7	301.24	301.83	26.2	301.24	301.24	301.24	39.7	301.24	301.24	2.9	301.24	301.24	4.3	301.24	301.24	2.6	301.24	301.24	2.7
J1	94	10	335.01	335.12	36.9	335.01	335.01	335.01	40.2	335.01	335.01	2.3	335.01	335.01	3.3	335.01	335.01	3.1	335.01	335.01	3.1
J2	94	8	310.42	310.42	32.8	310.42	310.42	310.42	40.7	310.42	310.42	2.6	310.42	310.42	5	310.42	310.42	3.7	310.42	310.42	4.2
J3	94	6	279.22	279.34	42.9	279.22	279.22	279.22	45.1	279.22	279.22	2.7	279.22	279.22	4.5	279.22	279.22	3.4	279.22	279.22	3.6
J4	94	7	296.53	296.58	34.7	296.53	296.53	296.53	54.3	296.53	296.53	2.7	296.53	296.54	4.9	296.53	296.53	4.1	296.53	296.53	4.8
K1	113	10	394.07	396.14	58	394.07	394.07	394.35	64.5	394.07	394.09	3.4	394.07	394.07	10	394.07	394.07	9.7	394.07	394.07	8.1
K2	113	8	362.13	362.56	53.6	362.13	362.13	362.13	67.9	362.13	362.13	3.5	362.13	362.13	7.7	362.13	362.13	5.6	362.13	362.13	5.7
K3	113	9	365.69	366.71	63.2	365.69	365.69	365.69	63	365.69	365.69	3.7	365.69	365.69	7.5	365.69	365.69	5.8	365.69	365.69	6.0
K4	113	7	348.95	350.32	58.6	348.95	348.95	348.95	62.5	348.95	348.95	4.0	348.95	348.95	8.1	348.95	348.95	5.5	348.95	348.95	5.8
L1	150	10	417.90	420.06	82.1	417.90	417.90	418.16	191	417.90	417.90	6.1	417.90	417.90	19.2	417.90	417.90	13.6	417.90	417.90	14.2
L2	150	8	401.23	401.36	78.6	401.23	401.23	401.23	143.5	401.23	401.23	6.1	401.23	401.23	20.3	401.23	401.23	15	401.23	401.23	16.3
L3	150	9	402.68	404.32	88.9	402.68	402.68	402.68	132.3	402.68	402.68	6.2	402.68	402.68	17.9	402.68	402.68	11.4	402.68	402.68	12.6
L4	150	7	384.64	384.83	88.2	384.64	384.64	384.64	113.9	384.64	384.64	6.9	384.64	384.64	19.6	384.64	384.64	11.7	384.64	384.64	12.2
L5	150	8	387.56	390.33	87.4	387.56	387.56	387.56	131.8	387.56	387.56	6.9	387.56	387.57	20.6	387.56	387.56	12.1	387.56	387.58	12.5
M1	125	11	398.59	399.12	127	398.59	398.59	398.66	76.6	398.59	398.59	3.9	398.59	398.80	21.4	398.59	398.84	22.3	398.59	398.77	20.2
M2	125	10	396.92	398.16	101.4	396.92	396.92	396.93	115.9	396.92	397.24	3.7	396.92	397.09	64.7	396.92	397.08	68.3	396.92	397.05	48.5
M3	125	9	375.70	377.81	101.6	375.70	375.70	375.93	130.9	375.70	375.76	4.5	375.70	376.18	28.7	375.70	375.96	21.4	375.70	375.93	29.2
M4	125	7	348.14	348.46	90.8	348.14	348.14	348.20	83.4	348.14	348.31	4.8	348.14	348.14	13.3	348.14	348.14	10.2	348.14	348.14	11.6
N1	150	11	408.10	408.17	118.3	408.10	408.10	408.10	131.8	408.10	408.10	6.1	408.10	408.10	22.4	408.10	408.10	18.2	408.10	408.10	19.6
N2	150	10	408.07	408.25	118.6	408.07	408.07	408.13	123.6	408.07	408.07	5.9	408.07	408.07	24.8	408.07	408.07	21.1	408.07	408.07	20.4
N3	150	9	394.34	394.70	110.5	394.34	394.34	394.94	119.2	394.34	394.34	6.9	394.34	394.34	20.3	394.34	394.34	14.3	394.34	394.34	15.2
N4	150	10	394.79	394.87	121.1	394.79	394.79	395.13	114.9	394.79	394.79	6.9	394.79	394.79	22	394.79	394.79	16.2	394.79	394.79	17.1
N5	150	7	373.48	374.12	150.5	373.48	373.48	373.55	151.2	373.48	373.48	6.8	373.48	373.52	20.8	373.48	373.56	14.5	373.48	373.49	16.5
N6	150	8	373.76	374.79	141.1	373.76	373.76	373.76	165.8	373.76	373.76	6.5	373.76	373.76	22.2	373.76	373.76	15.6	373.76	373.76	17.3
O1	200	10	–	–	–	478.35	–	–	–	–	–	–	**478.13**	478.95	40.2	**478.13**	478.88	27.9	**478.13**	478.38	52.5

(*Continues on the next page*)

Instance	$n+m$	K	MACS Best	MACS Avg.	MACS CPU	ILS-1000 Best	UHGS Best	UHGS Avg.	UHGS CPU	SISRs Best	SISRs Avg.	SISRs CPU	ILS-SP Best	ILS-SP Avg.	ILS-SP CPU	ILS_B-SP Best	ILS_B-SP Avg.	ILS_B-SP CPU	ILS_B-SP_B Best	ILS_B-SP_B Avg.	ILS_B-SP_B CPU
O2	200	11	–	–	–	477.26	–	–	–	–	–	–	477.26	477.26	39.1	477.26	477.26	25.7	477.26	477.26	29.2
O3	200	9	–	–	–	457.29	–	–	–	–	–	–	457.29	458.55	42	457.29	458.62	28.1	457.29	457.75	30.2
O4	200	10	–	–	–	458.87	–	–	–	–	–	–	458.87	459.98	40.7	458.87	460.44	26	458.87	459.19	29.1
O5	200	7	–	–	–	436.97	–	–	–	–	–	–	436.97	437.21	47.7	436.97	437.16	28	436.97	437.02	31.2
O6	200	8	–	–	–	438.00	–	–	–	–	–	–	438.00	438.47	46.8	438.00	438.31	27.5	438.00	438.09	30.6
Average (up to N6)					37.4				51.2			**2.6**			10.5			7.7			8.3

Table J.3: New best solution of O1. The last linehaul customer of a route is highlighted in bold.

Cost	Routes
478,126.75	0 127 128 145 195 153 152 187 180 148 116 **168** 15 90 100 87 84 54 25 19 6 65 0
	0 138 190 112 113 129 146 123 **115** 32 80 24 52 12 58 27 70 0
	0 111 110 132 165 107 131 194 114 177 136 143 119 **124** 46 56 1 77 14 30 83 2 79 92 38 0
	0 166 151 159 192 122 117 125 191 175 118 **164** 99 64 50 13 61 3 73 4 7 75 0
	0 147 140 188 189 198 120 106 156 172 **200** 60 72 94 97 78 81 66 48 67 62 17 0
	0 184 163 158 197 149 178 186 170 **183** 68 89 11 8 51 43 33 63 39 0
	0 109 108 196 155 162 193 181 **199** 23 82 10 34 71 22 5 69 42 0
	0 134 174 182 104 169 137 144 **102** 37 96 91 26 18 57 41 76 44 74 93 0
	0 126 133 130 135 167 142 173 103 160 121 176 **154** 59 40 16 29 45 98 21 53 85 95 55 28 0
	0 150 139 179 101 157 141 171 185 161 **105** 9 20 36 47 49 86 35 88 31 0

Table J.4: Detailed results for the TV instances. CPU times (measured in seconds) were scaled to the machine of Cuervo et al. [32] for a fair comparison. For ILS-1000, Avg. and CPU were not reported by the authors.

				MACS			ILS-1000	ILS-SP			ILS_B-SP			ILS_B-SP_B		
Instance	BKS	$n+m$	K	Best	Avg.	CPU	Best	Best	Avg.	CPU	Best	Avg.	CPU	Best	Avg.	CPU
E-n22-50-k3	371	22	3	371	371	0.6	371	371	371	0.08	371	371	0.081	371	371	0.081
E-n22-66-k3	366	22	3	366	366	0.1	366	366	366	0.07	366	366	0.074	366	366	0.073
E-n22-80-k3	375	22	3	375	375	0.3	375	375	375	0.08	375	375	0.079	375	375	0.082
E-n23-50-k2	682	23	2	682	682	0.8	682	682	682	0.11	682	682	0.071	682	682	0.072
E-n23-66-k2	649	23	2	649	649	0.3	649	649	649	0.09	649	649	0.071	649	649	0.075
E-n23-80-k2	623	23	2	623	623	1.7	623	623	623	0.10	623	623	0.09	623	623	0.094
E-n30-50-k2	501	30	2	501	501	0.8	501	501	501	0.19	501	501	0.146	501	501	0.149
E-n30-66-k3	537	30	3	537	537	1.1	537	537	537	0.22	537	537	0.183	537	537	0.186
E-n30-80-k3	514	30	3	514	514	1.7	514	514	514	0.19	514	514	0.176	514	514	0.179
E-n33-50-k3	738	33	3	738	738	2.4	738	738	738	0.25	738	738	0.205	738	738	0.201
E-n33-66-k3	750	33	3	750	750	1.6	750	750	750	0.24	750	750	0.21	750	750	0.208
E-n33-80-k3	736	33	3	736	736	1.1	736	736	736	0.21	736	736	0.227	736	736	0.232
E-n51-50-k3	559	51	3	559	559	4.6	559	559	559	0.94	559	559	0.677	559	559	0.668
E-n51-66-k4	548	51	4	548	548	5.9	548	548	548	0.97	548	548	0.73	548	548	0.759
E-n51-80-k4	565	51	4	565	565	7.3	565	565	565	1.05	565	565	1.037	565	565	1.03
E-n76-A-50-k6	739	76	6	739	739.25	10.5	739	739	739	2.49	739	739	1.983	739	739	2.017
E-n76-A-66-k7	768	76	7	768	768	13.1	768	768	768	2.31	768	768	2.001	768	768	1.965
E-n76-A-80-k8	781	76	8	781	782.63	29.6	781	781	782.48	2.74	781	782	2.807	781	782.34	2.832
E-n76-B-50-k8	801	76	8	801	801	11.8	801	801	801	2.16	801	801	1.887	801	801	1.963
E-n76-B-66-k10	873	76	10	873	873.13	14.0	873	873	873	2.51	873	873	2.374	873	873	2.353
E-n76-B-80-k12	919	76	12	919	920.13	21.2	919	919	919	2.45	919	919	2.516	919	919	2.48
E-n76-C-50-k5	713	76	5	713	713	10.4	713	713	713	2.86	713	713	2.126	713	713	2.138
E-n76-C-66-k6	734	76	6	734	734	15.4	734	734	734.02	2.54	734	734.08	2.013	734	734.02	2.072
E-n76-C-80-k7	733	76	7	733	736.13	19.9	733	733	733.08	3.24	733	733.12	3.016	733	733.06	3.24
E-n76-D-50-k4	690	76	4	690	690	15.4	690	690	690	2.97	690	690	2.052	690	690	2.03
E-n76-D-66-k5	715	76	5	715	715	16.0	715	715	715	2.86	715	715	2.148	715	715	2.203
E-n76-D-80-k6	694	76	6	694	698.63	23.3	694	694	694.22	2.86	694	694	2.545	694	694.04	2.719
E-n101-A-50-k4	831	101	4	831	834.75	31.1	831	831	831.88	7.71	831	831	4.841	831	831.02	4.954
E-n101-A-66-k6	846	101	6	846	846	32.7	846	846	846	7.78	846	846	5.66	846	846	5.577
E-n101-A-80-k6	856	101	6	857	866.88	48.8	856	856	862.7	17.87	856	860.16	16.742	856	860.06	18.284
E-n101-B-50-k7	923	101	7	923	927.63	37.1	923	923	924.74	16.26	923	924.32	14.661	923	923.22	21.27
E-n101-B-66-k9	982	101	9	988	1,008.88	47.0	983	982	991.44	57.35	982	986.5	60.235	982	986.3	94.113
E-n101-B-80-k11	1,008	101	11	1,008	1,008.5	40.9	1,008	1,008	1,008.06	16.66	1,008	1,008.02	17.398	1,008	1,008.04	19.819
Average						14.19				3.34			**3.14**			4.08

J.2 Detailed results for the X instances

Table J.5: Detailed results for the X instances

Instance	BKS	$n+m$	ILS-SP Best	ILS-SP Avg.	ILS-SP CPU (s)	ILS_B-SP Best	ILS_B-SP Avg.	ILS_B-SP CPU (s)	ILS_B-SP_B Best	ILS_B-SP_B Avg.	ILS_B-SP_B CPU (s)
X-n101-50-k13	19,033	101	19,033	19,036.18	15,21	19,033	19,034.30	24.56	19,033	19,033.10	31.99
X-n101-66-k17	20,490	101	20,490	20,490.48	6,78	20,490	20,490.00	7.32	20,490	20,490.00	9.21
X-n101-80-k21	23,305	101	23,305	23,307.40	8,14	23,305	23,308.84	9.24	23,305	23,307.28	9.99
X-n106-50-k7	15,413	106	15,413	15,413.00	9,06	15,413	15,413.00	9.83	15,413	15,413.00	8.97
X-n106-66-k9	18,984	106	18,984	18,996.36	109,37	18,984	19,004.96	92.12	18,984	18,991.60	110.99
X-n106-80-k11	22,131	106	22,155	22,177.90	69,76	22,151	22,167.90	70.60	22,139	22,167.98	76.09
X-n110-50-k7	13,103	110	13,103	13,103.00	7,38	13,103	13,103.00	5.44	13,103	13,103.00	5.49
X-n110-66-k9	13,598	110	13,598	13,598.00	8,38	13,598	13,598.00	6.53	13,598	13,598.00	8.41
X-n110-80-k11	14,302	110	14,302	14,315.90	19,64	14,302	14,316.48	17.61	14,302	14,315.22	27.24
X-n115-50-k8	13,927	115	13,927	14,061.96	10,16	13,927	13,927.00	6.68	13,927	13,927.00	8.84
X-n115-66-k8	14,032	115	14,032	14,033.92	9,64	14,032	14,032.78	7.54	14,032	14,033.56	8.94
X-n115-80-k9	13,536	115	13,536	13,536.12	10,29	13,536	13,540.10	8.72	13,536	13,536.84	8.71
X-n120-50-k3	12,416	120	12,421	12,437.32	16,15	12,416	12,443.94	9.40	12,416	12,425.52	13.25
X-n120-66-k4	13,145	120	13,145	13,159.28	15,4	13,145	13,156.92	10.85	13,145	13,157.06	11.83
X-n120-80-k5	13,528	120	13,532	13,533.28	16,34	13,532	13,532.64	13.83	13,532	13,532.86	18.41
X-n125-50-k16	32,224	125	32,224	32,230.82	115,99	32,224	32,234.98	103.05	32,224	32,236.86	96.63
X-n125-66-k19	36,400	125	36,464	36,609.66	129,45	36,478	36,628.18	129.92	36,450	36,547.24	158.51
X-n125-80-k23	43,960	125	43,963	44,004.78	162,83	43,960	44,023.34	155.53	43,962	44,025.66	142.49
X-n129-50-k10	19,468	129	19,468	19,486.52	117,17	19,468	19,479.82	116.57	19,468	19,469.14	88.63
X-n129-66-k12	22,606	129	22,625	22,749.14	128,98	22,606	22,763.36	79.42	22,606	22,676.70	130.61
X-n129-80-k14	24,575	129	24,575	24,639.64	120,37	24,575	24,628.16	130.82	24,575	24,598.60	120.49
X-n134-50-k7	8,369	134	8,369	8,375.66	82,52	8,369	8,387.26	89.29	8,369	8,371.16	105.31
X-n134-66-k9	8,974	134	8,993	9,032.08	73,48	8,999	9,029.02	113.94	8,974	9,004.82	70.11
X-n134-80-k11	9,699	134	9,699	9,703.68	65,59	9,699	9,704.24	66.71	9,699	9,704.30	53.63
X-n139-50-k5	13,281	139	13,293	13,340.98	29,62	13,281	13,316.44	21.04	13,281	13,293.52	58.35
X-n139-66-k7	13,512	139	13,512	13,514.24	19,4	13,512	13,512.00	14.30	13,512	13,512.00	15.98
X-n139-80-k8	13,662	139	13,662	13,672.08	21,52	13,662	13,667.10	15.90	13,662	13,667.96	23.75
X-n143-50-k4	14,539	143	14,539	14,544.80	28,54	14,539	14,546.40	17.04	14,539	14,540.00	17.52
X-n143-66-k4	14,310	143	14,310	14,320.22	28,3	14,310	14,310.54	19.62	14,310	14,310.00	18.06
X-n143-80-k5	14,447	143	14,447	14,447.54	26,56	14,447	14,447.16	20.46	14,447	14,447.02	19.28
X-n148-50-k25	28,210	148	28,210	28,231.96	119,79	28,210	28,272.86	128.76	28,210	28,210.42	109.91
X-n148-66-k29	30,482	148	30,484	30,513.80	79,36	30,482	30,509.66	84.73	30,482	30,499.56	100.18
X-n148-80-k36	35,442	148	35,430	35,475.78	101,65	35,430	35,482.36	97.26	35,430	35,472.94	129.91
X-n153-50-k19	20,536	153	20,536	20,604.26	22,32	20,610	20,610.20	21.90	20,610	20,610.52	26.03
X-n153-66-k20	20,613	153	20,632	20,684.62	22,38	20,680	20,681.74	21.91	20,680	20,681.22	23.53
X-n153-80-k21	20,819	153	20,819	21,000.40	23,44	20,874	21,024.32	23.92	20,819	21,007.12	27.77
X-n157-50-k7	11,727	157	11,727	11,727.86	30,18	11,727	11,727.26	20.26	11,727	11,727.00	21.82
X-n157-66-k9	13,651	157	13,651	13,658.62	32,56	13,651	13,653.08	26.10	13,651	13,651.18	31.78
X-n157-80-k11	15,264	157	15,264	15,274.76	269,18	15,264	15,272.44	314.51	15,264	15,267.34	942.22
X-n162-50-k6	12,812	162	12,812	12,813.40	34,27	12,812	12,812.50	20.08	12,812	12,812.20	21.58
X-n162-66-k8	13,450	162	13,443	13,451.56	29,89	13,450	13,450.40	19.28	13,450	13,452.00	21.14
X-n162-80-k9	13,854	162	13,854	13,862.72	29,81	13,854	13,857.76	22.81	13,854	13,855.16	24.61
X-n167-50-k5	16,489	167	16,489	16,515.66	41,33	16,489	16,544.04	26.45	16,489	16,502.10	28.18
X-n167-66-k7	17,827	167	17,827	17,876.42	39,84	17,827	17,856.82	28.29	17,827	17,853.26	47.24
X-n167-80-k8	19,415	167	19,440	19,587.88	41,08	19,443	19,568.34	35.11	19,429	19,532.08	56.29
X-n172-50-k27	30,634	172	30,634	30,638.12	28,93	30,634	30,641.36	31.81	30,634	30,634.12	55.66

(*Continues on the next page*)

Instance	BKS	$n+m$	ILS-SP Best	ILS-SP Avg.	ILS-SP CPU (s)	ILS_B-SP Best	ILS_B-SP Avg.	ILS_B-SP CPU (s)	ILS_B-SP_B Best	ILS_B-SP_B Avg.	ILS_B-SP_B CPU (s)
X-n172-66-k31	31,864	172	31,864	31,882.08	33,69	31,864	31,877.82	35.69	31,864	31,886.92	53.78
X-n172-80-k39	36,803	172	36,803	36,821.78	30,99	36,803	36,815.42	32.51	36,803	36,822.18	42.26
X-n176-50-k23	45,239	176	45,239	45,371.04	41,35	45,239	45,406.60	41.13	45,239	45,353.10	41.74
X-n176-66-k24	46,416	176	46,532	46,796.86	46,88	46,655	46,852.54	44.12	46,561	46,835.86	47.62
X-n176-80-k25	47,033	176	47,293	47,448.68	45,44	47,288	47,469.24	49.53	47,288	47,474.02	77.99
X-n181-50-k12	16,549	181	16,549	16,573.92	38,06	16,549	16,565.74	30.92	16,549	16,562.36	47.31
X-n181-66-k15	18,832	181	18,832	18,832.00	38,84	18,832	18,832.00	31.15	18,832	18,832.00	45.46
X-n181-80-k18	21,241	181	21,241	21,241.00	42,01	21,241	21,241.00	37.33	21,241	21,241.00	52.67
X-n186-50-k8	17,978	186	18,040	18,084.38	46,66	17,990	18,037.80	31.03	18,040	18,060.86	52.43
X-n186-66-k10	19,751	186	19,751	19,803.66	47,32	19,751	19,797.02	35.45	19,751	19,781.00	38.76
X-n186-80-k12	21,754	186	21,770	21,825.78	57,15	21,773	21,806.44	51.56	21,761	21,801.00	61.38
X-n190-50-k4	11,552	190	11,552	11,562.72	75,83	11,552	11,558.36	43.36	11,552	11,554.08	44.47
X-n190-66-k5	12,784	190	12,823	12,929.02	73,38	12,843	12,918.36	46.82	12,824	12,890.10	51.29
X-n190-80-k6	14,410	190	14,410	14,508.44	80,19	14,410	14,472.04	66.68	14,416	14,469.16	90.33
X-n195-50-k27	29,470	195	29,470	29,530.90	62,18	29,470	29,488.10	37.93	29,470	29,478.78	112.84
X-n195-66-k34	33,137	195	33,137	33,211.60	44,22	33,137	33,215.60	43.08	33,137	33,199.28	60.27
X-n195-80-k42	38,629	195	38,629	38,660.30	41,94	38,629	38,652.72	45.16	38,629	38,660.90	62.86
X-n200-50-k18	34,416	200	34,843	34,870.50	113,15	34,828	34,882.52	181.50	34,799	34,836.18	473.69
X-n200-66-k24	40,474	200	40,474	40,494.76	479,69	40,485	40,492.74	509.35	40,474	40,489.68	719.47
X-n200-80-k29	47,741	200	47,743	47,784.60	240,15	47,763	47,785.00	253.97	47,763	47,783.30	413.55
X-n204-50-k10	15,877	204	15,877	15,910.92	61,78	15,886	15,925.72	43.77	15,877	15,890.14	82.16
X-n204-66-k12	16,703	204	16,745	16,754.74	51,18	16,745	16,754.50	36.49	16,745	16,746.88	40.15
X-n204-80-k15	17,832	204	17,832	17,853.38	60	17,832	17,849.70	54.45	17,832	17,839.76	93.34
X-n209-50-k8	21,837	209	21,840	21,963.40	81,54	21,880	21,953.16	59.25	21,840	21,911.42	119.24
X-n209-66-k11	24,378	209	24,384	24,507.14	75,33	24,378	24,502.62	57.57	24,378	24,462.50	101.09
X-n209-80-k13	27,177	209	27,178	27,293.14	99,86	27,178	27,266.78	93.81	27,180	27,261.38	194.29
X-n214-50-k6	9,574	214	9,580	9,589.26	103	9,580	9,590.86	64.56	9,580	9,583.14	64.08
X-n214-66-k8	10,001	214	10,033	10,121.08	99,52	10,048	10,112.88	63.15	10,044	10,099.72	76.04
X-n214-80-k9	10,457	214	10,513	10,601.06	100,13	10,484	10,563.80	80.86	10,500	10,562.10	91.65
X-n219-50-k37	64,691	219	64,691	64,694.64	64,68	64,691	64,692.92	72.27	64,691	64,691.08	183.76
X-n219-66-k48	80,405	219	80,405	80,405.00	59,38	80,405	80,405.00	64.14	80,405	80,405.00	106.82
X-n219-80-k59	95,845	219	95,845	95,845.00	47,17	95,845	95,845.00	56.19	95,845	95,845.00	65.38
X-n223-50-k18	27,449	223	27,449	27,522.26	151,16	27,449	27,499.46	152.61	27,442	27,477.70	1,397.53
X-n223-66-k23	30,717	223	30,717	30,796.70	138,76	30,720	30,798.00	125.23	30,719	30,769.96	351.74
X-n223-80-k27	34,440	223	34,440	34,480.92	111,99	34,440	34,478.56	108.98	34,440	34,481.94	293.65
X-n228-50-k19	23,128	228	23,128	23,130.90	88,69	23,128	23,131.40	63.94	23,128	23,128.90	75.37
X-n228-66-k20	24,114	228	24,160	24,208.64	79,46	24,184	24,208.34	61.77	24,160	24,204.50	65.51
X-n228-80-k21	24,592	228	24,664	24,687.68	84,28	24,664	24,679.34	77.87	24,612	24,672.40	81.86
X-n233-50-k10	17,186	233	17,233	17,460.78	104,13	17,190	17,243.10	62.00	17,186	17,317.66	87.78
X-n233-66-k12	18,026	233	18,026	18,048.18	95,02	18,026	18,038.28	61.20	18,026	18,031.34	64.96
X-n233-80-k14	18,885	233	18,885	18,923.22	88,82	18,888	18,913.80	72.65	18,885	18,910.86	81.47
X-n237-50-k7	20,745	237	20,745	20,785.32	133,58	20,760	20,782.14	81.68	20,749	20,776.16	87.10
X-n237-66-k9	22,471	237	22,471	22,569.24	133,74	22,471	22,525.68	90.39	22,471	22,497.40	115.04
X-n237-80-k11	24,357	237	24,392	24,544.30	146,4	24,423	24,511.20	121.45	24,357	24,456.00	137.34
X-n242-50-k25	47,949	242	47,976	48,255.26	515,17	47,988	48,236.08	466.37	47,981	48,180.62	1,975.12
X-n242-66-k32	57,197	242	57,197	57,248.86	256,14	57,197	57,234.48	243.08	57,197	57,242.94	956.72
X-n242-80-k39	68,978	242	68,984	69,052.20	630,77	68,969	69,038.78	602.84	68,989	69,058.16	1,144.40
X-n247-50-k42	36,701	247	36,701	36,707.36	89,95	36,701	36,714.34	87.50	36,701	36,708.90	90.33
X-n247-66-k43	36,994	247	36,999	37,071.32	92,77	36,996	37,072.56	95.06	36,996	37,066.16	100.36
X-n247-80-k45	37,220	247	37,286	37,379.90	87,33	37,286	37,360.06	101.49	37,239	37,316.90	99.60

(*Continues on the next page*)

Instance	BKS	$n+m$	ILS-SP Best	ILS-SP Avg.	ILS-SP CPU (s)	ILS_B-SP Best	ILS_B-SP Avg.	ILS_B-SP CPU (s)	ILS_B-SP_B Best	ILS_B-SP_B Avg.	ILS_B-SP_B CPU (s)
X-n251-50-k14	24,968	251	25,052	25,333.02	340,93	25,142	25,343.96	489.02	25,065	25,200.68	1,241.92
X-n251-66-k18	27,817	251	27,922	28,064.48	197,36	27,916	28,048.40	162.24	27,881	27,994.04	792.93
X-n251-80-k22	32,170	251	32,229	32,351.90	296,74	32,196	32,333.86	311.08	32,197	32,321.50	596.30
X-n256-50-k8	15,922	256	15,922	15,923.70	127,15	15,922	15,927.02	77.35	15,922	15,922.00	96.10
X-n256-66-k11	17,250	256	17,250	17,276.70	124,73	17,250	17,291.30	85.73	17,250	17,263.12	114.44
X-n256-80-k13	18,189	256	18,213	18,359.98	122,43	18,216	18,421.86	94.96	18,201	18,325.74	147.60
X-n261-50-k7	21,555	261	21,627	21,770.38	201,99	21,635	21,771.94	119.59	21,652	21,743.28	129.13
X-n261-66-k9	23,065	261	23,328	23,436.48	194,33	23,284	23,405.08	128.11	23,214	23,348.52	156.96
X-n261-80-k11	25,128	261	25,204	25,558.70	196,41	25,287	25,481.62	167.72	25,213	25,472.14	261.23
X-n266-50-k30	47,815	266	47,845	47,949.86	828,73	47,815	47,934.00	808.08	47,783	47,839.68	978.74
X-n266-66-k39	55,962	266	55,961	55,997.80	769,31	55,963	56,003.08	770.33	55,938	55,970.88	856.93
X-n266-80-k47	63,947	266	63,885	63,961.14	786,81	63,880	63,944.92	731.49	63,880	63,949.20	930.69
X-n270-50-k18	24,776	270	24,844	24,887.46	227,59	24,843	24,890.46	293.42	24,844	24,858.60	1,639.11
X-n270-66-k24	26,377	270	26,377	26,385.14	110,32	26,377	26,386.48	78.59	26,377	26,382.32	227.09
X-n270-80-k29	29,789	270	29,789	29,817.14	132,03	29,789	29,811.76	115.62	29,789	29,811.54	267.58
X-n275-50-k14	15,561	275	15,563	15,638.78	172,78	15,563	15,663.62	132.98	15,561	15,587.78	612.28
X-n275-66-k19	16,944	275	16,970	16,993.00	174,05	16,970	16,989.72	132.38	16,944	16,966.76	268.04
X-n275-80-k22	18,690	275	18,694	18,721.08	241,31	18,697	18,713.52	241.90	18,689	18,703.94	669.37
X-n280-50-k13	29,132	280	29,170	29,396.66	249,45	29,198	29,353.50	159.74	29,200	29,316.26	211.75
X-n280-66-k15	31,315	280	31,438	31,692.14	254,39	31,461	31,661.28	196.68	31,427	31,603.18	414.69
X-n280-80-k16	32,332	280	32,409	32,547.06	246,96	32,405	32,529.32	223.05	32,412	32,508.26	411.12
X-n284-50-k8	15,944	284	15,969	16,052.64	252,94	15,997	16,051.28	145.35	15,944	16,019.60	159.20
X-n284-66-k10	17,277	284	17,341	17,434.18	241,5	17,316	17,391.62	155.19	17,312	17,371.68	165.56
X-n284-80-k12	18,830	284	18,868	18,951.34	237,78	18,871	18,944.92	192.76	18,873	18,926.52	270.90
X-n289-50-k34	57,957	289	58,008	58,259.72	764,55	57,971	58,220.34	791.97	57,950	58,073.66	1,791.63
X-n289-66-k38	63,446	289	63,496	63,801.58	896,65	63,496	63,775.36	937.29	63,441	63,657.52	1,458.47
X-n289-80-k47	75,963	289	75,970	76,225.44	1072,58	76,040	76,236.38	1,156.53	76,023	76,216.14	1,805.94
X-n294-50-k26	30,859	294	30,859	30,925.00	157,24	30,859	30,932.14	101.57	30,859	30,881.50	1,095.26
X-n294-66-k33	34,636	294	34,669	34,742.14	146,12	34,636	34,722.56	127.60	34,644	34,709.48	260.96
X-n294-80-k40	39,269	294	39,265	39,313.98	226,59	39,269	39,318.62	227.60	39,269	39,333.10	688.80
X-n298-50-k16	25,081	298	25,081	25,138.86	178,44	25,081	25,104.80	124.72	25,081	25,085.50	219.47
X-n298-66-k21	27,644	298	27,694	27,836.76	337,08	27,643	27,783.76	242.42	27,678	27,770.98	1,497.88
X-n298-80-k25	30,222	298	30,222	30,345.24	402,56	30,222	30,325.88	460.03	30,222	30,330.16	1,479.26
X-n303-50-k11	17,763	303	17,771	17,802.50	248,23	17,776	17,834.54	150.57	17,739	17,789.78	283.20
X-n303-66-k13	18,120	303	18,121	18,159.22	224,91	18,120	18,152.22	145.73	18,122	18,142.56	176.79
X-n303-80-k16	19,603	303	19,772	19,950.94	265,08	19,660	19,727.10	186.91	19,710	19,814.04	305.20
X-n308-50-k9	22,544	308	22,637	22,903.52	365,33	22,699	22,868.00	199.52	22,673	22,827.48	231.64
X-n308-66-k11	24,154	308	24,315	24,592.12	368,26	24,225	24,422.92	235.34	24,227	24,436.34	303.74
X-n308-80-k12	25,164	308	25,291	25,419.10	355,99	25,172	25,382.26	284.34	25,172	25,356.98	437.52
X-n313-50-k39	57,762	313	57,774	57,884.38	930,09	57,765	57,869.64	934.92	57,778	57,864.74	2,042.98
X-n313-66-k44	60,089	313	60,216	60,485.46	953,36	60,282	60,481.82	1,054.46	60,204	60,357.74	1,586.63
X-n313-80-k56	73,869	313	73,856	73,987.34	723,01	73,834	74,022.44	847.19	73,893	74,019.74	1,335.42
X-n317-50-k27	43,396	317	43,397	43,440.44	1431,88	43,397	43,430.44	1,686.92	43,391	43,411.72	2,972.38
X-n317-66-k35	54,505	317	54,502	54,520.60	993,88	54,502	54,521.56	1,059.65	54,502	54,508.80	1,202.35
X-n317-80-k43	63,687	317	63,683	63,683.46	359,6	63,683	63,683.20	330.54	63,683	63,683.36	792.75
X-n322-50-k14	23,309	322	23,310	23,427.12	217	23,309	23,385.46	144.01	23,309	23,334.18	415.75
X-n322-66-k19	25,034	322	25,034	25,078.82	206,69	25,034	25,074.00	152.75	25,034	25,057.62	239.81
X-n322-80-k23	27,500	322	27,526	27,638.46	337,69	27,519	27,633.82	315.24	27,506	27,619.74	1,030.98
X-n327-50-k10	21,610	327	21,781	21,930.52	398,71	21,755	21,890.16	255.32	21,728	21,827.26	603.45
X-n327-66-k13	23,322	327	23,413	23,605.34	353,95	23,423	23,544.16	247.27	23,340	23,483.22	410.57

(*Continues on the next page*)

Instance	BKS	$n+m$	ILS-SP Best	ILS-SP Avg.	ILS-SP CPU (s)	ILS_B-SP Best	ILS_B-SP Avg.	ILS_B-SP CPU (s)	ILS_B-SP_B Best	ILS_B-SP_B Avg.	ILS_B-SP_B CPU (s)
X-n327-80-k16	24,990	327	25,109	25,281.66	432,69	25,106	25,227.88	420.18	25,046	25,205.80	1,078.86
X-n331-50-k8	24,152	331	24,207	24,342.08	405,82	24,259	24,345.72	234.77	24,152	24,316.32	271.64
X-n331-66-k10	26,247	331	26,369	26,599.16	401,49	26,366	26,536.16	269.38	26,305	26,496.74	395.80
X-n331-80-k12	28,265	331	28,366	28,534.74	456,33	28,358	28,478.10	361.27	28,284	28,428.66	524.12
X-n336-50-k45	81,760	336	81,850	82,095.26	1081,79	81,818	82,134.74	1,116.63	81,783	82,003.10	2,226.02
X-n336-66-k57	99,283	336	99,275	99,459.48	981,19	99,226	99,471.24	1,084.34	99,251	99,410.14	1,701.81
X-n336-80-k68	116,238	336	116,186	116,321.62	906,6	116,185	116,353.78	1,050.20	116,207	116,353.74	1,750.12
X-n344-50-k22	28,527	344	28,574	28,700.04	412,95	28,602	28,702.36	344.90	28,527	28,595.74	2,181.79
X-n344-66-k29	31,873	344	31,845	31,926.54	620,53	31,845	31,936.72	597.92	31,837	31,933.62	2,465.84
X-n344-80-k35	35,743	344	35,743	35,832.54	541,29	35,751	35,830.96	575.11	35,761	35,844.06	1,079.15
X-n351-50-k21	18,584	351	18,702	18,793.56	553,03	18,658	18,765.62	514.01	18,635	18,776.72	2,474.45
X-n351-66-k26	19,758	351	19,799	20,011.68	794,5	19,842	19,942.80	546.97	19,865	19,963.38	2,018.64
X-n351-80-k32	22,158	351	22,258	22,371.74	728,46	22,257	22,326.64	604.08	22,236	22,318.52	1,184.00
X-n359-50-k15	33,255	359	33,640	33,934.24	645,79	33,581	33,878.76	568.40	33,679	33,884.38	2,375.60
X-n359-66-k19	37,695	359	37,886	38,181.78	778,39	37,835	38,074.14	637.55	37,748	38,083.38	2,207.93
X-n359-80-k23	43,412	359	43,585	43,907.40	1309,93	43,566	43,818.50	1,321.96	43,561	43,864.46	2,520.75
X-n367-50-k12	20,526	367	20,804	20,911.18	560,39	20,799	20,842.52	313.23	20,700	20,866.94	586.92
X-n367-66-k14	21,479	367	21,493	21,567.02	542,85	21,491	21,549.08	333.22	21,479	21,535.12	382.21
X-n367-80-k15	22,386	367	22,439	22,568.58	566,03	22,405	22,531.80	482.80	22,408	22,487.54	492.41
X-n376-50-k47	80,794	376	80,736	80,791.48	539,89	80,736	80,780.16	537.80	80,736	80,748.24	1,352.74
X-n376-66-k62	100,616	376	100,613	100,613.64	895,45	100,613	100,613.36	885.54	100,613	100,613.30	1,141.06
X-n376-80-k75	119,581	376	119,581	119,581.00	303,21	119,581	119,581.00	306.14	119,581	119,581.00	533.75
X-n384-50-k27	41,206	384	41,365	41,682.84	1703,4	41,380	41,684.14	1,748.78	41,384	41,740.80	3,284.00
X-n384-66-k35	47,373	384	47,422	47,675.94	1058,74	47,484	47,701.86	1,105.37	47,436	47,670.76	2,009.43
X-n384-80-k42	55,386	384	55,490	55,658.10	1481,68	55,442	55,658.40	1,434.17	55,469	55,667.08	2,401.40
X-n393-50-k19	30,005	393	30,212	30,352.56	1076,38	30,093	30,324.08	1,102.95	30,089	30,256.02	3,119.32
X-n393-66-k25	29,340	393	29,441	29,582.04	932,81	29,450	29,568.14	850.72	29,444	29,618.16	2,903.91
X-n393-80-k31	32,619	393	32,644	32,787.78	1364,19	32,640	32,783.20	1,305.44	32,675	32,840.76	3,055.94
X-n401-50-k15	39,746	401	40,074	40,230.80	1199,5	39,987	40,196.74	1,029.33	39,995	40,203.66	3,128.46
X-n401-66-k20	47,658	401	47,703	47,920.88	1473,83	47,707	47,871.76	1,164.73	47,681	47,909.92	2,730.72
X-n401-80-k23	54,270	401	54,512	54,666.08	1957,54	54,343	54,569.58	1,977.74	54,329	54,572.30	2,858.43
X-n411-50-k14	17,959	411	18,095	18,245.82	752,56	18,128	18,256.42	445.36	18,083	18,233.06	546.56
X-n411-66-k15	18,785	411	18,835	18,919.08	776,96	18,816	18,903.40	473.03	18,810	18,880.10	649.76
X-n411-80-k17	19,496	411	19,552	19,759.92	816,08	19,607	19,735.88	708.86	19,553	19,688.90	1,176.16
X-n420-50-k67	75,549	420	75,598	75,869.72	2792,19	75,527	75,895.58	2,887.30	75,497	75,747.30	3,213.14
X-n420-66-k86	76,109	420	76,095	76,171.24	1712,09	76,079	76,176.68	1,717.76	76,079	76,125.50	2,328.46
X-n420-80-k105	89,391	420	89,383	89,494.78	1255,12	89,381	89,478.40	1,270.78	89,396	89,533.20	2,273.54
X-n429-50-k31	41,284	429	41,448	41,632.10	1389,94	41,378	41,587.28	1,274.06	41,364	41,640.98	3,175.85
X-n429-66-k40	47,793	429	47,850	48,106.62	1291,78	47,893	48,101.72	1,296.19	47,872	48,143.34	2,482.53
X-n429-80-k48	54,835	429	54,924	55,086.28	1653,12	54,908	55,066.92	1,597.41	54,969	55,114.06	2,496.20
X-n439-50-k19	27,011	439	27,086	27,225.32	739,34	27,039	27,256.48	530.80	27,010	27,086.82	1,784.11
X-n439-66-k25	28,895	439	28,882	28,978.92	662,31	28,883	28,961.24	511.33	28,878	28,935.20	1,824.67
X-n439-80-k30	32,074	439	32,098	32,155.42	1143,67	32,084	32,138.72	1,039.67	32,085	32,160.06	2,709.43
X-n449-50-k15	36,929	449	37,311	37,563.18	1187,05	37,108	37,545.92	848.72	37,230	37,534.04	2,685.97
X-n449-66-k20	41,846	449	42,265	42,612.38	1350,9	42,228	42,503.98	1,020.04	42,047	42,545.54	2,576.99
X-n449-80-k23	46,738	449	47,146	47,450.20	1676,76	47,103	47,378.96	1,662.08	47,140	47,413.96	2,899.36
X-n459-50-k14	18,891	459	18,931	19,024.18	1058,5	18,947	19,017.32	679.52	18,879	18,965.78	1,170.96
X-n459-66-k18	20,561	459	20,632	20,739.32	1068,7	20,594	20,716.60	776.70	20,575	20,682.48	1,611.84
X-n459-80-k21	22,047	459	22,108	22,263.16	1491,7	22,060	22,233.86	1,369.07	22,102	22,233.30	2,502.88
X-n469-50-k70	123,817	469	123,890	124,428.98	3074,85	124,190	124,541.22	3,161.39	123,980	124,426.44	3,645.45

(*Continues on the next page*)

Instance	BKS	$n+m$	ILS-SP Best	ILS-SP Avg.	ILS-SP CPU (s)	ILS_B-SP Best	ILS_B-SP Avg.	ILS_B-SP CPU (s)	ILS_B-SP_B Best	ILS_B-SP_B Avg.	ILS_B-SP_B CPU (s)
X-n469-66-k90	148,455	469	148,568	148,891.42	1831,76	148,476	148,873.36	1,826.92	148,571	148,779.38	2,657.46
X-n469-80-k109	178,511	469	178,602	178,814.22	1794,86	178,538	178,739.34	1,691.35	178,570	178,793.60	2,569.95
X-n480-50-k36	52,309	480	52,645	52,954.58	2865,16	52,697	52,951.36	2,722.02	52,751	53,030.64	3,660.36
X-n480-66-k47	63,577	480	63,639	63,737.90	1974,21	63,605	63,714.20	1,820.70	63,624	63,741.30	2,847.27
X-n480-80-k56	73,993	480	74,132	74,444.18	2142,54	74,200	74,438.86	2,115.58	74,326	74,465.44	2,884.06
X-n491-50-k30	43,952	491	44,392	44,595.46	3248,68	44,310	44,506.04	2,916.42	44,400	44,602.40	3,870.00
X-n491-66-k39	49,627	491	49,903	50,308.20	3392,14	49,901	50,275.68	3,202.94	50,012	50,322.60	4,059.82
X-n491-80-k47	56,141	491	56,304	56,676.46	2336,32	56,309	56,613.76	2,197.72	56,472	56,662.86	2,904.17
X-n502-50-k20	40,591	502	40,632	40,662.88	1794,59	40,633	40,674.18	2,861.03	40,616	40,677.42	3,589.01
X-n502-66-k26	49,285	502	49,338	49,393.10	3662,14	49,318	49,382.86	3,368.36	49,307	49,420.56	3,965.62
X-n502-80-k31	56,997	502	57,015	57,059.56	1593,09	57,001	57,043.12	1,454.89	56,992	57,039.66	2,807.98
X-n513-50-k11	21,675	513	21,812	21,985.32	1661,12	21,769	21,947.04	832.33	21,696	21,819.24	1,184.20
X-n513-66-k14	22,426	513	22,585	22,824.26	1387,65	22,475	22,660.12	794.47	22,454	22,595.84	1,050.45
X-n513-80-k17	23,448	513	23,600	23,718.48	1204,13	23,502	23,625.76	911.95	23,509	23,597.46	1,111.15
X-n524-50-k125	154,162	524	154,137	154,144.54	678,33	154,137	154,158.18	988.64	154,137	154,160.76	1,238.25
X-n524-66-k129	154,454	524	154,416	154,464.34	806,6	154,416	154,463.12	1,041.80	154,416	154,458.44	1,145.98
X-n524-80-k132	154,534	524	154,497	154,653.56	755,12	154,497	154,631.56	1,003.81	154,497	154,597.94	969.52
X-n536-50-k49	54,658	536	54,846	55,077.56	2562,37	54,873	55,077.36	2,289.58	55,012	55,175.32	3,558.97
X-n536-66-k64	66,032	536	66,143	66,316.66	2727	66,131	66,303.52	2,589.26	66,133	66,308.34	3,254.26
X-n536-80-k77	77,811	536	78,384	78,501.72	3300,44	78,370	78,479.58	3,263.29	78,384	78,500.94	3,887.19
X-n548-50-k25	53,049	548	53,353	53,627.02	3078,13	53,295	53,554.26	2,677.05	53,259	53,576.74	4,519.00
X-n548-66-k33	61,421	548	61,415	61,722.52	3454,66	61,434	61,646.08	2,823.55	61,429	61,664.46	4,751.30
X-n548-80-k40	71,962	548	71,887	72,132.12	4068,65	71,867	72,065.58	3,589.45	71,969	72,161.58	4,335.31
X-n561-50-k22	31,826	561	32,090	32,233.24	1790,98	32,115	32,237.62	1,251.14	32,060	32,232.40	3,479.13
X-n561-66-k28	34,370	561	34,599	34,791.48	1660,66	34,550	34,764.54	1,384.45	34,461	34,744.88	2,863.92
X-n561-80-k34	38,053	561	38,197	38,380.08	2227,36	38,066	38,341.00	2,190.15	38,160	38,351.94	3,456.24
X-n573-50-k22	40,239	573	40,769	41,268.12	2935,02	40,660	41,183.74	2,133.38	40,582	41,009.34	3,793.69
X-n573-66-k25	44,151	573	44,320	44,501.82	2668,17	44,247	44,444.28	1,954.25	44,347	44,483.16	3,244.08
X-n573-80-k27	47,054	573	47,266	47,408.86	2967,32	47,175	47,328.12	2,776.15	47,190	47,366.70	4,155.16
X-n586-50-k80	122,632	586	122,876	123,085.80	4637,98	122,954	123,147.68	4,715.40	122,849	123,119.72	5,036.44
X-n586-66-k105	140,396	586	140,673	140,920.64	3822,45	140,678	140,954.28	3,828.81	140,535	140,913.58	4,217.18
X-n586-80-k127	160,460	586	160,419	160,744.00	3482	160,390	160,713.00	3,599.47	160,463	160,768.56	3,980.49
X-n599-50-k47	65,292	599	65,687	65,916.96	4529,38	65,549	65,873.08	4,156.63	65,547	65,899.96	4,401.40
X-n599-66-k61	76,472	599	76,706	77,012.96	3076,46	76,604	77,003.66	2,887.26	76,700	77,069.50	3,588.14
X-n599-80-k74	89,844	599	90,101	90,376.66	3464,39	89,981	90,357.60	3,426.41	90,162	90,416.94	3,908.59
X-n613-50-k32	40,838	613	41,255	41,467.54	2872,82	41,074	41,406.48	2,410.45	41,126	41,512.40	4,137.92
X-n613-66-k41	46,074	613	46,284	46,553.70	3298,62	46,181	46,509.32	2,943.49	46,212	46,594.26	4,913.78
X-n613-80-k50	52,096	613	52,389	52,637.52	4082,34	52,241	52,585.86	3,889.53	52,365	52,640.62	4,145.17
X-n627-50-k22	38,096	627	38,423	38,644.90	4552,96	38,430	38,598.54	3,529.69	38,418	38,586.82	4,997.05
X-n627-66-k29	44,782	627	44,996	45,171.46	5412,62	44,792	45,100.38	4,484.16	45,002	45,125.86	5,073.93
X-n627-80-k35	52,429	627	52,622	52,785.20	6363,34	52,530	52,721.26	5,625.66	52,576	52,730.08	5,843.21
X-n641-50-k18	42,333	641	42,631	42,927.80	4200,11	42,544	42,901.22	2,937.90	42,645	42,892.36	4,734.33
X-n641-66-k23	47,501	641	47,859	48,228.18	4085,35	47,819	48,118.08	3,086.13	47,940	48,159.38	6,379.56
X-n641-80-k28	54,116	641	54,313	54,820.84	4871,43	54,332	54,695.14	4,267.43	54,352	54,727.50	5,157.62
X-n655-50-k66	59,442	655	59,416	59,516.02	3407,75	59,451	59,536.32	3,379.35	59,442	59,552.12	4,281.04
X-n655-66-k87	72,491	655	72,457	72,503.16	2679,14	72,456	72,504.62	2,585.18	72,456	72,494.64	3,757.15
X-n655-80-k105	86,588	655	86,564	86,575.20	2742,04	86,564	86,571.10	2,588.08	86,564	86,579.58	3,595.05
X-n670-50-k112	144,707	670	144,688	144,823.88	1623,14	144,723	144,846.36	1,850.98	144,720	144,830.58	1,930.46
X-n670-66-k117	144,990	670	144,981	145,162.10	1631,79	144,991	145,218.66	1,732.61	144,960	145,122.54	1,762.80
X-n670-80-k120	145,490	670	145,247	145,627.30	1756,92	145,275	145,585.36	2,086.33	145,151	145,484.72	2,035.45

(*Continues on the next page*)

Instance	BKS	$n+m$	ILS-SP Best	ILS-SP Avg.	ILS-SP CPU (s)	ILS_B-SP Best	ILS_B-SP Avg.	ILS_B-SP CPU (s)	ILS_B-SP_B Best	ILS_B-SP_B Avg.	ILS_B-SP_B CPU (s)
X-n685-50-k43	48,023	685	48,300	48,615.42	4438,54	48,228	48,574.98	3,864.07	48,312	48,656.68	4,683.08
X-n685-66-k54	53,240	685	53,353	53,787.34	4323,23	53,508	53,798.82	3,806.79	53,610	53,906.20	4,357.33
X-n685-80-k62	59,301	685	59,644	59,898.34	3576,4	59,624	59,808.76	2,406.18	59,656	59,999.52	5,352.39
X-n701-50-k23	51,390	701	52,231	52,523.78	6718,85	52,254	52,469.70	4,644.50	52,274	52,482.38	6,259.41
X-n701-66-k30	58,844	701	59,760	60,042.74	6379,35	59,642	59,920.98	4,636.17	59,688	59,912.80	5,541.16
X-n701-80-k36	68,618	701	69,104	69,607.76	6830,29	69,187	69,496.60	5,824.26	69,160	69,464.48	6,188.58
X-n716-50-k18	29,757	716	30,201	30,453.44	7945,31	30,110	30,334.56	4,224.00	30,145	30,428.84	6,362.58
X-n716-66-k23	32,527	716	32,966	33,156.66	6669,41	32,886	33,077.10	4,576.08	32,953	33,076.46	6,520.48
X-n716-80-k28	37,976	716	38,313	38,541.02	7,431	38,358	38,469.24	6,179.14	38,330	38,461.72	8,122.80
X-n733-50-k83	80,585	733	80,874	81,245.28	4,253	80,774	81,258.44	3,936.58	80,904	81,277.36	5,120.46
X-n733-66-k102	92,156	733	92,345	92,664.86	4,157	92,435	92,689.20	4,113.29	92,358	92,751.70	4,608.45
X-n733-80-k125	110,659	733	110,801	111,106.86	4,308	110,806	111,105.88	4,415.33	110,868	111,174.84	4,698.41
X-n749-50-k49	47,740	749	48,341	48,570.72	7,018	48,360	48,549.62	5,824.83	48,318	48,588.18	6,573.66
X-n749-66-k63	55,560	749	55,964	56,168.42	6,393	55,985	56,151.42	5,615.70	56,017	56,170.76	5,694.44
X-n749-80-k78	63,991	749	64,437	64,612.10	5,058	64,284	64,564.26	4,729.02	64,439	64,588.46	5,157.63
X-n766-50-k58	95,674	766	96,639	97,017.38	8,543	96,599	96,985.94	6,209.82	96,500	96,997.04	6,909.81
X-n766-66-k62	101,566	766	101,999	102,747.22	8,667	102,178	102,504.40	6,471.67	102,189	102,569.92	7,226.96
X-n766-80-k65	106,758	766	107,191	107,955.82	9,077	107,427	107,707.64	7,701.13	107,327	107,759.82	8,838.75
X-n783-50-k24	49,027	783	49,835	50,738.66	8,815	49,987	50,263.96	5,078.68	49,876	50,259.10	6,859.64
X-n783-66-k31	53,429	783	54,392	54,985.58	8,178	54,063	54,593.00	5,016.52	54,190	54,579.58	7,539.50
X-n783-80-k38	60,937	783	61,670	61,930.74	7,847	61,411	61,797.92	6,433.57	61,505	61,786.18	7,056.14
X-n801-50-k20	48,459	801	49,338	49,670.18	14,236	49,258	49,541.04	7,231.37	49,177	49,610.44	10,468.79
X-n801-66-k27	54,929	801	55,385	55,638.34	12,269	55,232	55,478.34	7,216.42	55,238	55,501.48	9,612.64
X-n801-80-k32	62,698	801	63,089	63,262.94	12,984	62,858	63,127.84	9,598.68	62,900	63,131.66	11,655.50
X-n819-50-k86	89,296	819	89,536	89,831.50	5,797	89,594	89,837.82	5,173.58	89,533	89,926.40	6,294.31
X-n819-66-k112	108,431	819	108,587	108,840.12	5,362	108,636	108,837.12	5,190.30	108,655	108,911.72	5,784.03
X-n819-80-k136	128,617	819	128,925	129,279.64	5,732	128,901	129,259.10	5,793.37	128,922	129,327.80	6,198.00
X-n837-50-k71	116,553	837	116,759	117,029.56	11,938	116,868	117,128.60	10,212.23	116,934	117,107.84	11,242.33
X-n837-66-k94	129,183	837	129,593	129,910.02	7,090	129,691	129,894.46	5,929.24	129,708	129,914.60	8,160.78
X-n837-80-k114	154,966	837	155,536	155,719.52	6,565	155,390	155,667.52	6,302.08	155,412	155,695.90	7,267.38
X-n856-50-k48	57,777	856	57,911	58,109.54	9,024	57,936	58,114.44	7,317.24	57,855	58,100.26	8,018.41
X-n856-66-k63	63,550	856	63,602	63,892.90	7,086	63,542	63,862.26	6,039.45	63,735	63,923.24	7,136.64
X-n856-80-k76	73,802	856	73,881	74,094.48	7,030	73,848	74,068.40	6,252.84	73,958	74,091.12	6,739.03
X-n876-50-k30	58,780	876	59,261	59,547.24	14,926	59,268	59,512.92	8,585.59	59,434	59,588.10	10,245.76
X-n876-66-k38	69,617	876	70,052	70,271.38	14,095	70,038	70,184.76	8,916.99	70,018	70,239.02	9,764.81
X-n876-80-k46	80,983	876	81,439	81,749.62	14,126	81,462	81,635.70	10,611.29	81,512	81,704.26	12,270.26
X-n895-50-k19	40,668	895	41,302	41,735.88	13,365	41,324	41,580.46	6,593.86	41,204	41,779.64	9,787.44
X-n895-66-k25	44,059	895	44,642	44,935.04	11,116	44,353	44,792.08	6,539.38	44,658	44,858.92	10,614.66
X-n895-80-k30	48,451	895	49,073	49,399.70	12,030	49,063	49,324.66	9,666.41	48,858	49,233.60	11,600.48
X-n916-50-k105	190,413	916	189,911	190,140.22	11,686	190,108	190,290.92	10,371.39	190,040	190,302.86	12,541.55
X-n916-66-k136	222,807	916	223,211	223,586.82	9,129	223,229	223,575.76	8,378.68	223,227	223,576.68	9,829.22
X-n916-80-k165	263,885	916	264,488	264,994.44	8,082	264,625	264,902.78	8,232.98	264,577	264,924.12	8,702.22
X-n936-50-k132	127,522	936	127,479	127,696.88	4,301	127,497	127,787.32	3,967.18	127,445	127,745.36	4,251.39
X-n936-66-k138	128,871	936	128,832	129,387.84	4,890	128,978	129,385.34	4,626.05	128,899	129,232.64	4,946.79
X-n936-80-k143	130,808	936	130,935	131,333.48	5,205	130,823	131,229.32	5,339.50	130,736	131,274.98	5,783.04
X-n957-50-k44	57,019	957	57,282	57,490.40	18,072	57,189	57,380.88	12,446.31	57,234	57,438.90	13,523.84
X-n957-66-k58	62,593	957	62,782	62,942.70	12,316	62,756	62,890.16	8,805.03	62,760	62,891.56	9,927.86
X-n957-80-k70	71,855	957	72,008	72,118.86	12,881	71,931	72,084.92	9,895.91	71,990	72,094.30	11,658.77
X-n979-50-k30	69,739	979	70,367	71,219.92	26,013	70,124	70,736.52	13,812.99	70,194	70,633.06	18,093.25
X-n979-66-k39	84,499	979	84,967	85,415.90	22,968	84,979	85,268.60	13,699.44	84,994	85,225.58	17,756.82

(Continues on the next page)

Instance	BKS	$n+m$	ILS-SP			ILS_B-SP			ILS_B-SP_B		
			Best	Avg.	CPU (s)	Best	Avg.	CPU (s)	Best	Avg.	CPU (s)
X-n979-80-k47	99,605	979	100,252	100,642.08	22,511	100,107	100,442.10	17,266.25	99,983	100,315.92	16,338.46
X-n1001-50-k22	49,978	1,001	51,158	51,542.36	26,543	51,132	51,501.80	11,818.16	51,279	51,558.32	14,322.40
X-n1001-66-k28	56,126	1,001	57,418	57,645.24	25,849	56,850	57,473.24	12,371.27	56,930	57,498.82	14,946.78
X-n1001-80-k34	63,278	1,001	64,208	64,650.36	24,643	64,218	64,421.96	17,155.69	64,110	64,412.80	18,572.64
Average					2,599.77			2,005.48			2,660.19

APPENDIX K – A representative small subset of X

A minimum subset of the instances X which covers all the characteristics considered in Uchoa et al. [115] is described below:

- Route size (interval for n/K_{min}):
  - $[3, 5]$: X-n469-k138
  - $(5, 8]$: X-n670-k130
  - $(8, 11]$: X-n393-k38
  - $(11, 14]$: X-n561-k42
  - $(14, 17]$: X-n979-k58
  - $(17, 20]$: X-n801-k40
  - $(20, 25]$: X-n716-k35
- Depot positioning:
  - Random: X-n670-k130, X-n716-k35
  - Center: X-n393-k38, X-n561-k42
  - Corner: X-n469-k138, X-n801-k40, X-n979-k58
- Customers distribution:
  - Random: X-n469-k138, X-n670-k130, X-n801-k40
  - Clustered: X-n716-k35, X-n979-k58
  - Random-clustered: X-n393-k38, X-n561-k42
- Customers demands:
  - Unitary: X-n801-k40

- Small values, large CV[1]: X-n561-k42
- Small values, small CV: X-n393-k38
- Large values, large CV: X-n716-k35
- Large values, small CV: X-n469-k138
- Depending on quadrant: X-n979-k58
- Many small values, few large values: X-n670-k130

Thus, the subset X_R is composed by: X-n393-k38, X-n469-k138, X-n561-k42, X-n670-k130, X-n716-k35, X-n801-k40, X-n979-k58. The reader is referred to Uchoa et al. [115] for a detailed description of all characteristics.

[1] Coefficient of variation

APPENDIX L – Comparison of HGS and HGS^r

Figure L.1 compares the performance of a single execution of HGS and HGS^r for all X instances over 8 hours. The convergence curves of both algorithms report the average gap (considering the gap obtained for each instance) found at different times. The final average gaps obtained by HGS and HGS^r were 0.295% and 0.236%, respectively.

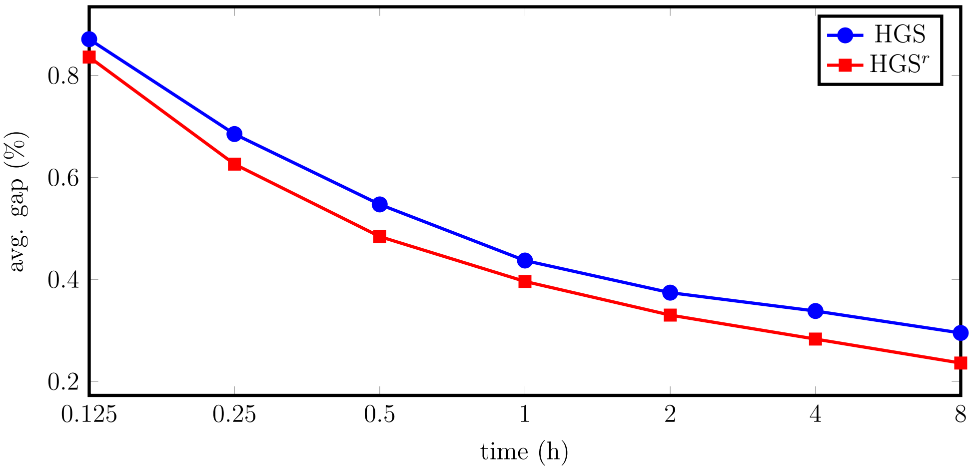


Figure L.1: Comparison of HGS and HGS^r w.r.t. the convergence curve based on the average gap for all X instances over 8 hours. The time axis is on a log_2 scale.

APPENDIX M – VRPSolver Parameterizations

Table M.1 shows the default VRPSolver CVRP parameterization, as well as the changed parameterization in the version used to solve subproblems in POP. The reader is referred to the documentation of Bulhões et al. [23] to obtain the description of each parameter. The parameters which have special notation indicated in the second column of Table M.1 are also described in Pessoa et al. [92]. Default values for the last four parameters are not defined because these parameters are not active when the restricted master heuristic is not used. `RCSPmaxNumOfEnumSolsForEndOfNodeMIP` is a previously undocumented VRPSolver parameter. If the number of enumerated routes gets becomes smaller than `RCSPmaxNumOfEnumSolutionsForMIP`, at any point of a node solution, then the node is immediately solved by MIP.

Table M.1: Default and used parameters of the VRPSolver CVRP application.

Parameter	Notation	Default value	Used value
`RCSPhardTimeThresholdInPricing`	τ^{hard}	25 secs	8 secs
`RCSPstopCutGenTimeThresholdInPricing`	τ^{soft}	10 secs	3 secs
`RCSPnumberOfBucketsPerVertex`	τ^{hard}	25	50
`RCSPmaxNumOfLabelsInEnumeration`	ω^{labels}	$5 \cdot 10^6$	$3 \cdot 10^5$
`RCSPmaxNumOfEnumeratedSolutions`	ω^{routes}	$5 \cdot 10^6$	10^6
`RCSPmaxNumOfEnumSolutionsForMIP`	ω^{MIP}	10^4	$5 \cdot 10^3$
`RCSPmaxNumOfEnumSolsForEndOfNodeMIP`	–	10^4	10^4
`RCSPuseBidirectionalSearch`	ϕ^{bidir}	2	1
`RCSPrankOneCutsMemoryType`	θ^{mem}	0	0
`CutTailingOffThreshold`	δ^{gap}	0.015	0.03
`StrongBranchingPhaseOneCandidatesNumber`	ζ_1^{num}	100	50
`StrongBranchingPhaseOneTreeSizeEstimRatio`	ζ_1^{estim}	0.2	0.2
`StrongBranchingPhaseTwoCandidatesNumber`	ζ_2^{num}	5	3
`StrongBranchingPhaseTwoTreeSizeEstimRatio`	ζ_2^{estim}	0.02	0.02
`MaxTimeForRestrictedMasterIpHeur`	χ^{rm}	-1 (off)	40
`CallFrequencyOfRestrictedMasterIpHeur`	–	–	1
`MIPemphasisInRestrictedMasterIpHeur`	–	–	1
`RCSPmaxNumOfLabelsInHeurEnumeration`	–	–	10^5
`MaxNumEnumSolsInRestrictedMasterIpHeur`	–	–	10^4

Table M.2 shows the additional parameters used for obtaining BCP_H. While they reduce running times, the optimal solution of a subproblem may be missed.

Table M.2: Additional parameters for obtaining BCP_H

Parameter	Notation	Default value	Used value
`GlobalTimeLimit`	–	∞	3600 secs
`MaxNbOfBBtreeNodeTreated`	–	∞	10
`RCSPfalseGapFactor`	–	0 (off)	3

APPENDIX N – Detailed results for the HFVRP and VRPB

Table N.1: Best solutions found by HILS and $POP^h_{0.5}$ after 32 hours. The column BKS reports the best upper bound in Pessoa, Sadykov, and Uchoa [90]. Instances with the substring "FSM" in their names belong to XH-FSM, while the other ones belong to XH-HVRP.

Instance	BKS	HILS	$POP^h_{0.5}$
X303-FSMFD	35993.50	36112.06	**35960.75**
X308-FSMF	51965.10	52057.16	52438.06
X313-FSMD	93361.60	93337.55	**93325.55**
X317-HVRP	165763.00	165763.39	165763.39
X322-HD	33507.80	33521.93	33535.29
X327-FSMFD	38672.80	38748.57	**38340.48**
X331-FSMF	63082.70	63750.1	63232.26
X336-FSMF	212635.90	211357.4	**210312.95**
X344-FSMD	42369.70	42370.48	42369.72
X351-HVRP	54124.90	54240.66	**53936.57**
X359-HD	60737.70	60923.07	**59947.73**
X367-FSMFD	51605.00	51627.35	**51592.72**
X376-HD	161394.20	161394.21	161394.21
X384-FSMF	105143.60	105381.32	**100719.64**
X393-HVRP	72748.10	73180.22	**72230.09**
X401-FSMFD	89755.70	**89716.66**	90000.85
X411-FSMD	18430.90	18461.53	**18219.79**
X420-FSMD	112984.80	112753.18	**112622.13**
X429-HVRP	91732.20	92114.81	**91547.96**
X439-FSMF	71877.00	72859.22	**70320.17**
X449-FSMFD	113204.30	113399.96	**112857.46**
X459-HD	25359.10	25356.2	**25009.24**
X469-HD	217177.80	217233.11	**216780.09**
X480-FSMD	100583.50	100697.55	**100561.86**
X491-FSMF	131315.50	131013.31	**126177.75**
X502-FSMFD	–	89935.24	**88014.99**
X513-HVRP	–	41586.54	**41480.64**
X524-HD	–	122415.92	**120964.77**
X536-FSMFD	–	199346	**199268.21**
X548-FSMF	–	135027.05	**126883.21**
X561-FSMD	–	46949.38	**46648.97**
X573-HVRP	–	105451.71	**105129.49**
X586-FSMF	–	367195.62	**359094.43**
X599-FSMD	–	133243.14	**133225.46**

Continued on next page

Table N.1 – continued from previous page

Instance	BKS	HILS	$POP^h_{0.5}$
X613-HD	–	63591.25	**62406.85**
X627-HVRP	–	**108524.69**	108571.75
X641-FSMFD	–	100590.06	**100146.59**
X655-HD	–	96607.45	**96589.42**
X670-FSMF	–	233678.54	**213111.59**
X685-FSMD	–	81197.95	**80877.56**
X701-HVRP	–	173635.78	**172527.97**
X716-FSMFD	–	59128.97	**59108.63**
X733-FSMFD	–	291937.39	**289248.44**
X749-FSMF	–	128356.03	**124994.19**
X766-FSMD	–	115751.08	**114536.82**
X783-HD	–	87217.09	**84600.07**
X801-HVRP	–	131696.8	**130860.09**
X819-FSMD	–	137333.07	**128516.50**
X837-HD	–	209229.81	**208995.91**
X856-HVRP	–	122669.55	**122683.39**
X876-FSMF	–	**160588.86**	161002.45
X895-FSMFD	–	75033.11	**74645.13**
X916-FSMFD	–	700308.45	**683329.86**
X936-FSMD	–	127111.15	**125923.09**
X957-HD	–	83611.04	**82995.84**
X979-HVRP	–	**218473.55**	219745.05
X1001-FSMF	–	88567.65	**84233.42**

Table N.2: Best solutions found by ILS_B-SP_B and $POP^b_{0.5}$ after 32 hours.

Instance	BKS	ILS_B-SP_B	$POP^b_{0.5}$
X-n303-50-k11	17739	**17728**	**17728**
X-n303-66-k13	18120	18120	18120
X-n303-80-k16	19603	19737	19613
X-n308-50-k9	22544	22615	22655
X-n308-66-k11	24154	24218	**24131**
X-n308-80-k12	25164	25243	25251
X-n313-50-k39	57762	57805	**57711**
X-n313-66-k44	60069	60028	**60018**
X-n313-80-k56	73834	73899	73850
X-n317-50-k27	43391	43397	43391
X-n317-66-k35	54502	54502	54502
X-n317-80-k43	63683	63683	63683
X-n322-50-k14	23309	23309	23309
X-n322-66-k19	25034	25034	25034
X-n322-80-k23	27500	27557	27500
X-n327-50-k10	21610	21731	21684
X-n327-66-k13	23322	23322	**23315**
X-n327-80-k16	24990	25089	25032
X-n331-50-k8	24152	24152	24152
X-n331-66-k10	26247	26302	26247
X-n331-80-k12	28265	28271	**28189**
X-n336-50-k45	81760	81823	**81650**
X-n336-66-k57	99226	**99101**	99146
X-n336-80-k68	116185	116216	116298

Continued on next page

Table N.2 – continued from previous page

Instance	BKS	ILS$_B$-SP$_B$	POP$^b_{0.5}$
X-n344-50-k22	28527	28563	28532
X-n344-66-k29	31837	31936	31871
X-n344-80-k35	35743	35765	35743
X-n351-50-k21	18584	18768	18621
X-n351-66-k26	19758	19854	19767
X-n351-80-k32	22158	22292	**22130**
X-n359-50-k15	33255	33833	**33241**
X-n359-66-k19	37695	37943	37766
X-n359-80-k23	43412	43785	43494
X-n367-50-k12	20526	20784	20784
X-n367-66-k14	21479	21491	21479
X-n367-80-k15	22386	22411	22397
X-n376-50-k47	80736	80736	80736
X-n376-66-k62	100613	100613	100613
X-n376-80-k75	119581	119581	119581
X-n384-50-k27	41206	41667	41214
X-n384-66-k35	47373	47315	**47246**
X-n384-80-k42	55386	**55361**	55373
X-n393-50-k19	30005	30273	30008
X-n393-66-k25	29340	29536	29392
X-n393-80-k31	32619	32755	**32576**
X-n401-50-k15	39746	40099	39814
X-n401-66-k20	47658	47737	**47603**
X-n401-80-k23	54270	54554	54293
X-n411-50-k14	17959	18020	17959
X-n411-66-k15	18785	18832	18801
X-n411-80-k17	19496	19635	**19483**
X-n420-50-k67	75497	75559	**75442**
X-n420-66-k86	76079	76099	76126
X-n420-80-k105	89356	89450	89356
X-n429-50-k31	41284	41545	**41191**
X-n429-66-k40	47793	48057	**47763**
X-n429-80-k48	54835	55084	54873
X-n439-50-k19	27010	27010	27010
X-n439-66-k25	28878	28883	28895
X-n439-80-k30	32074	32111	32080
X-n449-50-k15	36929	37394	**36922**
X-n449-66-k20	41846	42361	41846
X-n449-80-k23	46738	47224	**46714**
X-n459-50-k14	18879	18940	18925
X-n459-66-k18	20561	20635	**20521**
X-n459-80-k21	22047	22077	**21961**
X-n469-50-k70	123773	124309	**123360**
X-n469-66-k90	148455	148518	**148442**
X-n469-80-k109	178511	**178461**	178511
X-n480-50-k36	52309	52730	52428
X-n480-66-k47	63577	63733	**63522**
X-n480-80-k56	73993	74450	74252
X-n491-50-k30	43952	44454	**43907**
X-n491-66-k39	49627	50234	49637
X-n491-80-k47	56141	56546	**56036**

Continued on next page

Table N.2 – continued from previous page

Instance	BKS	ILS_B-SP_B	$POP^b_{0.5}$
X-n502-50-k20	40591	40641	**40557**
X-n502-66-k26	49285	49355	**49251**
X-n502-80-k31	56992	57001	**56977**
X-n513-50-k11	21675	21779	21687
X-n513-66-k14	22426	22467	22582
X-n513-80-k17	23448	23501	23468
X-n524-50-k125	154137	154137	154137
X-n524-66-k129	154416	154416	154446
X-n524-80-k132	154446	154446	154446
X-n536-50-k49	54658	55089	54697
X-n536-66-k64	66032	66335	**65937**
X-n536-80-k77	77811	78445	77822
X-n548-50-k25	53049	53160	**52898**
X-n548-66-k33	61415	61572	**61346**
X-n548-80-k40	71867	71907	**71803**
X-n561-50-k22	31826	32074	31860
X-n561-66-k28	34370	34717	34475
X-n561-80-k34	38053	38277	38060
X-n573-50-k22	40239	40763	40278
X-n573-66-k25	44151	44320	44179
X-n573-80-k27	47054	47286	47104
X-n586-50-k80	122632	122947	**122326**
X-n586-66-k105	140396	140693	**140284**
X-n586-80-k127	160390	160791	**160311**
X-n599-50-k47	65292	65744	**65093**
X-n599-66-k61	76472	76935	**76432**
X-n599-80-k74	89844	90263	**89663**
X-n613-50-k32	40838	41321	40873
X-n613-66-k41	46074	46483	**46023**
X-n613-80-k50	52096	52458	**52084**
X-n627-50-k22	38096	38483	38217
X-n627-66-k29	44782	44998	**44499**
X-n627-80-k35	52429	52550	**52223**
X-n641-50-k18	42333	42584	42449
X-n641-66-k23	47501	48000	**47410**
X-n641-80-k28	54116	54544	**54030**
X-n655-50-k66	59416	59542	**59388**
X-n655-66-k87	72456	72511	72456
X-n655-80-k105	86564	86575	86564
X-n670-50-k112	144688	**144685**	**144685**
X-n670-66-k117	144960	144960	**144882**
X-n670-80-k120	145151	145317	145221
X-n685-50-k43	48023	48489	48037
X-n685-66-k54	53240	53715	53279
X-n685-80-k62	59301	59825	59358
X-n701-50-k23	51390	52259	51668
X-n701-66-k30	58844	59626	**58816**
X-n701-80-k36	68618	69311	**68400**
X-n716-50-k18	29757	30210	**29638**
X-n716-66-k23	32527	32941	32632
X-n716-80-k28	37976	38270	38117

Continued on next page

Table N.2 – continued from previous page

Instance	BKS	ILS_B-SP_B	$POP^b_{0.5}$
X-n733-50-k83	80585	81110	**80408**
X-n733-66-k102	92156	92646	92162
X-n733-80-k125	110659	111008	**110536**
X-n749-50-k49	47740	48385	47811
X-n749-66-k63	55560	56064	**55398**
X-n749-80-k78	63991	64417	**63758**
X-n766-50-k58	95674	96805	95981
X-n766-66-k62	101566	102295	101574
X-n766-80-k65	106758	107612	**106644**
X-n783-50-k24	49027	50112	49043
X-n783-66-k31	53429	54424	54021
X-n783-80-k38	60937	61613	**60834**
X-n801-50-k20	48459	49377	**48404**
X-n801-66-k27	54929	55205	**54816**
X-n801-80-k32	62698	63054	**62605**
X-n819-50-k86	89296	89788	**89151**
X-n819-66-k112	108431	108682	**108216**
X-n819-80-k136	128617	129220	**128265**
X-n837-50-k71	116553	116842	**115892**
X-n837-66-k94	129183	129826	**128883**
X-n837-80-k114	154966	155395	**154678**
X-n856-50-k48	57777	58007	**57676**
X-n856-66-k63	63550	63791	**63355**
X-n856-80-k76	73802	74013	**73663**
X-n876-50-k30	58780	59411	**58657**
X-n876-66-k38	69617	70154	**69483**
X-n876-80-k46	80983	81569	**80753**
X-n895-50-k19	40668	41380	40669
X-n895-66-k25	44059	44680	44153
X-n895-80-k30	48451	48996	48501
X-n916-50-k105	189911	190197	**188372**
X-n916-66-k136	222807	223260	**222357**
X-n916-80-k165	263885	264590	**263723**
X-n936-50-k132	127445	127425	**127416**
X-n936-66-k138	128832	128863	**128588**
X-n936-80-k143	130736	131176	**130267**
X-n957-50-k44	57019	57289	**56719**
X-n957-66-k58	62593	62700	**62431**
X-n957-80-k70	71855	72036	**71725**
X-n979-50-k30	69739	70460	**69660**
X-n979-66-k39	84499	85163	**84392**
X-n979-80-k47	99605	100287	**99581**
X-n1001-50-k22	49978	51452	50462
X-n1001-66-k28	56126	57315	56667
X-n1001-80-k34	63278	64319	**63233**