Universidade Estadual de Campinas
Instituto de Computação

Lucas Porto Maziero

Exact and heuristic algorithms for covering routing problems

Algoritmos exatos e heurísticos para problemas de roteamento por cobertura

CAMPINAS
2023

Lucas Porto Maziero

Exact and heuristic algorithms for covering routing problems

Algoritmos exatos e heurísticos para problemas de roteamento por cobertura

Tese apresentada ao Instituto de Computação da Universidade Estadual de Campinas como parte dos requisitos para a obtenção do título de Doutor em Ciência da Computação.

Thesis presented to the Institute of Computing of the University of Campinas in partial fulfillment of the requirements for the degree of Doctor in Computer Science.

Supervisor/Orientador: Prof. Dr. Fábio Luiz Usberti
Co-supervisor/Coorientador: Prof. Dr. Celso Cavellucci

Este exemplar corresponde à versão final da Tese defendida por Lucas Porto Maziero e orientada pelo Prof. Dr. Fábio Luiz Usberti.

CAMPINAS
2023

Ficha catalográfica
Universidade Estadual de Campinas
Biblioteca do Instituto de Matemática, Estatística e Computação Científica
Silvania Renata de Jesus Ribeiro - CRB 8/6592

Maziero, Lucas Porto, 1992-
M457e Exact and heuristic algorithms for covering routing problems / Lucas Porto Maziero. – Campinas, SP : [s.n.], 2023.

Orientador: Fábio Luiz Usberti.
Coorientador: Celso Cavellucci.
Tese (doutorado) – Universidade Estadual de Campinas, Instituto de Computação.

1. Programação linear inteira. 2. Meta-heurística. 3. Matheurística. 4. Otimização combinatória. 5. Pesquisa operacional. I. Usberti, Fábio Luiz, 1982-. II. Cavellucci, Celso, 1951-. III. Universidade Estadual de Campinas. Instituto de Computação. IV. Título.

Informações Complementares

Título em outro idioma: Algoritmos exatos e heurísticos para problemas de roteamento por cobertura
Palavras-chave em inglês:
Integer linear programming
Metaheuristic
Matheuristic
Combinatorial optimization
Operations research
Área de concentração: Ciência da Computação
Titulação: Doutor em Ciência da Computação
Banca examinadora:
Fábio Luiz Usberti [Orientador]
Pedro Belin Castellucci
Pedro Henrique Del Bianco Hokama
Washington Alves de Oliveira
Kelly Cristina Poldi
Data de defesa: 11-12-2023
Programa de Pós-Graduação: Ciência da Computação

Identificação e informações acadêmicas do(a) aluno(a)
- ORCID do autor: https://orcid.org/0000-0002-9130-0791
- Currículo Lattes do autor: http://lattes.cnpq.br/6769959937671105

Universidade Estadual de Campinas
Instituto de Computação

Lucas Porto Maziero

Exact and heuristic algorithms for covering routing problems

Algoritmos exatos e heurísticos para problemas de roteamento por cobertura

Banca Examinadora:

- Prof. Dr. Fábio Luiz Usberti
  IC/UNICAMP
- Prof. Dr. Pedro Belin Castellucci
  INE/UFSC
- Prof. Dr. Pedro Henrique Del Bianco Hokama
  IMC/UNIFEI
- Prof. Dr. Washington Alves de Oliveira
  FCA/UNICAMP
- Profa. Dra. Kelly Cristina Poldi
  IMECC/UNICAMP

A ata da defesa, assinada pelos membros da Comissão Examinadora, consta no SIGA/Sistema de Fluxo de Dissertação/Tese e na Secretaria do Programa da Unidade.

Campinas, 11 de dezembro de 2023

Dedicatória

Essa obra é dedicada à minha mãe, Eliane, e aos meus avós, Delia e Antônio (*in memoriam*).

No que diz respeito ao empenho, ao compromisso, ao esforço, à dedicação, não existe meio termo. Ou você faz uma coisa bem feita ou não faz. *(Ayrton Senna)*

Agradecimentos

Agradeço a Deus, pela dádiva dessa vida e por me guiar durante a minha missão como ser humano. Nhá Chica, minha confidente, conselheira repleta de sabedoria.

Eliane, minha mãe, a que me deu a oportunidade dessa vida, me criou e transformou na pessoa que eu sou. A melhor Professora, que transmitiu os conhecimentos mais importantes e que nenhuma escola ou universidade pode oferecer.

Agradeço meus avós, Delia e Antônio (*in memoriam*), exemplos de seres humanos que irei carregar sempre. Meu caráter é reflexo de tudo o que vocês fazem/fizeram por mim desde que eu nasci.

Ana Júlia, minha esposa, que conheci e casei durante o doutorado. A pessoa que mais acredita em mim, incentivadora do meu trabalho e que nunca me deixa desistir. Eu não teria conseguido sem você. Obrigado por todo o apoio e paciência ao longo desse doutorado. Você me ensinou a ser uma pessoa melhor.

Fábio, meu orientador, que tem sido minha referência. Um exemplo de Professor e pessoa. Aprendi muito com você durante todos esses anos e agradeço por todo o suporte e compreensão. Seus ensinamentos foram essenciais na minha formação acadêmica, profissional e pessoal.

Celso, meu coorientador, por todas as contribuições indispensáveis para este trabalho. Seus ensinamentos foram muito importantes para minha evolução. Sou muito grato por ter sido orientado por você.

À minha família, pessoas que mais me encorajaram na busca dos meus sonhos. O suporte de vocês foi essencial durante este trabalho.

Agradeço o Breno, Lucas e Rogério, por todo o apoio incondicional, companheiros de estudos, dos momentos felizes e também das horas mais difíceis. Vocês são verdadeiras amizades que a UNICAMP me proporcionou.

Aos meus amigos de Três Corações e Campinas, que me incentivaram e acreditaram no meu trabalho durante todo o doutorado.

Agradeço o corpo docente do Instituto de Computação da UNICAMP, referências acadêmicas que tive a oportunidade de ser aluno e adquirir conhecimentos fundamentais para minha formação acadêmica.

Aos funcionários do Instituto de computação e da UNICAMP, por toda a dedicação em promover um ambiente acolhedor, humano e de excelência em pesquisa e desenvolvimento.

Agradeço, pelo suporte financeiro desta tese, o Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq), processo nº 140960/2017-1, e a Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES), código de financiamento 001. As hipóteses, análises, conclusões e recomendações contidas nessa tese são de responsabilidade dos autores e não refletem a visão do CNPQ e da CAPES.

Resumo

Os problemas de roteamento possuem diversas aplicações no mundo real, cada uma com restrições particulares como janelas de tempo, logística verde, disponibilidade de recursos e acessibilidade aos clientes. Quando se trata de disponibilidade de recursos e acessibilidade aos clientes, problemas de roteamento por cobertura são propostos para lidar com cenários reais em que regiões de difícil acesso possuem clientes que precisam ser atendidos, mas não podem ser visitados *in loco*. Os problemas de roteamento por cobertura visam encontrar rotas de custo mínimo cujas demandas dos clientes são atendidas servindo-os localmente ou remotamente com um veículo. Nesta tese, são propostas metodologias computacionais para dois problemas de roteamento por cobertura: *Covering Salesman Problem* (CSP) e *Multi-Depot Covering Tour Vehicle Routing Problem* (MDCTVRP). Além disso, dois novos problemas de roteamento por cobertura são introduzidos: *Capacitated Covering Salesman Problem* (CCSP) e *Electric Capacitated Covering Tour Problem* (ECCTP).

O CSP é uma generalização do Problema do Caixeiro Viajante em que o objetivo é encontrar um ciclo de custo mínimo tal que todos os vértices sejam visitados ou cobertos pelo ciclo. Nesta tese, desigualdades válidas da literatura são adaptadas para o CSP. Além disso, é proposto um novo conjunto de desigualdades válidas que se utilizam de aspectos únicos do problema. O primeiro *branch-and-cut framework*, com rotinas exatas e heurísticas para separar as desigualdades válidas, é proposto para resolver o CSP. Experimentos computacionais realizados em um *benchmark* de instâncias mostram que o *framework* proposto foi capaz de superar metodologias do estado-da-arte da literatura, além de obter soluções ótimas para diversas instâncias anteriormente não resolvidas.

O CCSP é introduzido nesta tese com a intenção de abordar restrições de cobertura em problemas de roteamento de veículos capacitados. O objetivo do CCSP é atender os clientes usando uma frota de veículos, minimizando a distância percorrida pelos veículos. Os veículos podem atender os clientes localmente ou remotamente. Metodologias computacionais baseadas em Programação Linear Inteira (PLI) e na meta-heurística *Biased Random-Key Genetic Algorithm* (BRKGA) são propostas para resolver o CCSP. Além disso, uma formulação de PLI oriunda do CCSP é estendida para resolver o MDCTVRP. Os resultados dos experimentos computacionais mostram que a nova formulação para o MDCTVRP superou a melhor abordagem exata existente até então. Especificamente, soluções ótimas foram obtidas para vários instâncias anteriormente não resolvidas.

No intuito de abordar um cenário real envolvendo logística verde, o ECCTP é introduzido nesta tese. O ECCTP é uma nova variante do problema de roteamento de veículos onde as demandas dos clientes podem ser atendidas por veículos elétricos com autonomia limitada e os veículos elétricos podem recarregar em postos de recargas alternativos. Os veículos podem atender os clientes localmente ou remotamente. Uma formulação de PLI mista e uma meta-heurística (BRKGA) são propostas para resolver o ECCTP. Com um *benchmark* de instâncias adaptadas da literatura, foram realizados experimentos computacionais com o objetivo de avaliar a eficácia da formulação e da meta-heurística.

Abstract

Routing problems have several real-world applications, each with particular constraints such as time windows, green logistics, resources availability and customers accessibility. When it comes to resources availability and customers accessibility, covering routing problems are proposed to deal with real-world scenarios in which there are hard-to-reach regions containing customers that need to be served, but cannot be visited on site. The covering routing problems aim to find minimum cost routes such that the demands of the customers are satisfied by serving them locally or remotely with a vehicle. In this thesis, computational methodologies are proposed for two known covering routing problems: the Covering Salesman Problem (CSP) and the Multi-Depot Covering Tour Vehicle Routing Problem (MDCTVRP). Furthermore, two new covering routing problem are introduced: the Capacitated Covering Salesman Problem (CCSP) and the Electric Capacitated Covering Tour Problem (ECCTP).

The CSP is a generalization of the Traveling Salesman Problem in which the objective is to find a minimum cost tour such that every vertex is visited or covered by the tour. In this thesis, valid inequalities from the literature are adapted to the CSP. In addition, a novel set of valid inequalities is proposed, which make use of unique aspects of the problem. The first branch-and-cut framework, with exact and heuristic routines to separate the valid inequalities, is proposed to solve the CSP. Computational experiments conducted on benchmark of instances show that the proposed framework was able to outperform state-of-the-art methodologies from literature, in addition to having proven optimal solutions for several previously unresolved instances.

The CCSP is introduced in this thesis with the intention of addressing covering constraints in capacitated vehicle routing problems. The objective of the CCSP is to serve customers using a fleet of vehicles, minimizing the distance traversed by the vehicles. The vehicles can serve customers locally or remotely. Computational methodologies based on Integer Linear Programming (ILP) and Biased Random-Key Genetic Algorithm (BRKGA) metaheuristic are proposed to solve the CCSP. Moreover, a CCSP ILP formulation is extended to solve the MDCTVRP. The results of the computational experiments show that the new formulation for the MDCTVRP outperformed the best existing exact approach so far. Specifically, optimal solutions were obtained for several previously unsolved instances.

In order to tackle a real-world scenario involving green logistics, the ECCTP is introduced in this thesis. The ECCTP is a new variant of the vehicle routing problem where customers demands can be served by a electric vehicles with limited autonomy and the electric vehicles can recharge at alternative fuel stations. The vehicles can serve customers locally or remotely. A mixed ILP formulation and a (BRKGA) metaheuristic are proposed to solve the ECCTP. With a benchmark of instances adapted from the literature, computational experiments were conducted with the purpose of evaluating the effectiveness of the formulation and metaheuristic.

List of Figures

List of Tables

Contents

Chapter 1

Introduction

1.1 Combinatorial optimization problems

In a Combinatorial Optimization Problem (COP), there is a collection of instances, a finite set of feasible solutions for each instance, and a cost function. A feasible solution is the one that meets all problem constraints.

A COP can be a minimization or maximization of the cost function. Without loss of generality, we consider a minimization problem, whose objective is to find a feasible solution such that the value of this solution, evaluated by the cost function, is minimum.

According to Papadimitriou and Steiglitz [29], an instance of a COP can be defined as a pair (F, c), where F is the domain of feasible solutions and c is the cost function mapped as:

$$c : F \to \mathbb{R}.$$

An optimal solution for a COP is a feasible solution $f \in F$ such that

$$c(f) \leqslant c(y), \qquad \forall y \in F.$$

Wolsey [44] presented an alternative definition for a COP as follows: let a finite set $N = \{1, ..., n\}$, weights c_j for each $j \in N$, and a family F of feasible subsets of N. The problem of finding a feasible subset of minimum weight can be modeled as:

$$\min_{S \subseteq N} \{\sum_{j \in S} c_j : S \in F\}.$$

A naive algorithm for solving a COP consists of enumerating all feasible solutions in order to obtain an optimal solution. However, this algorithm becomes intractable in practice, as many COPs have an exponential number of feasible solutions. Although there are COPs in which polynomial time algorithms are known, many of them are NP-hard problems where there is no polynomial algorithm, except if P = NP [29].

1.1.1 Exact methods for COPs

Bertsimas and Tsitsiklis [3] stated that a wide range of COPs can be modeled as integer linear programming problems. Consider matrices A, B, and vectors b, c, and d. The problem

$$
\begin{aligned}
&(MIP)\\
&Minimize \quad cx + dy,\\
&\text{subject to}\\
&Ax + By = b,\\
&x, y \geqslant 0,\\
&x \quad integer,
\end{aligned}
$$

is defined as Mixed Integer Programming (MIP) problem, as x and y are integer and continuous variables, respectively. In the case of having only integer variables, the problem

$$
\begin{aligned}
&(IP)\\
&Minimize \quad cx,\\
&\text{subject to}\\
&Ax = b,\\
&x \geqslant 0 \quad and \quad integer,
\end{aligned}
$$

is considered as Integer Programming (IP) problem. If all variables are restricted to 0-1, the problem

$$
\begin{aligned}
&(BIP)\\
&Minimize \quad cx,\\
&\text{subject to}\\
&Ax = b,\\
&x \in \{0,1\}^n,
\end{aligned}
$$

is called Binary Integer Programming (BIP) problem, where n is the dimension of the vector of variables x. It is known that there are no efficient algorithms to solve integer programming problems [3] unless P = NP [29]. Thus, different approaches are proposed for these problems, such as exact, approximation and heuristic methodologies.

The cutting plane method is an exact approach in which an integer programming problem is solved through its linear programming relaxation, i.e., the same problem but with its integrality conditions removed [3]. Consider the IP problem. Its linear programming

relaxation consists of

$$\begin{aligned} &Minimize \quad cx, \\ &\text{subject to} \\ &Ax = b, \\ &x \geqslant 0. \end{aligned}$$

The first step of the cutting plane method is to find an optimal solution x^* for the linear programming relaxation. The second step checks if x^* is integer, if so, x^* is also an optimal solution for the integer programming problem and then the method stops; otherwise, the method proceeds to the next step. The third and final step adds a linear inequality constraint to the linear programming relaxation such that x^* is cut and all integer solutions of the integer programming problem are preserved. After the third step, the method repeats the entire process. Algorithm 1 illustrates the cutting plane method.

Algorithm 1 Cutting plane method.

1: Find an optimal solution x^* for the linear programming relaxation
2: If x^* is integer, stop
3: Add linear inequality constraint to the linear programming relaxation, cutting x^* and preserving all integer solutions of the integer programming problem. Repeat step 1.

Another exact approach to solve integer programming problems is the Branch-and-Bound algorithm [3]. Based on divide and conquer technique, the Branch-and-Bound algorithm consists of exploring the set of feasible integer solutions, using bounds information to avoid examining the entire feasible set. Let F be the set of feasible solutions to the BIP problem

$$\begin{aligned} &Minimize \quad cx, \\ &\text{subject to} \\ &x \in F. \end{aligned}$$

The set F can be partitioned into subsets $\{F_1, ..., F_k\}$. Then, each F_i such that $1, ..., k$ can be solved separately

$$\begin{aligned} &Minimize \quad cx, \\ &\text{subject to} \\ &x \in F_i \quad i = 1, ..., k. \end{aligned}$$

For instance, the problem F_1 and F_2 after partitioning F could be $F_1 = \{x \in F : x_1 = 0\}$ and $F_2 = \{x \in F : x_1 = 1\}$, respectively. Each sub-problem can be solved recursively so that for F_1 we have two new sub-problems: $F_3 = \{x \in F_1 : x_2 = 0\} = \{x \in F : x_1 = x_2 = 0\}$ and $F_4 = \{x \in F_1 : x_2 = 1\} = \{x \in F : x_1 = 0, x_2 = 1\}$. This step of the algorithm is called branching and creates an enumeration tree of sub-problems, which Figure 1.1 shows as an example.

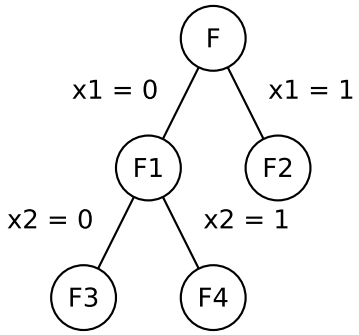


Figure 1.1: Branching step of a Branch-and-Bound algorithm.

Exploring the space of feasible solutions using branching strategy can lead to an enumeration tree with an exponential number of nodes. However, it is possible to use bounding information to implicitly enumerate the tree. Let $F = F_1 \bigcup F_2 \bigcup ... \bigcup F_k$ be a decomposition of F into subproblems and x_i the optimal solution of F_i for $i = 1, ..., k$. Then $\overline{x}_i$ and $\underline{x}_i$ are upper and lower bounds respectively for x_i. Therefore, $\overline{x} = \min_i \overline{x}_i$ is an upper bound of x and $\underline{x} = \min_i \underline{x}_i$ is a lower bound of x.

Using the upper and lower bounds, it is possible to prune branches of the enumeration tree that contains no optimal solution. The types of pruning are: optimality, bound and infeasibility pruning [44].

Figures 1.2 and 1.3 show decompositions of F into subproblems, where each subproblem has its corresponding upper and lower bounds.

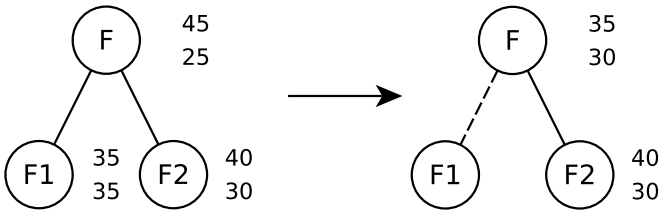


Figure 1.2: Example of pruning by optimality.

In the example of Figure 1.2, $\overline{x} = \min_i \overline{x}_i = \min\{35, 40\}$ and $\underline{x} = \min_i \underline{x}_i = \min\{35, 30\}$. Since the lower and upper bounds for the sub-problem F_1 are the same, the sub-problem F_1 can be pruned by optimally.

In Figure 1.3, $\overline{x} = \min_i \overline{x}_i = \min\{44, 40\}$ and $\underline{x} = \min_i \underline{x}_i = \min\{41, 30\}$. The lower bound for the sub-problem F_1 is $\underline{x}_1 = 41$. However, in the sub-problem F_2 we have an upper bound with cost $\overline{x}_2 = 40$, so the optimal solution will not be found through the branch of the sub-problem F_1, because $\underline{x}_1 > \overline{x}_2$. Therefore, the sub-problem F_1 is pruned by bound.

A sub-problem is pruned by infeasibility when its branch has no feasible solutions. For each sub-problem, the primal (upper) and dual (lower) bounds can be obtained by feasible solutions and linear programming relaxation, respectively.

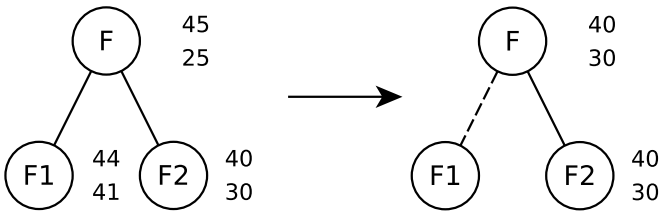


Figure 1.3: Example of pruning by bound.

The Branch-and-Cut algorithm is also an exact approach to solve integer programming problems [44]. It integrates the cutting plane method into the Branch-and-Bound algorithm so that cutting planes are generated during the tree enumeration. The idea of the Branch-and-Cut algorithm is to obtain tighter dual bounds for a sub-problem by adding cuts, thus reducing the number of nodes explored during the enumeration. Moreover, other techniques can be used inside the Branch-and-Cut algorithm, such as pre-processing procedures and primal heuristics.

During the design of a Branch-and-Cut algorithm, a trade-off should be considered. By adding many cuts for a sub-problem, the process of reoptimization can become slower than usual. While in the Branch-and-Bound algorithm only bounds information need to be kept, the Branch-and-Cut algorithm demands a cut pool in which all added cuts are stored. Besides the bounds information for the sub-problems, the Branch-and-Cut algorithm also requires a mechanism to indicate which constraints from the cut pool are needed for a determined sub-problem.

1.1.2 Heuristics methods for COPs

In contrast to exact methods, heuristic methods for COPs are capable of obtaining primal solutions without any guarantee of optimality. The main objective of heuristic methods is to provide good feasible solutions with less computational effort. Furthermore, heuristic methods are more appropriate to tackle hard and large size instances as it can be difficult for exact methods to find feasible solutions in such scenarios.

One of the heuristic methods is the constructive heuristic [28, 34, 21]. The main idea is to build a feasible solution for a COP progressively such that a single element is added to the solution at each iteration. In general, greedy algorithms and their variants are used to decide which element is added to the solution.

Local search procedures are heuristic methods that start from a feasible solution for a COP and explore iteratively the solution space [36, 38, 39]. The procedure consists of applying repeatedly local modifications to a solution in order to move to a neighbor solution, not necessarily feasible or with better cost, until a stopping criterion is reached. In general, the stopping criterion is based on a predefined number of iterations or when no further improvement can be made to a feasible solution.

Metaheuristics are the most employed and successful heuristic methods to addresses COPs. They consist of high-level procedures that combine diversification and intensifi-

cation strategies in order to explore the solution space effectively for a COP and find good feasible solutions. Many metaheuristics have been proposed to solve COPs in the literature, such as Genetic Algorithm [17, 37], Simulated Annealing [19, 41], Tabu Search [13, 23], Greedy Randomized Adaptive Search Procedures [30, 20] and Biased Random-Key Genetic Algorithm [14, 35].

Another heuristic method that have been studied recently in the literature for COPs are the Matheuristics [8, 22, 4]. By combining heuristic and exact methods, the main idea of Matheuristics is to provide hybrid methodologies that explore the solution space effectively, ensure that the solutions obtained are feasible and improve the cost by applying exact and heuristic approaches. Matheuristics have been applied to several COPs, specially for large size instances, in which exact methods cannot find feasible solutions with a reasonable computational effort.

1.2 Covering routing problems

The Traveling Salesman Problem (TSP) is one of the most famous and investigated COPs due to its various applications, such as distribution of goods or services [2]. Consider a single vehicle starting from a depot, visiting a set of customers and then returning to the initial depot. The objective of the TSP is to find a minimum route for that vehicle in terms of total travel time or distance.

Another well-known and studied COP is the Vehicle Routing Problem (VRP), introduced by Dantzig and Ramser [7]. Given a set of customers, the goal of the capacitated VRP is to serve all of them using a fleet of homogeneous vehicles located at a central depot, minimizing the total travel time or distance. Each vehicle must start and finish its route at the depot and cannot serve more than its capacity.

In addition to the applications of the original versions of the TSP and VRP, several variants of them have been introduced in the literature, for instance variants considering constraints of time windows, backhauls, and pickup-and-delivery [42]. In general, these variants assume that all customers are reachable by a vehicle. However, many applications can lead to scenarios where it is not possible to visit all customers due to limited time or lack of access to reach customers, such as the design of satellite distribution center network to supply humanitarian aid to the affected people in a disaster area [27, 33], hard-to-reach rural areas that need primary health care [16, 24], and disaster relief [9, 18]. To tackle these scenarios, covering routing problems have been introduced in the literature so that unvisited customers are served remotely by a vehicle within a reasonable distance.

The Covering Salesman Problem (CSP), one of the first covering routing problems introduced in the literature, was proposed by Current and Schilling [6]. Given a set of customers, the goal of the CSP is to find a minimum single-vehicle route such that all customers are served, either directly on the route or within a reasonable distance of another customer directly visited by the route. In the case each customer covers only itself, the CSP reduces to the TSP, which follows that CSP is also an NP-hard problem.

Gendreau et al. [12] studied the Covering Tour Problem, a variant of the CSP. In the CTP, the set of customers is divided into two groups. The first one, denoted as T,

are customers that must be visited. The second one, called W, are customers that must be covered. Also, there are generic locations in which the vehicle can visit. The goal of the CTP is to determine a minimum single-vehicle route such that every customer in T is visited and every customer in W is covered by a customer in T or a generic location visited by the vehicle. A multi-vehicle variant of the CTP, denominated the Multi-Vehicle Covering Tour Problem (m-CTP) was studied by Hachicha et al. [15]. The m-CTP generalizes the CTP by incorporating multiple vehicles, with each route limited to predefined length and number of customers or generic locations visited.

Allahyari et al. [1] proposed a covering routing problem called the Multi-Depot Covering Tour Vehicle Routing Problem (MDCTVRP). It combines the Multi-Depot Vehicle Routing Problem (MDVRP) [40] and CSP. In the MDCTVRP, each customer is served either directly on the route or remotely as long as its location is within a predetermined distance of another customer visited by the route. The depots have a fleet of homogeneous vehicles with limited capacity. Each vehicle must finish its route at the same depot from it started and cannot serve more than its capacity.

The scientific questions of Table 1.1 were addressed during the research. These questions are briefly answered in the next section, where the contributions obtained through the thesis are highlighted, and in more detail in the Chapters 2, 3 and 4.

Table 1.1: Research questions addressed during doctorate course.

Research problem	**Research question**
Exact methodology for the CSP	Is there an exact method more effective than the state-of-the-art for CSP?
Valid inequalities for the CSP	Is there a way to obtain specific valid inequalities for the CSP?
Multi-capacitated-vehicle CSP variant	Is it possible to develop effective methodologies for a Multi-capacitated-vehicle CSP variant?
Extend multi-capacitated-vehicle CSP variant exact methodology to the MDCTVRP	Is there an exact method more robust than the state-of-the-art for MDCTVRP?
Combine covering concept with green logistic	Is it reasonable to combine covering concept with green logistic in a new problem and develop methodologies to solve it?

1.3 Contributions

In this thesis, computational methodologies are explored with the objective of providing good solutions for different covering routing problems. These methodologies consist of exact and heuristic algorithms. The exact algorithms are based on Integer Linear Programming (ILP) formulations and the goal is to obtain optimal solutions and improve dual bounds. The heuristic algorithms are designed for scenarios that require solving large-size instances within a short computational time. With the intention of evaluating

both methodologies, extensive computational experiments are conducted using a large benchmark of instances.

Although effective heuristics methods have been proposed to solve the CSP, optimal solutions for many instances were unknown previous to this research. Besides, effective exact methods capable of improving the bounds, to the best of our knowledge, were not yet been proposed for the CSP. Therefore Chapter 2 describes a novel set of valid inequalities and the first branch-and-cut framework for the CSP to tackle this matter. Also, valid inequalities from the generalized traveling salesman problem are leveraged to the CSP. The branch-and-cut framework implements exact and heuristic routines to separate the valid inequalities. The computational experiments show that the new framework outperformed the state-of-the-art exact methodology from literature for the CSP. Among 48 instances from literature, only 9 instances had known optimal solutions. The new framework was able to obtain optimal solutions for all instances except one, representing a major step towards the exact solution of the CSP.

This thesis also introduces new covering routing problems. Chapter 3 proposes the CCSP, a new problem that combines the CSP and the VRP. The goal is to address covering constraints in a problem with multiple vehicles and limited capacities. Exact methodologies based on ILP are proposed for the CCSP, while the Biased Random-Keys Genetic Algorithm (BRKGA) metaheuristic is implemented to tackle large-size instances. Furthermore, an ILP formulation proposed for the CCSP is extended to solve the MDCTVRP. Computational experiments demonstrate that the new Mixed Integer Linear Programming (MILP) formulation for the MDCTVRP outperformed the state-of-the-art exact methodology. Several optimal solutions were proven for previously unsolved instances. In a benchmark of 280 instances from literature, the new MILP formulation obtained optimal solutions for 118 and improved all known lower bounds. These results show a substantial contribution to the state-of-the-art exact methodology for the MDCTVRP regarding bounds and optimal solutions.

The Chapter 4 proposes another covering routing problem called the Electric Capacitated Covering Tour Problem (ECCTP). In this problem, customers are served by a electric vehicle with limited autonomy and the vehicles can recharge at alternative fuel stations. The vehicles serve customers by visiting or covering them remotely. An exact approach based on MILP is proposed for the ECCTP, as well as a heuristic approach consisted of a BRKGA metaheuristic implementation. Besides, a benchmark of ECCTP instances are created based on instances originally proposed for the CVRP. The effectiveness of both exact and heuristic approaches was evaluated through computational experiments. The results indicate that the MILP formulation outperforms BRKGA in terms of optimal solutions for small instances. However, BRKGA proves to be more efficient regarding optimality gaps when addressing the more challenging instances.

Conference papers were also published and presented in the Brazilian Symposium of Operational Research conference in the years of 2016 and 2019, containing partial research results related to the Chapter 2 and 4. In 2016, the paper [26] presented preliminary studies about exact approaches for a CSP variant:

Title. *Um algoritmo branch-and-cut para o problema do ciclo dominante*

Authors. *Lucas Porto Maziero, Fábio Luiz Usberti, Celso Cavellucci*

Abstract. *This paper presents a study of a generalization of the Travelling Salesman Problem (TSP), called Dominant Cycle Problem (DCP). This generalization is the composition of two NP-hard problems: TSP and Set Problem k-Dominante in Graphs (k-DSP). The aim of the DCP is to find a minimum cost cycle in an undirected graph, from a source node. The cycle is trafficked by a traveler who needs to visit a set of customers (we dominant). The motivation of DCP study is in its practical application to power distribution companies, gas or water; in particular, the process of defining routes for reading the consumption of such services. In this paper, an integer linear programming formulation and branch-and-cut algorithm for the DCP are presented. The results obtained by exact methods are compared with the results obtained by metaheuristic already developed for the DCP.*

The proposed ILP formulation, valid inequalities and separation routines were assessed on a benchmark of instances generated randomly. Computational experiments showed that the branch-and-cut algorithm obtained better bounds than the ILP formulation, especially for instances with 200 nodes or more. In 2019, the paper [25] presented early ILP formulation and results for the ECCTP:

Title. *The Electric Capacitated Covering Tour Problem*

Authors. *Lucas Porto Maziero, Rafael Kendy Arakaki, Matheus Diógenes Andrade, Fábio Luiz Usberti*

Abstract. *This paper proposes a new problem based on the classic Vehicle Routing Problem (VRP). The problem is called Electric Capacitated Covering Tour Problem (ECCTP) and model a situation where customers can be serviced remotely by covering and only electric vehicles are employed. The ECCTP integrates two research areas: the covering problems and the green logistics. The problem is modeled by Mixed Integer Linear Program (MILP) and solved by a constructive heuristic. A set of benchmark instances is proposed and computational experiments were conducted.*

A constructive heuristic for the ECCTP was proposed and evaluated on a benchmark of instances based on a set of public instances originally proposed for the CVRP. The computational experiments show that the constructive heuristic obtained feasible solution for the ECCTP in a very short computational time.

Furthermore, contributions to other research areas were made during doctorate course. The first one is a publication in the XVIII Brazilian Symposium on Remote Sensing conference [10]. This work was done together with researchers from Fundação de Ciência, Aplicações e Tecnologias Espaciais and Embrapa Informática Agropecuária:

Title. *Otimização de um banco de dados geográficos utilizando PostGIS*

Authors. *João Luís dos Santos, João Francisco Gonçalves Antunes, Júlio César Dalla Mora Esquerdo, Alexandre Camargo Coutinho, Lucas Porto Maziero*

Abstract. *The pervasive usage of geoprocessing tools between researchers, students and users' communities leads to resources integration and allowed online tools to accomplish demands of people's daily life. With the progress of Information and Communications Technology (ICT), new computational resources are constantly becoming available and the effort of research institutes goes towards to release online interactive systems to the public that allow geospatial data visualization and manipulation. However, behind all provided resources there are a lot of process dealing with data preparation, storage and recovery. In this way, this work was carried out to deal with these tasks, implementing logical models of geographic database in order to store land use and land cover maps produced for the Brazilian Legal Amazon. The dataset used to handle the experiments was obtained from TerraClass initiative, a project created to qualify the deforested areas for that region. The main objective of this study was to identify which thematic classes intersect with a set of municipalities in the state of Pará, returning results in an acceptable time window for a web application. To accomplish that, a PostGIS spatial database was established and geoprocessing techniques and spatial operators were used to perform the tasks. To speed-up the queries execution time, a new model was proposed, reorganizing data in a new object structure which results in an 80 times faster performance of data queries. Thereafter, indexes were implemented and the cost of query execution was optimized more than 50%, addressing an important issue regarding a multi-user environment scenario.*

A new logical model for a geographic database was proposed in order to store information about the land-use in the Brazilian Legal Amazon. Moreover, a indexing strategy was developed with the goal of optimizing the runtime for queries made on the database. The computational experiments show that this new approach is promising for multi-user scenarios where the database must be able to manage several simultaneous requests.

The second and last contribution made to another research area is a publication in the XXXVIII Brazilian Symposium on Computer Networks and Distributed Systems conference [31]. This work was developed together with researchers from University of Campinas, University of Brasilia, Federal University of Rio Grande do Norte and Federal University of Bahia:

Title. *Protocolo baseado em geometria computacional para descoberta de cache em redes veiculares de dados nomeados.*

Authors. *Lucas Borges Rondon, Lucas Porto Maziero, Geraldo Pereira Rocha Filho, Augusto José Venâncio Neto, Maycon Leone Maciel Peixoto, Leandro Aparecido Villas*

Abstract. *Efficient content distribution in large scale ad hoc vehicular networks (VANETs) is extremely challenging due to the highly topology dynamics that VANETs impose. The Vehicle Named Data Network (VNDN) architecture addresses the performance and reliability challenges of delivering large-scale content delivery across VANETs,*

by supporting content-centric network communication and caching capabilities. However, the success of VNDNs depends critically on mitigating the transmission packet storm of interest occurrence in the cache discovery process, which results in network performance degradations due to the waste of resources generated. In light of this, this paper proposes a new geometric cache discovery protocol (PERSEU), which aims to pave the way for efficient large-scale content distribution in VNDNs, through the ability to mitigate the broadcast storm problem in VNDN cache discovery. Compared to other literature solutions, the PERSEU protocol enhances the cache discovery step by 337.7% while allowing a content delivery rate of 81.8% while reducing the number of streams in the cache discovery process at 82.7%.

In order to tackle the broadcast storm problem during vehicle communication in a VNDN, a new geometric cache discovery protocol was developed. This protocol, called PERSEU, was based on the convex hull definition. The interest package transmitting vehicles are chosen after applying an algorithm to find the convex hull according to the vehicles coordinates. Computational experiments show that PERSEU, when compared to other protocols, obtained better performance for cache discovering and content delivery hit, as well as reduced the amount of interest packet transmissions.

1.4 Structure of the thesis

This thesis is a composition of research papers written by the author in conjunction with other researchers throughout the course of the doctoral program. Chapter 2 contains a published paper detailing valid inequalities and a branch-and-cut framework with exact and heuristic separations routines for the Covering Salesman Problem (CSP). Chapter 3 shows a preprint proposing a new covering routing problem called the Capacitated Covering Salesman Problem (CCSP) in which ILP formulations, a Biased Random-Keys Genetic Algorithm (BRKGA) and a matheuristic are proposed to solve this new problem. Moreover, the ILP formulation for the CCSP is extended to solve the Multi-Depot Covering Tour Vehicle Routing Problem (MDCTVRP). Chapter 4 presents a published chapter book regarding a new variant of the Vehicle Routing Problem (VRP) that permits customers to be serviced by electric vehicles. A MILP formulation and a BRKGA metaheuristic are employed to solve this new variant. Chapter 5 consists of a discussion about the exact and heuristic approaches developed for the problems and the results achieved by them. Chapter 6 holds the final remarks about the covering routing problems studied, the computational methodologies used to tackle these problems, the scientific contributions obtained through this thesis and the promising future works.

Chapter 2

Branch-and-cut algorithms for the covering salesman problem

The paper presented next is a full article published in RAIRO - Operations Research in 2023 and it is co-authored with Fábio Luiz Usberti and Celso Cavellucci (DOI: `https://doi.org/10.1051/ro/2023055`). In this text we present the first branch-and-cut framework for the covering salesman problem in which a new family of valid inequalities is derived. Exact and heuristic separation algorithms are implemented for the new family of inequalities. Computational studies show the proposed branch-and-cut framework outperforming the state-of-the-art exact methodology.

The Covering Salesman Problem (CSP) is a generalization of the Traveling Salesman Problem in which the tour is not required to visit all vertices, as long as all vertices are covered by the tour. The objective of CSP is to find a minimum length Hamiltonian cycle over a subset of vertices that covers an undirected graph. In this paper, valid inequalities from the generalized traveling salesman problem are applied to the CSP in addition to new valid inequalities that explore distinct aspects of the problem. A branch-and-cut framework assembles exact and heuristic separation routines for integer and fractional CSP solutions. Computational experiments show that the proposed framework outperformed methodologies from literature with respect to optimality gaps. Moreover, optimal solutions were proven for several previously unsolved instances.

2.1 Introduction

Consider a set of sites scattered in the plane that must be covered by a single-vehicle tour. Knowing that each site covers some of its neighbors, what is the minimum length of an enclosed vehicle tour in which all sites are covered? This question is addressed by the Covering Salesman Problem (CSP), proposed by Current and Schilling [6] in 1989. More formally, given an undirected graph, the CSP objective is to find the shortest Hamiltonian cycle on a subset of vertices that covers the graph. The special case where each vertex covers strictly itself is the Travelling Salesman Problem (TSP) [2], which follows that CSP

is also NP-hard.

Since its proposal, the CSP has attracted the attention of researchers due to its complexity and numerous applications. These applications arise in scenarios where it is unrealistic to visit all locations, e.g., rural health services, areas affected by natural disasters, or planning mobile service units [6].

Several variants of CSP were investigated. Current et al. [5] studied the Shortest Covering Path Problem (SCPP). The goal is to find a minimum cost s-t path in a network that covers all vertices. The authors proposed two methods to solve the SCPP: a Lagrangian relaxation and a branch-and-bound algorithm that makes use of the obtained dual bounds.

Current and Schilling [7] introduced two bi-criterion routing problems: the Median Tour Problem (MTP) and the Maximal Covering Tour Problem (MCTP). Assuming a network with n vertices and a value p ($p \leqslant n$), the criteria for both problems are (i) to find a minimum length tour that visits exactly p of the n vertices and (ii) to maximize the accessibility of the vertices that are not in the tour. The problems differ in the way the accessibility of the second criterion is evaluated. In MTP, the second criterion is to minimize the sum of distances from each unvisited vertex to its closest vertex in the tour. In MCTP, the second criterion is to minimize the number of uncovered vertices. The authors proposed mathematical formulations and heuristics to solve both MTP and MCTP. Their methodologies were tested on a real-life scenario requiring the optimal location and sequence of stops for overnight mail service.

Another variant of CSP, studied by Gendreau et al. [13], is the Covering Tour Problem (CTP). Let $G = (V \cup W, E)$ be an undirected graph, where $V \cup W$ is the set of vertices and E is the set of edges. Vertex v_0 is the depot, V is the set of vertices that can be visited, $T \subseteq V$ is the set of vertices that must be visited ($v_0 \in T$), and W is the set of vertices that must be covered but cannot be visited. The goal of the CTP is to determine a minimum length tour that visits a subset of vertices $S \subseteq V$ such that $T \subseteq S$ and each vertex of W is covered by some vertex in S. The authors proposed heuristics and a branch-and-cut algorithm to solve the CTP.

Golden et al. [14] proposed a generalized version of the CSP called the Generalized Covering Salesman Problem (GCSP). Given an undirected graph $G = (V, E)$, each vertex $i \in V$ has a covering demand k_i, meaning vertex i has to be covered at least k_i times. In addition, there is a fixed cost F_i that incurs when the tour visits vertex i. The objective of the GCSP is to minimize the solution cost which is given by the sum of the tour length and the costs of the visited vertices. The authors developed local searches that explore exchange, removal, and insertion of tour vertices to escape from local optimum.

Another similar problem to the CSP is the Generalized Traveling Salesman Problem (GTSP). In GTSP, the vertices are partitioned into disjoint subsets, called clusters, and the goal is to determine the minimum length tour that visits exactly one vertex from each cluster. The GTSP is a special case of the CSP, where each cluster can be modeled as a subset of vertices that mutually cover themselves. Fischetti, González, and Toth [12] propose a branch-and-cut algorithm based on exact and heuristic separation routines for some families of valid inequalities for the GTSP. These inequalities are translated for the CSP in Section 2.3.

Zhang and Xu [26] proposed the online CSP, where the vehicle will face up to k blocked edges not known a priori during its tour traversal. The objective is to find a minimum length tour that covers all vertices while bypassing the blocked edges. The authors presented a $(k + \alpha)$-competitive algorithm, where $\alpha = \frac{1}{2} + \frac{(4k+2)r}{OPT} + 2\upsilon\rho$, υ is the approximation ratio for the Steiner Tree Problem, ρ is the maximum number of vertices that can cover an arbitrary vertex and r is the radius which defines the covering neighbourhood of each vertex.

Many works in literature have given attention to the geometric CSP, also known as the Close Enough Traveling Salesman Problem (CETSP). In this version, each vertex has its neighborhood defined as a compact region of the plane. The goal is to find a minimum length tour that starts from a depot and intercepts all neighborhood sets, thus covering all its corresponding vertices. Approximation algorithms, heuristics and methodologies based on ILP were developed for this version (Dumitrescu and Mitchell [11], Gulczynski et al. [15], Dong et al. [10], Shuttleworth et al. [21], Behdani and Smith [3], Coutinho et al. [4]).

Table 2.1 emphasizes the main differences between CSP and its counterparts. In CTP, among the vertices that can be visited, for some of them the visitation is mandatory. As for the vertices that must be covered, in CTP these vertices cannot be visited. In GTSP, the vertices are clustered into disjoint neighborhoods, meaning each vertex covers exactly the vertices in the cluster it belongs. The vertices in GCSP may require multiple coverings and each visitation incurs into a fixed cost. Finally, in CETSP the vertices are covered by a compact region on the plane instead of being covered by a subset of vertices. All of these problems, despite sharing the idea of joining vehicle routing with set covering, contain important distinctions with respect to CSP. This explains why this problem still requires customized exact and heuristic methodologies.

Table 2.1: Summary of the main differences between CSP, CTP, GTSP, GCSP and CETSP.

	Required/Forbidden visitations	**Disjoint neighborhoods**	**Multiple coverings**	**Geometric neighborhood**
CSP	✗	✗	✗	✗
CTP	✓	✗	✗	✗
GTSP	✗	✓	✗	✗
GCSP	✗	✗	✓	✗
CETSP	✗	✗	✗	✓

Some solution methodologies were proposed in the literature for the CSP. Current and Schilling [6], for example, developed a two-step heuristic to solve the CSP: the first step solves a set cover problem; the second step solves the TSP on the vertices determined by the first step. More than two decades later Salari and Naju-Azimi [19] revisited the problem by proposing a heuristic for the CSP embedded within an Integer Linear Programming (ILP) framework. First, they employ constructive heuristics to find good initial solutions and then the tour vertices are rearranged by the use of ILP techniques in an attempt to reduce its length. Salari et al. [20] gave a polynomial-size formulation and a hybrid heuristic for the CSP, which combines ant colony optimization and dynamic programming. The formulation of Salari et al., to the best of our knowledge, composes the state-of-the-art exact methodology for the CSP.

Venkatesh et al. [24] proposed a Multi-Start Iterated Local Search (MS-ILS) algorithm for the CSP with a variable degree of perturbation. Computational results show that the proposed approach is competitive with other state-of-the-art heuristic approaches. Zang et al. [25] reformulated the CSP as a bilevel CSP with a leader and a follower sub-problem and proposed two Parallel Variable Neighborhood Search (PVNS) heuristics, namely, synchronous "master–slave" PVNS and asynchronous cooperative PVNS. Computational results show that the PVNS has improved previously best known solutions. Pandiri et al. [17] developed two hybrid metaheuristic approaches for the CSP. The first is based on the artificial bee colony algorithm and the second is based on the genetic algorithm. Both approaches were competitive with the state-of-the-art heuristics. Lu et al. [16] presented a Hybrid Evolutionary Algorithm (HEA) that assembles a crossover operator with solution reconstruction, a destroy-and-repair mutation operator and a two-phase tabu search. The HEA also uses Lin-Kernighan local search on multiple stages. Computational results show that for 21 out of the 27 large instances, the HEA has improved previously best known solutions, while for small and medium instances the HEA has achieved previously best known solutions.

Recent applications of the CSP concern the routing of drones, which involves additional operational constraints. For example, Vásquez et al. [23] studied the Travelling Salesman Problem with Drone (TSP-D). Given a complete digraph, each node must be visited either by a vehicle route starting and ending at a depot or by a drone executing dispatches to customers from the vehicle during the route. When a drone visits a customer, it must fly back to meet the vehicle at a scheduled location on its route. The objective is to minimize travel time. The authors proposed a mixed integer programming formulation and valid inequalities for the TSP-D, solving it by Bender's decomposition using a two-stage approach: first selecting a subset of customers to be visited by the vehicle and then defining where the drone will be dispatched for the remaining customers. Computational experiments validated the performance of the algorithm using a benchmark of instances randomly generated.

A generalization of the TSP-D that considers more than one drone is the Travelling Salesman Problem with Multiple Drones (TSP-MD). Tiniç et al. [22] proposed a flow-based and a cut-based mixed integer linear programming formulations for TSP-MD. Also, branch-and-cut algorithms were developed for relaxations of the proposed formulations. The computational experiments showed that the branch-and-cut approaches outperformed the flow-based formulation. A sensitivity analysis was also made with various problem parameters, exploring scenarios such as vehicle operating cost, speed, and drone endurance.

The Flying Sidekick Travelling Salesman Problem (FSTSP) is another drone routing problem similar to the TSP-D. In FSTSP, there are some nodes that can be visited only by the vehicle, and each drone flight is limited by battery duration. Dell'Amico et al. [8] proposed a branch-and-bound exact algorithm and a heuristic, testing them on instances from the literature. The results showed that the branch-and-bound algorithm was effective in solving small instances while the heuristic approach was able to produce high-quality solutions for larger instances.

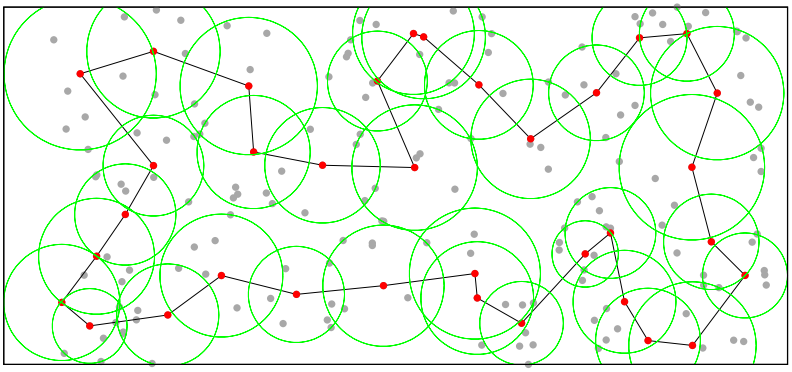
Our contributions Despite being well studied in the point of view of heuristics, the CSP still lacks effective exact methods. Many of the current best-known solutions still had not been proven optimal or had an open optimality gap due to the absence of a dual bound. The first branch-and-cut framework is proposed for the CSP to address this matter. The framework employs exact and heuristic routines to separate families of valid inequalities, some from the GTSP and others original for the CSP. Computational experiments performed on a benchmark set of instances compare our methodology with the state-of-the-art exact methodology from literature. Previous to this work, from a set of 48 instances, for only 9 instances there were proven optimal solutions. Our methodology improves this by certifying optimality for all except one instance. This represents a major contribution to the current body of knowledge regarding exact approaches on the CSP.

This paper is organized as follows. Section 2.2 formally defines the CSP and presents an integer linear programming formulation. Section 2.3 shows new valid inequalities for the CSP. Section 2.4 describes separation routines for the proposed valid inequalities, which constitute the branch-and-cut framework. In Section 2.5 computational experiments are conducted on a benchmark of instances, and results are analyzed and discussed. Section 2.6 gives the concluding remarks.

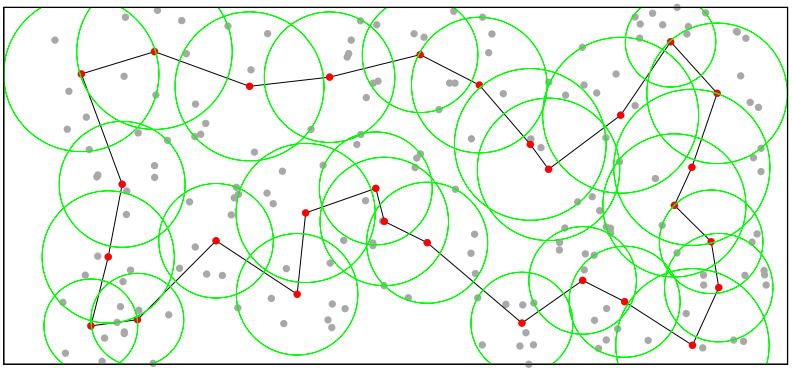
2.2 Problem Description and Formulation

The CSP can be formally stated as follows. Consider an undirected graph $G(V, E)$, where V is the set of vertices and E is the set of edges. Each edge $e \in E$ is associated with a non-negative cost c_e. For each vertex $v \in V$, $C(v)$ is the set of vertices that cover v and $D(v)$ is the set of vertices that are covered by v. It is considered that $v \in C(v)$ and $v \in D(v)$, $\forall v \in V$. An optimal solution to the CSP is a minimum length Hamiltonian cycle (tour) over a subset of vertices that covers all vertices in G. Figures 2.1a, 2.1b and 2.1c show optimal solutions for three CSP instances with 200 vertices.

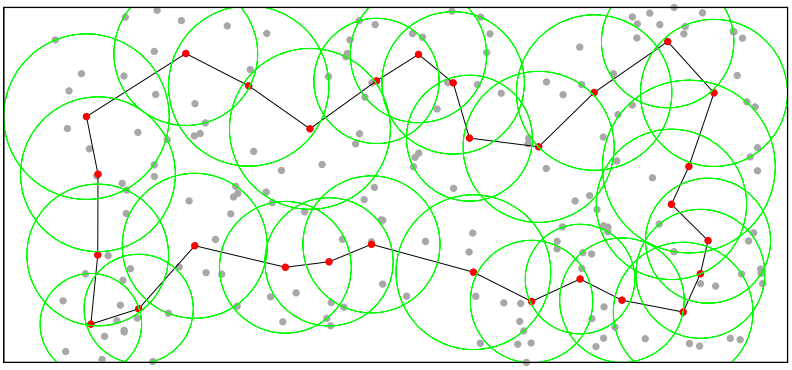
An Integer Linear Programming (ILP) formulation for the CSP is presented. Binary variable x_e indicates if an edge $e \in E$ belongs (1) or not (0) to the tour and binary variable y_v represents if a vertex belongs (1) or not (0) to the tour. We denote $\delta(v)$ the set of edges incident to $v \in V$, $\delta(S)$ the set of edges with one endpoint in $S \subset V$ and the other in $V \backslash S$ and $E(S)$ the set of edges with both endpoints in S.



(a) Instance kroB200-7



(b) Instance kroB200-9



(c) Instance kroB200-11

Figure 2.1: Optimal solutions for instances kroB200-7, kroB200-9, and kroB200-11, where each vertex covers its closest 7, 9, and 11 neighbors, respectively. Highlighted vertices belong to the tour and their covering sets are represented by circumferences.

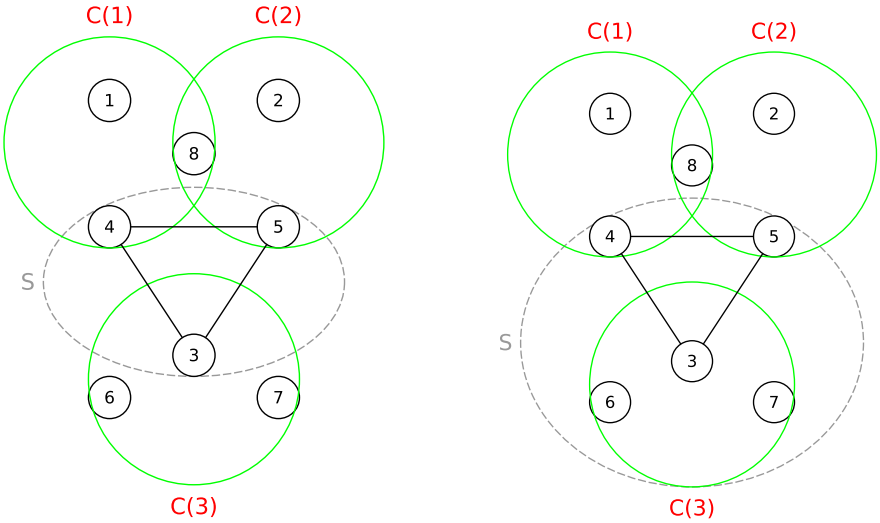
$$
\begin{aligned}
&(CSP)\\
&MIN \quad \sum_{e \in E} c_e x_e, && && (1)\\
&\text{subject to}\\
&\sum_{e \in \delta(v)} x_e = 2y_v && \forall v \in V, && (2)\\
&\sum_{i \in C(v)} y_i \geqslant 1 && \forall v \in V, && (3)\\
&\sum_{e \in \delta(S)} x_e \geqslant 2(y_i + y_j - 1) && \forall S \subset V, i \in S, j \in V \backslash S, && (4)\\
&x_e \in \{0,1\} && \forall i,j \in V, && (5)\\
&y_v \in \{0,1\} && \forall i \in V. && (6)
\end{aligned}
$$

The CSP formulation is based on the ideas of Fischetti, González, and Toth [12] for the GTSP. The objective function (1) minimizes the cost of a solution given by the sum of the costs of its edges. Constraints (2) ensure the number of edges incident at a vertex is 2 (if v is in the tour) or 0 (otherwise). Constraints (3) impose that each vertex must be covered at least once. Constraints (4) are subtour elimination constraints which state that every cut separating two vertices in the tour contains at least two edges.

2.3 Valid Inequalities

This section presents valid inequalities proposed by Fischetti, González, and Toth [12] for the GTSP, and here translated for the CSP. It is worth reminding that the GTSP is a special case of the CSP in which the vertices are partitioned into clusters, and each cluster is formed by vertices which mutually cover themselves, i.e., any two vertices u and v from the same cluster would have $C(u) = C(v)$.

Let $D(S)$ be the union of sets $D(v)$ for all $v \in S$, i.e., $D(S) = \bigcup_{v \in S} D(v)$ and let $\gamma(V)$ be the family of all the subsets S of vertices that contains $C(v)$ for at least one vertex $v \in S$, i.e., $\gamma(V) = \{F \subseteq \mathcal{P}(V) : \forall S \in F, \exists v \in S, C(v) \subseteq S\}$ where $\mathcal{P}(V)$ is the power set of V. To exemplify the concept of $\gamma(V)$, consider sets $C(1) = \{1,4,8\}$, $C(2) = \{2,5,8\}$ and $C(3) = \{3,6,7\}$ as shown in Figure 2.2. As exemplified in Figure 2.2a, if $S = \{3,4,5\}$, then none of the sets $C(1)$, $C(2)$ and $C(3)$ is a subset of S, thus $S \notin \gamma(V)$. However, if $S = \{3,4,5,6,7\}$, then set $C(3)$ is contained in S, thus $S \in \gamma(V)$, as shown in Figure 2.2b. The following family of inequalities are valid for the CSP:



(a) $S \notin \gamma(V)$ (b) $S \in \gamma(V)$

Figure 2.2: Example of $S \notin \gamma(V)$ (a) and $S \in \gamma(V)$ (b).

$$\sum_{e \in \delta(S)} x_e \geqslant 2 \qquad \forall S \in \gamma(V) : D(S) \neq V, \tag{7}$$

$$\sum_{e \in \delta(S)} x_e \geqslant 2y_i \qquad \forall S \notin \gamma(V) : D(S) \neq V, i \in S, \tag{8}$$

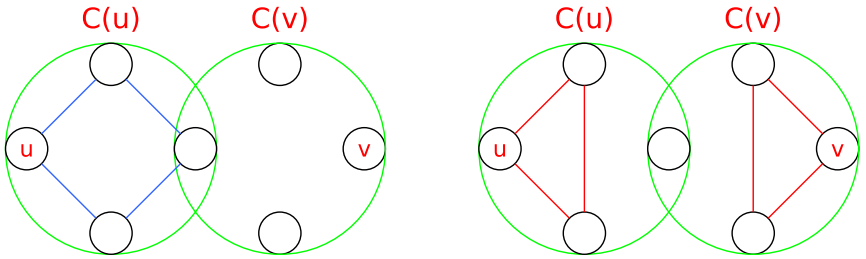
$$\sum_{e \in \delta(S)} x_e \geqslant 2(y_i + y_j - 1) \qquad \forall S \notin \gamma(V) : D(S) = V, i \in S, j \in V \backslash S. \tag{9}$$

Inequalities (7) ensure that each cut separating two sets $C(v)$ and $C(w)$ must be crossed at least twice. Inequalities (8) imply that each cut separating one vertex in the tour and one set $C(v)$ must be crossed at least twice. Inequalities (9) ensure that each cut separating two vertices in the tour must be crossed at least twice. Originally in GTSP, inequalities (7), (8), and (9) were applied to every subset of vertices containing at least one cluster, i.e., any subset of family $\gamma(V)$.

In the following, a new family of valid inequalities is proposed to consider a scenario particular to the CSP.

2.3.1 Cover Intersection Inequalities

Consider the case in which two covering sets $C(v)$ and $C(u)$ strictly overlap for some pair of vertices v and u, i.e., $C(v) \cap C(u) \neq \emptyset$ and $C(v) \neq C(u)$. This is a distinguished scenario for the CSP, and it does not occur on the GTSP since in that problem the clusters are disjoint. The new valid inequalities extend the idea of inequalities (7), in the sense of requiring a minimum weight for any edge cut-set separating two covering sets. However, to address the overlap of covering sets, the new valid inequalities (10) also take into account the edge cut-set weight of the intersection $C(v) \cap C(u)$.



(a) Feasible solution (b) Infeasible solution

Figure 2.3: Example of feasible (a) and infeasible (b) solutions in the context of overlap of covering sets.

For the following new valid inequalities (10), consider $S_v = S \cap C(v)$ for any $v \in V$. These inequalities are here called *CI inequalities* (cover intersection inequalities), and they only require a proper subset $S \subset V$ such that $S \in \gamma(V)$, which means it can be employed even if $D(S) = V$, another case in which inequalities (7) cannot be employed.

$$(CI\ inequalities)$$
$$\sum_{e \in (\delta(S) \bigcup \delta(S_v))} x_e \geqslant 2 \qquad \forall v \in V, \forall S \subset V : S \in \gamma(V) \qquad (10)$$

According to constraints (3), for any given vertex v, at least one vertex of $C(v)$ must be visited by the tour. In other words, for any subset $S \subset V$ such that $S \in \gamma(V)$, the tour must visit S_v or $C(v) \setminus S_v$. If set S does not intersect with $C(v)$, then S_v is empty, and (10) reduces to (7). Otherwise, S_v is not empty, and in this case, to satisfy constraints (3), the solution must contain at least two edges in either $\delta(S_v)$ or $\delta(S \setminus S_v)$. Figures 2.3a and 2.3b consider $V = C(u) \cup C(v)$ and $S = C(u)$. In this case, $S \in \gamma(V)$, and S_v contains a single vertex since $|C(u) \cap C(v)| = 1$. In Figure 2.3a, a feasible solution is presented such that, even though the cut-set $\delta(S)$ is empty (the tour is inside S), the edge cut-set $\delta(S_v)$ contains two edges, guaranteeing that node v is covered. An infeasible solution is presented in Figure 2.3b such that both edge cut-sets $\delta(S)$ and $\delta(S_v)$ are empty.

2.4 Branch-and-cut framework

This section presents the separation routines for inequalities (7)-(10). Sections 2.4.1 and 2.4.2 present the separation routines for integer and fractional solutions, respectively. In the following sections, consider $\{\mathbf{x^I}, \mathbf{y^I}\}$ and $\{\mathbf{x^F}, \mathbf{y^F}\}$ as integer and fractional solutions for the CSP formulation without the subcycle elimination constraints (4) but possibly including some of the valid inequalities (7)-(10). Also, let $G^I(V^I, E^I)$ and $G^F(V^F, E^F)$ be the graphs induced by $\{\mathbf{x^I}, \mathbf{y^I}\}$ and $\{\mathbf{x^F}, \mathbf{y^F}\}$, respectively. In G^I, every vertex $v \in V^I$ has a weight y_v, such that $y_v \in \mathbf{y^I}$, and every edge $e \in E^I$ has a cost x_e, such that $x_e \in \mathbf{x^I}$. Similarly, in G^F, every vertex $v \in V^F$ has a weight y_v, such that $y_v \in \mathbf{y^F}$, and every edge

$e \in E^F$ has a cost x_e, such that $x_e \in \mathbf{x^F}$.

2.4.1 Separation routine for integer solutions

The proposed separation routine searches, in a lazy constraint fashion, for inequalities (7-10) that are possibly violated by an integer solution $\{\mathbf{x^I}, \mathbf{y^I}\}$. First, the routine performs a depth-first search in G^I to check for the existence of illegal subcycles.

Let $S \subset V$ be the vertices of an illegal subcycle in G^I. To apply inequality (7) or (10) with respect to set S, it is necessary that $S \in \gamma(V)$. If this is not the case, the proposed routine attempts to augment S into S_{aug} by including the set $C(v)$ for some $v \in S$. However, the choice of which $C(v)$ will be included in S_{aug} is relevant to the effectiveness of the corresponding inequalities, as will be explained next.

Consider Figure 2.4, which shows a solution formed by two subcycles in graph G^I. In this figure $C(v_4) = \{v_4, v_8\}$, $C(5) = \{v_5, v_7\}$, and $C(v_6) = \{v_6, v_9\}$. By taking the illegal subcycle represented by $S = \{v_4, v_5, v_6\}$, it is not possible to apply inequalities (7) or (10), since $S \notin \gamma(V)$. By taking $S_{aug} = S \cup C(5)$, then $S_{aug} \in \gamma(V)$, however S_{aug} would not generate an effective cut, since vertex $v_7 \in V^I$. Otherwise, effective cuts can be derived from $S_{aug} = S \cup C(v_4)$ or $S_{aug} = S \cup C(v_6)$. Therefore, for an inequality (7) or (10) associated with S_{aug} to be effective in cutting solution $\{\mathbf{x^I}, \mathbf{y^I}\}$, set S_{aug} cannot contain any vertex in $V^I \setminus S$.

Algorithm 2 presents the implementation details of the separation routine for integer solutions, which searches for inequalities (7-10) associated with each subcycle found in $\{\mathbf{x^I}, \mathbf{y^I}\}$. The overall complexity of Algorithm 2 is bounded by $O(V^3)$.

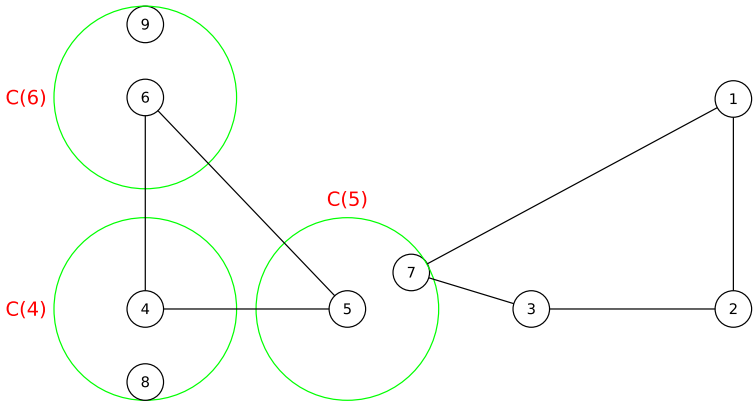


Figure 2.4: Example of an invalid CSP solution with two subcycles.

2.4.2 Exact separation routine for fractional solutions

This section gives the exact separation routines of inequalities (7-10) for a fractional CSP solution $\{\mathbf{x^F}, \mathbf{y^F}\}$. In particular, the separation of inequalities (7), (8) and (9) follows the methodology proposed by Fischetti, González and Toth [12] for the GTSP. As for the CI inequalities (10), a transformation of the solution graph G' is proposed to tackle the overlap of covering sets. The routines are described next.

Algorithm 2 Separation routine for integer solutions.

Input: graph $G^I(V^I, E^I)$ induced by an infeasible integer solution $\{\mathbf{x^I}, \mathbf{y^I}\}$ for the CSP.

Output: a set T of valid inequalities that cuts $\{\mathbf{x^I}, \mathbf{y^I}\}$.

for each subcycle S in $\{\mathbf{x^I}, \mathbf{y^I}\}$ **do**
  $T \leftarrow \emptyset$
  if $D(S) \neq V$ **then**
    if $S \in \gamma(V)$ **then**
      $T \leftarrow T \cup$ inequality (7) associated with S
    else
      $T \leftarrow T \cup$ inequality (8) associated with S
      for each $v \in S$ **do**
        $S_{aug} \leftarrow S \cup C(v)$
        if $S_{aug} \cap (V^I \setminus S) = \emptyset$ **then**
          if $D(S_{aug}) \neq V$ **then**
            $T \leftarrow T \cup$ inequality (7) associated with S_{aug}
          else
            for $u \in V$ **do**
              $S_u \leftarrow S \cap C(u)$;
              if $\delta(S_u) \cap E^I = \emptyset$ **then**
                $T \leftarrow T \cup$ CI inequality (10) associated with S_{aug} and S_u
  else
    for each subcycle S' in $\{\mathbf{x^I}, \mathbf{y^I}\} : S' \neq S$ **do**
      $T \leftarrow T \cup$ inequality (9) associated with S and S'
return T;

As observed by Fischetti, González and Toth [12], the separation problem to find one or more inequalities (9) violated by $\{\mathbf{x^F}, \mathbf{y^F}\}$ can be reduced to the problem of computing a minimum cut between two vertices i and j in graph G^F, $i \in S$ and $j \in V^F \backslash S$, i.e., finding the maximum flow from i to j [1]. Similarly, the separation of inequalities (8) can be reduced to computing a minimum cut in graph G^F that separates $i \in S$ and $C(u) \subseteq V^F \backslash S$. In other words, finding the maximum flow from i to t [1], where t is an artificial vertex connected to each $j \in C(u)$ through edges with infinite capacity. As for inequalities (7), the separation problem can be reduced to computing a minimum cut between covering sets $C(v)$ and $C(u)$ in graph G^F, with $C(v) \subseteq S$, $C(u) \subseteq V \backslash S$, and $C(v) \cap C(u) = \emptyset$. A maximum flow from s to t can be computed, such that s and t are artificial vertices connected, respectively, to each vertex in $C(v)$ and $C(u)$ with infinite capacity edges, as illustrated in Figure 2.5. It is worth noting that the separation of inequality (7) does not work when $C(v)$ and $C(u)$ overlap, since every cut separating s and t has infinite weight, as exemplified in Figure 2.6a.

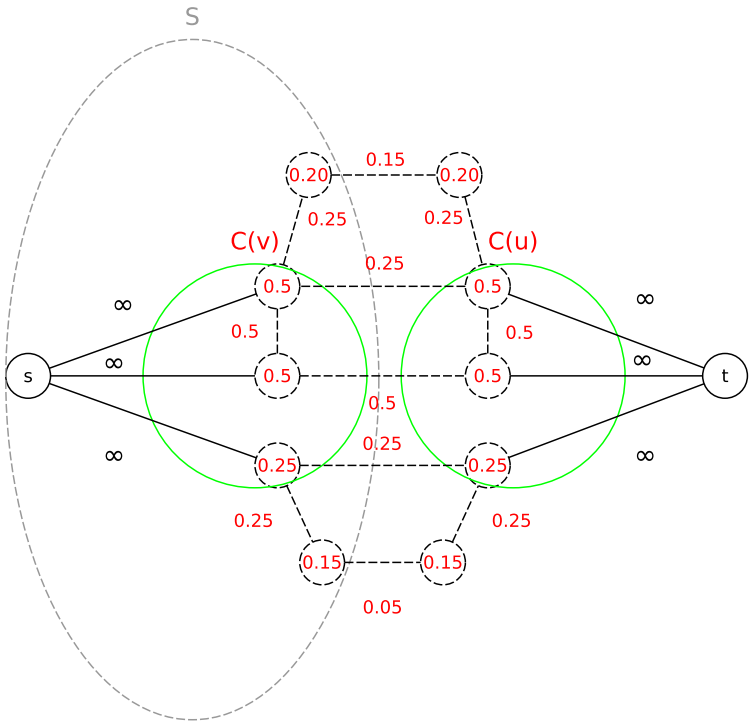
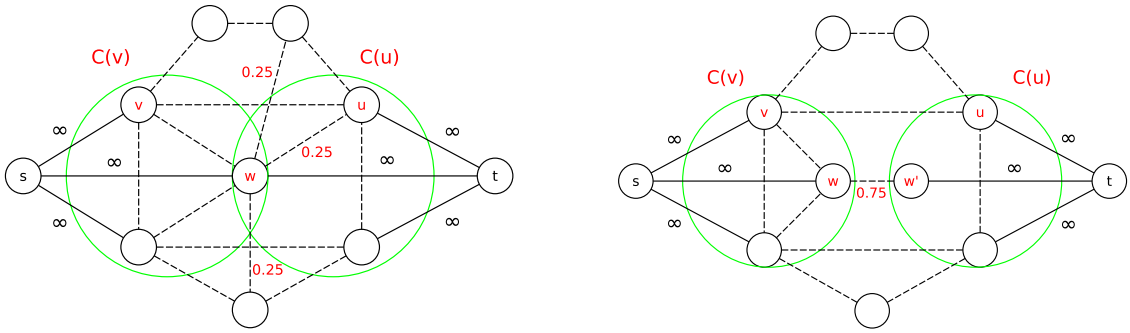


Figure 2.5: Max-flow instance for the separation of inequality (7) in the case where $C(v) \cap C(u) = \emptyset$.

An exact separation algorithm for CI inequalities (10) is proposed to accommodate the case when two covering sets $C(v)$ and $C(u)$ overlap. The first step is to augment graph G^F, by including an artificial vertex w' and an artificial edge (w, w') for every vertex $w \in C(v) \cap C(u)$. Vertex w is removed from $C(u)$ and vertex w' is included into $C(u)$. Finally, for each $w \in C(v) \cap C(u)$, let T_w be the set of edges with one endpoint being w and the other is in $V \backslash C(v)$. The edges of T_w are excluded from G^F and their total weight is transferred to the artificial edge (w, w'). This ensures that every artificial edge will be counted for in any minimum cut, in the sense that every edge in T_w contributes in

their purpose of connecting both covering sets $C(v)$ and $C(u)$, as expected in a feasible solution. Figure 2.6 illustrates the augmentation of graph G^F.



(a) Pre-augmentation

(b) Augmented graph

Figure 2.6: Graph augmentation for the separation of CI inequalities (10).

The separation of a CI inequality (10) reduces to computing a minimum cut between sets $C(v)$ and $C(u)$ in the augmented graph. Let $\delta(S_{min})$ be the minimum cut between $C(v)$ and $C(u)$ and $S_u = S_{min} \cap C(u)$. If $\sum_{e \in \delta(S_{min}) \bigcup \delta(S_u)} x_e$ has a value less than 2, then a violated CI inequality (10) was found.

Algorithm 3 presents the implementation of exact separation routine for fractional solutions. The separation consists in computing a max-flow for each pair of vertices, thus considering a push-relabel algorithm [1] to solve max-flow, the time complexity of Algorithm 3 is bounded by $O(V^4E)$.

Algorithm 3 Exact separation routine for fractional solutions.

Input: graph $G^F(V^F, E^F)$ induced by a fractional solution $\{\mathbf{x^F}, \mathbf{y^F}\}$ for the CSP.
Output: a set T of valid inequalities that cuts $\{\mathbf{x^F}, \mathbf{y^F}\}$.

1: $T \leftarrow \emptyset$
2: **for** $v \in V$ **do**
3: **for** $u \in V \setminus \{v\}$ **do**
4: **if** $D(C(v)) \neq V$ **and** $C(v) \cap C(u) = \emptyset$ **then**
5: $S \leftarrow \text{minCut}(C(v), C(u), G^F)$
6: $T \leftarrow T \cup$ inequality (7) associated with S.
7: **else**
8: $G^F_{aug} \leftarrow \text{augment}(G^F)$
9: $S \leftarrow \text{minCut}(C(v), C(u), G^F_{aug})$
10: $T \leftarrow T \cup$ CI inequality (10) associated with S and u.
11: **if** $y_v > 0$ **and** $v \notin C(u)$ **then**
12: $S \leftarrow \text{minCut}(v, C(u), G^F)$
13: $T \leftarrow T \cup$ inequality (8) associated with S and u.
14: **if** $y_v + y_u - 1 > 0$ **then**
15: $S \leftarrow \text{minCut}(v, u, G^F)$
16: $T \leftarrow T \cup$ inequality (9) associated with S, v and u.
17: **return** T;

Given the computational effort required for the exact separation of fractional solutions, two alternatives were investigated. The first is based on a *first-found* policy, which follows the same steps of Algorithm 3, however the execution is interrupted once the first inequality which surpasses a given violation threshold ϵ is found. For example, with respect to inequalities (7), given a vertex $v \in V$ and a set $S \in \gamma(V) : D(S) \neq V$, if the following holds, $(2 - \sum_{e \in \delta(S)} x_e > \epsilon)$, then the cut is included in the model and Algorithm 3 halts. The same goes for inequalities (8-10).

The second alternative for the exact separation routines resides in the heuristic separation of inequalities (7-10), described in the following section.

2.4.3 Heuristic separation routine for fractional solutions

A heuristic separation has the purpose of finding inequalities being violated by a fractional solution $\{\mathbf{x^F}, \mathbf{y^F}\}$ within short computational times. In contrast with the exact separation routine however, a heuristic does not come with any guarantee of finding a violated inequality even if one exists.

The heuristic separation routine for inequalities (7-10) is composed of four main steps. The first step searches for inequalities (7) and (10) for every $u \in V$ and its corresponding covering set $C(u)$. In more details, let $S = C(u)$ and consider two cases: (i) if $D(S) \neq V$ and $\sum_{e \in \delta(S)} x_e < 2$, then the inequality (7) associated with S cuts $\{\mathbf{x^F}, \mathbf{y^F}\}$; (ii) if $D(S) = V$ and $\sum_{e \in \delta(S)} x_e + \sum_{e \in \delta(S_v) \setminus \delta(S)} x_e < 2$ for some vertex $v \in V \setminus \{u\}$, then the CI inequality (10) associated with S and v cuts $\{\mathbf{x^F}, \mathbf{y^F}\}$.

In the second step, the connected components $S_1, \ldots, S_p$ of G^F are computed. For each component S_k, let $S = S_k$ and if $S \in \gamma(V)$ then two cases are considered: (i) if $D(S) \neq V$, then the inequality (7) associated with S cuts $\{\mathbf{x^F}, \mathbf{y^F}\}$; (ii) if $D(S) = V$ and $\sum_{e \in \delta(S_v)} x_e < 2$ for some vertex $v \in V$, then the CI inequality (10) associated with S and v cuts $\{\mathbf{x^F}, \mathbf{y^F}\}$. .

In the third step, for each connected component S_k, let $S = S_k$ and $i = \arg\max_v \{y_v : v \in S\}$. If $D(S) \neq V$, then the inequality (8) associated with S and v cuts $\{\mathbf{x^F}, \mathbf{y^F}\}$.

Finally, the fourth step iterates through all pairs of connected components S_k and S_l, $k \neq l$. For each pair, let $i = \arg\max_v \{y_v : v \in S_k\}$ and $j = \arg\max_v \{y_v : v \in S_l\}$. If $y_i + y_j > 1$, the inequality (9) associated with $S = S_k$, i, and j cuts $\{\mathbf{x^F}, \mathbf{y^F}\}$.

Algorithm 4, with a time complexity bounded by $O(V^3)$, details the heuristic separation routine for fractional solutions.

2.5 Computational Experiments

In this section, the proposed branch-and-cut methodologies are evaluated and compared to the state-of-the-art using the literature benchmark of instances, described in the following section.

Algorithm 4 Heuristic separation routine for fractional solutions.

Input: graph $G^F(V^F, E^F)$ induced by a fractional solution $\{\mathbf{x^F}, \mathbf{y^F}\}$ for the CSP.
Output: a set T of valid inequalities that cuts $\{\mathbf{x^F}, \mathbf{y^F}\}$.

$T \leftarrow \emptyset$
for $u \in V$ **do**
    $S \leftarrow C(u)$;
    if $D(S) \neq V$ **then**
        if $\sum_{e \in \delta(S)} x_e < 2$ **then**
            $T \leftarrow T \cup$ inequality (7) associated with S.
    else
        for $v \in V : v \neq u$ **do**
            $S_v \leftarrow S \cap C(v)$;
            if $\sum_{e \in \delta(S)} x_e + \sum_{e \in \delta(S_v) \setminus \delta(S)} x_e < 2$ **then**
                $T \leftarrow T \cup$ CI inequality (10) associated with S and v.
Compute the connected components $S_1, \ldots, S_p$ of G^F;
for $k = 1, ..., p$ **do**
    $S \leftarrow S_k$
    if $S \in \gamma(V)$ **then**
        if $D(S) \neq V$ **then**
            $T \leftarrow T \cup$ inequality (7) associated with S.
        else
            for $v \in V : v \neq w$ **do**
                $S_v \leftarrow S \cap C(v)$;
                if $\sum_{e \in \delta(S_v) \setminus \delta(S)} x_e < 2$ **then**
                    $T \leftarrow T \cup$ CI inequality (10) associated with S and v.
    else
        $i \leftarrow \arg\max_v \{y_v : v \in S\}$.
        if $D(S) \neq V$ **then**
            $T \leftarrow T \cup$ inequality (8) associated with S and i.
        for $l = k, ..., p$ **do**
            $j \leftarrow \arg\max_v \{y_v : v \in S_l\}$
            $T \leftarrow T \cup$ inequality (9) associated with S, i and j.
return T

2.5.1 Instances

The benchmark used in the computational experiments is composed of instances by Salari et al.[20] and a new set of instances, all of them based on the TSPLIB [18]. Salari et al.[20] divided the instances into three types: small (36 instances, $51 \leqslant |V| \leqslant 100$), medium (12 instances, $150 \leqslant |V| \leqslant 200$), and large (27 instances, $532 \leqslant |V| \leqslant 783$). The covering set of each vertex is defined by its k closest vertices. For each graph of small and medium instances, three values of k were used, $k = 7$, $k = 9$, and $k = 11$. In order to complement the medium instances, we also created 39 new instances with $225 \leqslant |V| \leqslant 493$ to complement the set from the literature. The large instances were tested in the literature for heuristics only, and for this reason, our results for these instances were not included in the paper. Nonetheless, full experimental data (including results for large instances) and source codes are available on-line[1]

2.5.2 Computational Settings

The branch-and-cut methodologies were implemented in C++ using solver Gurobi 8.1.1 and the Lemon graph library [9]. The experiments were conducted on a PC under Ubuntu and CPU Intel Xeon E5-2630 2.2 GHz, with 64GB of RAM and one-hour time limit.

2.5.3 Evaluated Methodologies

Five branch-and-cut methodologies were implemented and evaluated in the computational experiments:

1. $CSP\text{-}I$: exact separation routine for integer solutions (Algorithm 2) considering valid inequalities (7), (8), and (9), but excluding the CI inequalities (10);

2. $CSP\text{-}I\&F_{vp}$: uses the same exact separation routine for integer solutions as $CSP\text{-}I$. On the root node, exact separation routine for fractional solutions (Algorithm 3) considering inequalities (7), (8), and (9), but excluding the CI inequalities for CSP (10). For the non-root nodes, Algorithm 3 was implemented under the *first-found* policy with violation threshold $\epsilon = 1$ (see Section 2.4.2).

3. $CSP\text{-}I\&F_{vp}\text{-}X$: same as $CSP\text{-}I\&F_{vp}$, but including the CI inequalities (10);

4. $CSP\text{-}I\&F_h$: uses the same exact separation routine for integer solutions as $CSP\text{-}I$. On the root node, exact separation routine for fractional solutions (Algorithm 3) considering inequalities (7), (8), and (9), but excluding the CI inequalities (10). For the non-root nodes, heuristic separation for fractional solutions (Algorithm 4, considering inequalities (7), (8), and (9), but excluding the CI inequalities (10);

5. $CSP\text{-}I\&F_h\text{-}X$: Same as $CSP\text{-}I\&F_h$, but including inequalities (10);

[1] `http://www.ic.unicamp.br/~fusberti/problems/csp`.

These methodologies were compared with the integer linear programming formulation proposed by Salari et al.[20], denoted here as SRS. To the best of our knowledge, SRS is the best performing exact methodology for the CSP.

Preliminary experiments have shown that even in cases where the heuristic separation fails to find violated inequalities in methodologies $CSP\text{-}I\&F_h$ and $CSP\text{-}I\&F_h\text{-}X$, applying the exact separation does not improve the quality of the solutions obtained. This can be justified by the high computational effort spent by the exact separation routines.

2.5.4 Results

The results of the computational experiments are reported for the small instances in Table 2.2 and for the medium instances in Tables 2.3 and 2.4. Each table reports for each methodology and for each instance, the following:

- LB: best lower bound obtained;
- *Gap*: optimality gap $(\frac{UB - LB}{LB}) \cdot 100$;
- *Time*: execution time in seconds.

In Tables 2.2 and 2.3, the column group *BestUB* reports the best upper bounds known in the literature for each instance: column UB gives the best known upper bounds and column *References* cites the papers which attained them. For each instance, Tables 2.2, 2.3, and 2.4 highlight the optimal solutions (underlined) and the best lower bounds (in bold) obtained by each methodology.

Table 2.2: Results of computational experiments for the small-size instances.

		BestUB		*SRS*			*CSP-I*			*CSP-I&F_{vp}*			*CSP-I&F_h*			*CSP-I&F_{vp}-X*			*CSP-I&F_h-X*		
Instance	NC	UB	References	LB	Gap	Time	LB	Gap	Time	LB	Gap	Time	LB	Gap	Time	LB	Gap	Time	LB	Gap	Time
eil51	7	164	[14, 19, 20, 24, 25, 17, 16]	**164**	**0**	149	**164**	**0**	2	**164**	**0**	4	**164**	**0**	3	**164**	**0**	3	**164**	**0**	3
	9	159	[14, 19, 20, 24, 25, 17, 16]	**159**	**0**	220	**159**	**0**	1	**159**	**0**	2	**159**	**0**	2	**159**	**0**	4	**159**	**0**	3
	11	147	[14, 19, 20, 24, 25, 17, 16]	**147**	**0**	681	**147**	**0**	1	**147**	**0**	2	**147**	**0**	2	**147**	**0**	5	**147**	**0**	4
berlin52	7	3887	[14, 19, 20, 24, 25, 17, 16]	**3887**	**0**	140	**3887**	**0**	2	**3887**	**0**	2	**3887**	**0**	2	**3887**	**0**	4	**3887**	**0**	3
	9	3430	[14, 19, 20, 24, 25, 17, 16]	**3430**	**0**	212	**3430**	**0**	1	**3430**	**0**	3	**3430**	**0**	2	**3430**	**0**	6	**3430**	**0**	3
	11	3262	[14, 19, 20, 24, 25, 17, 16]	**3262**	**0**	255	**3262**	**0**	1	**3262**	**0**	2	**3262**	**0**	2	**3262**	**0**	4	**3262**	**0**	4
st70	7	288	[14, 19, 20, 24, 25, 17, 16]	**288**	**0**	490	**288**	**0**	3	**288**	**0**	7	**288**	**0**	5	**288**	**0**	7	**288**	**0**	7
	9	259	[14, 19, 20, 24, 25, 17, 16]	**259**	**0**	1391	**259**	**0**	3	**259**	**0**	7	**259**	**0**	6	**259**	**0**	13	**259**	**0**	9
	11	247	[14, 19, 20, 24, 25, 17, 16]	218	13.14	3600	**247**	**0**	3	**247**	**0**	7	**247**	**0**	5	**247**	**0**	14	**247**	**0**	9
eil76	7	207	[14, 19, 20, 24, 17, 16]	193	7.45	3600	**207**	**0**	6	**207**	**0**	29	**207**	**0**	9	**207**	**0**	15	**207**	**0**	12
	9	185	[19, 25, 17, 16]	161	14.65	3600	**185**	**0**	10	**185**	**0**	10	**185**	**0**	9	**185**	**0**	13	**185**	**0**	12
	11	170	[14, 19, 20, 24, 25, 17, 16]	145	17.08	3600	**170**	**0**	6	**170**	**0**	8	**170**	**0**	7	**170**	**0**	15	**170**	**0**	13
pr76	7	50275	[14, 19, 20, 24, 25, 17, 16]	**50275**	**0**	2488	**50275**	**0**	7	**50275**	**0**	8	**50275**	**0**	6	**50275**	**0**	12	**50275**	**0**	8
	9	45348	[14, 19, 20, 24, 25, 17, 16]	42935	5.62	3600	**45348**	**0**	6	**45348**	**0**	32	**45348**	**0**	10	**45348**	**0**	16	**45348**	**0**	14
	11	43028	[14, 19, 20, 24, 25, 17, 16]	39022	10.27	3600	**43028**	**0**	28	**43028**	**0**	17	**43028**	**0**	46	**43028**	**0**	27	**43028**	**0**	28
rat99	7	486	[14, 19, 20, 24, 25, 17, 16]	433	12.21	3600	**486**	**0**	238	**486**	**0**	19	**486**	**0**	17	**486**	**0**	33	**486**	**0**	17
	9	455	[14, 19, 20, 24, 25, 17, 16]	377	20.73	3600	438	3.88	3600	**455**	**0**	30	**455**	**0**	27	**455**	**0**	28	**455**	**0**	29
	11	444	[14, 19, 20, 24, 25, 17, 16]	350	26.81	3600	**444**	**0**	203	**444**	**0**	32	**444**	**0**	138	**444**	**0**	37	**444**	**0**	197

Continued on next page

Continued from previous page

		BestUB		*SRS*			*CSP-I*			*CSP-I&F_{vp}*			*CSP-I&F_h*			*CSP-I&F_{vp}-X*			*CSP-I&F_h-X*		
Instance	NC	UB	References	LB	Gap	Time	LB	Gap	Time	LB	Gap	Time	LB	Gap	Time	LB	Gap	Time	LB	Gap	Time
kroA100	7	9674	[14, 19, 20, 24, 25, 17, 16]	9177	5.42	3600	**9674**	**0**	15	**9674**	**0**	18	**9674**	**0**	27	**9674**	**0**	27	**9674**	**0**	30
	9	9159	[14, 19, 20, 24, 25, 17, 16]	7938	15.38	3600	**9159**	**0**	154	**9159**	**0**	28	**9159**	**0**	2040	**9159**	**0**	36	**9159**	**0**	2230
	11	8901	[14, 19, 20, 24, 25, 17, 16]	8593	3.59	3600	8608	3.40	3600	**8901**	**0**	45	8640	3.02	3600	**8901**	**0**	79	8801	1.14	3600
kroB100	7	9537	[14, 19, 20, 24, 25, 17, 16]	-	-	3600	**9537**	**0**	45	**9537**	**0**	22	**9537**	**0**	20	**9537**	**0**	26	**9537**	**0**	23
	9	9240	[14, 19, 20, 24, 25, 17, 16]	7678	20.34	3600	**9240**	**0**	363	**9240**	**0**	21	**9240**	**0**	21	**9240**	**0**	31	**9240**	**0**	28
	11	8842	[14, 19, 20, 24, 25, 17, 16]	-	-	3600	**8842**	**0**	141	**8842**	**0**	25	**8842**	**0**	29	**8842**	**0**	40	**8842**	**0**	36
kroC100	7	9723	[14, 19, 20, 24, 25, 17, 16]	8564	13.54	3600	**9723**	**0**	561	**9723**	**0**	107	**9723**	**0**	102	**9723**	**0**	67	**9723**	**0**	92
	9	9171	[14, 19, 20, 24, 25, 17, 16]	7663	19.68	3600	8920	2.81	3600	**9171**	**0**	45	**9171**	**0**	783	**9171**	**0**	123	**9171**	**0**	972
	11	8632	[14, 19, 20, 24, 25, 17, 16]	7590	13.73	3600	**8632**	**0**	254	**8632**	**0**	38	**8632**	**0**	820	**8632**	**0**	222	**8632**	**0**	870
kroD100	7	9626	[14, 19, 20, 24, 25, 17, 16]	8724	10.34	3600	**9626**	**0**	59	**9626**	**0**	17	**9626**	**0**	20	**9626**	**0**	34	**9626**	**0**	23
	9	8885	[14, 19, 20, 24, 25, 17, 16]	-	-	3600	**8885**	**0**	16	**8885**	**0**	22	**8885**	**0**	27	**8885**	**0**	62	**8885**	**0**	35
	11	8725	[14, 19, 20, 24, 25, 17, 16]	-	-	3600	**8725**	**0**	51	**8725**	**0**	35	**8725**	**0**	48	**8725**	**0**	63	**8725**	**0**	80
kroE100	7	10150	[14, 19, 20, 24, 25, 17, 16]	9274	9.44	3600	**10150**	**0**	520	**10150**	**0**	81	**10150**	**0**	42	**10150**	**0**	76	**10150**	**0**	32
	9	8991	[14, 19, 24, 25, 17, 16]	8500	5.77	3600	**8991**	**0**	336	**8991**	**0**	31	**8991**	**0**	55	**8991**	**0**	28	**8991**	**0**	88
	11	8450	[14, 19, 20, 24, 25, 17, 16]	7739	9.19	3600	**8450**	**0**	237	**8450**	**0**	23	**8450**	**0**	261	**8450**	**0**	33	**8450**	**0**	193
rd100	7	3461	[14, 19, 20, 24, 25, 17, 16]	3094	11.88	3600	**3461**	**0**	119	**3461**	**0**	20	**3461**	**0**	20	**3461**	**0**	24	**3461**	**0**	22
	9	3194	[14, 19, 20, 24, 25, 17, 16]	2664	19.90	3600	**3194**	**0**	63	**3194**	**0**	18	**3194**	**0**	18	**3194**	**0**	24	**3194**	**0**	25
	11	2922	[14, 19, 20, 24, 25, 17, 16]	2648	10.33	3600	**2922**	**0**	28	**2922**	**0**	21	**2922**	**0**	20	**2922**	**0**	34	**2922**	**0**	27
Avg				**7673.50**	**9.27**	**2867.39**	**8310.08**	**0.28**	**396.75**	**8325.67**	**0.00**	**23.28**	**8318.42**	**0.08**	**229.19**	**8325.67**	**0.00**	**35.69**	**8322.89**	**0.03**	**243.92**

Table 2.3: Results of computational experiments for the medium-size instances.

		BestUB		*CSP-I*			*CSP-I&F_{vp}*			*CSP-I&F_h*			*CSP-I&F_{vp}-X*			*CSP-I&F_h-X*		
Instance	NC	UB	Reference(s)	LB	Gap	Time	LB	Gap	Time	LB	Gap	Time	LB	Gap	Time	LB	Gap	Time
kroA150	7	11423	[14, 19, 20, 24, 25, 17, 16]	10658	7.18	3600	**11423**	**0**	174	**11423**	**0**	137	**11423**	**0**	147	**11423**	**0**	90
	9	10056	[14, 19, 20, 24, 25, 17, 16]	**10056**	**0**	147	**10056**	**0**	84	**10056**	**0**	85	**10056**	**0**	92	**10056**	**0**	122
	11	9439	[14, 19, 20, 24, 25, 17, 16]	9240	2.15	3600	**9439**	**0**	95	**9439**	**0**	67	**9439**	**0**	243	**9439**	**0**	91
kroB150	7	11457	[14, 19, 20, 24, 25, 17, 16]	10663	7.45	3600	**11457**	**0**	334	**11457**	**0**	116	**11457**	**0**	113	**11457**	**0**	81
	9	10121	[14, 19, 20, 24, 25, 17, 16]	9951	1.71	3600	**10121**	**0**	280	**10121**	**0**	130	**10121**	**0**	145	**10121**	**0**	112
	11	9611	[14, 19, 20, 24, 25, 17, 16]	**9611**	**0**	902	**9611**	**0**	849	**9611**	**0**	429	**9611**	**0**	947	**9611**	**0**	282
kroA200	7	13285	[14, 19, 25, 16]	11660	13.94	3600	12611	5.34	3600	12955	2.55	3600	12697	4.63	3600	**13108**	1.35	3600
	9	11708	[14, 19, 20, 24, 25, 17, 16]	10327	13.37	3600	11094	5.53	3600	**11708**	**0**	2252	11537	1.48	3600	**11708**	**0**	1008
	11	10748	[14, 19, 25, 17, 16]	9508	13.04	3600	10342	3.93	3600	**10748**	**0**	1044	**10748**	**0**	3582	**10748**	**0**	648
kroB200	7	13051	[14, 19, 20, 24, 25, 17, 16]	12260	6.45	3600	12462	4.73	3600	12904	1.14	3600	12697	2.79	3600	**13051**	**0**	1487
	9	11864	[19, 20, 24, 25, 17, 16]	11209	5.84	3600	11379	4.26	3600	**11864**	**0**	2281	11695	1.45	3600	**11864**	**0**	1242
	11	10644	[19, 20, 24, 25, 17, 16]	10405	2.30	3600	**10644**	**0**	800	**10644**	**0**	907	**10644**	**0**	514	**10644**	**0**	938
Avg				**10462.33**	**6.12**	**3087.42**	**10886.58**	**1.98**	**1718.00**	**11077.50**	**0.31**	**1220.67**	**11010.50**	**0.86**	**1681.92**	**11102.42**	**0.11**	**808.42**

Table 2.4: Results of computational experiments for the new medium-size instances created.

		CSP-I			*CSP-I&F_{vp}*			*CSP-I&F_h*			*CSP-I&F_{vp}-X*			*CSP-I&F_h-X*		
Instance	NC	LB	Gap	Time	LB	Gap	Time	LB	Gap	Time	LB	Gap	Time	LB	Gap	Time
	7	68805	3.74	3600	70254	1.79	3600	**70988**	0.50	3600	70802	0.83	3600	70976	0.25	3600
ts225	9	59718	10.36	3600	61834	16.17	3600	64048	1.72	3600	62899	4.77	3600	**64404**	1.22	3600
	11	50022	15.54	3600	51057	15.51	3600	53691	8.45	3600	52755	37.07	3600	**53786**	7.40	3600
	7	1565	8.05	3600	1622	3.70	3600	1649	1.88	3600	1644	2.19	3600	**1669**	0.66	3600
tsp225	9	1310	15.34	3600	1476	2.30	3600	1492	1.21	3600	1500	0.67	3600	**1510**	**0.00**	1218
	11	1223	12.26	3600	1310	6.49	3600	1346	1.56	3600	1340	2.01	3600	**1367**	**0.00**	1271
	7	**50479**	13.89	3600	47279	-	3600	50216	85.87	3600	48646	-	3600	50420	89.03	3600
pr226	9	**50916**	8.90	3600	47230	-	3600	49911	-	3600	51110	43.43	3600	50191	85.38	3600
	11	**50236**	6.51	3600	44162	-	3600	48925	-	3600	45569	-	3600	49444	127.37	3600
	7	938	12.15	3600	931	-	3600	**991**	6.76	3600	943	-	3600	965	-	3600
gil262	9	804	20.27	3600	865	-	3600	888	-	3600	877	-	3600	**901**	5.33	3600
	11	798	10.03	3600	850	3.06	3600	850	2.71	3600	**856**	1.99	3600	854	2.69	3600
	7	19057	191.89	3600	20360	-	3600	20970	-	3600	20425	-	3600	**20983**	-	3600
pr264	9	17962	-	3600	18986	-	3600	20257	-	3600	19115	-	3600	**20497**	375.64	3600
	11	17052	-	3600	18790	-	3600	**19748**	409.25	3600	19183	-	3600	19556	-	3600
	7	772	53.63	3600	1046	-	3600	1083	21.79	3600	**1088**	122.15	3600	1086	-	3600
a280	9	876	25.23	3600	946	15.22	3600	**1003**	2.89	3600	988	5.87	3600	983	-	3600
	11	817	20.93	3600	871	12.40	3600	911	5.16	3600	917	-	3600	**939**	2.56	3600
	7	17263	42.72	3600	20545	-	3600	21488	16.31	3600	21291	16.21	3600	**21517**	-	3600
pr299	9	18121	20.46	3600	19609	-	3600	20092	5.14	3600	19972	143.62	3600	**20305**	3.97	3600
	11	13806	45.84	3600	18256	-	3600	**18485**	5.17	3600	18275	23.76	3600	18452	5.84	3600

Continued on next page

Continued from previous page

		CSP-I			*CSP-I&F_{vp}*			*CSP-I&F_h*			*CSP-I&F_{vp}-X*			*CSP-I&F_h-X*		
Instance	NC	LB	Gap	Time	LB	Gap	Time	LB	Gap	Time	LB	Gap	Time	LB	Gap	Time
	7	17634	23.90	3600	18715	12.95	3600	**19331**	-	3600	18991	209.48	3600	18886	-	3600
lin318	9	14073	42.85	3600	16012	-	3600	16718	-	3600	16229	223.08	3600	**16920**	15.76	3600
	11	12389	48.97	3600	15511	-	3600	15425	247.70	3600	**15809**	-	3600	15526	236.91	3600
	7	4272	73.36	3600	5935	-	3600	**6093**	-	3600	6054	-	3600	6041	-	3600
rd400	9	3705	82.73	3600	5354	-	3600	5342	237.06	3600	**5389**	-	3600	5388	-	3600
	11	2948	93.69	3600	4755	-	3600	4749	22.53	3600	4819	-	3600	**4825**	21.24	3600
	7	4670	141.31	3600	5465	-	3600	5452	1125.84	3600	**5553**	-	3600	5540	-	3600
fl417	9	3900	246.90	3600	5011	-	3600	**5447**	-	3600	5097	-	3600	5281	-	3600
	11	4119	-	3600	**4894**	-	3600	4882	-	3600	4891	-	3600	4881	-	3600
	7	38920	81.77	3600	49869	-	3600	49794	-	3600	**50865**	-	3600	50704	-	3600
pr439	9	35796	121.95	3600	42732	-	3600	42734	-	3600	**46196**	-	3600	46046	-	3600
	11	30738	85.65	3600	43008	-	3600	42962	369.00	3600	**43389**	-	3600	43321	-	3600
	7	13992	89.07	3600	19760	-	3600	19768	20.62	3600	**20618**	-	3600	20560	-	3600
pcb442	9	12390	142.28	3600	17009	-	3600	17105	-	3600	**18434**	-	3600	18403	35.18	3600
	11	10760	103.79	3600	16436	-	3600	16144	-	3600	**17478**	13.13	3600	17448	16.94	3600
	7	12961	91.00	3600	16160	-	3600	16044	-	3600	**16628**	-	3600	16373	-	3600
d493	9	11939	78.78	3600	15010	-	3600	14796	-	3600	15078	-	3600	**15230**	-	3600
	11	10935	92.91	3600	13937	-	3600	13894	-	3600	**14189**	18.40	3600	14146	-	3600
Avg		**17658**		**3600**	**19586**		**3600**	**20146**		**3600**	**20151**		**3600**	**20419**		**3479**

For small-size instances, there were previously known lower bounds for 32 out of 36 instances, obtained by SRS, from which optimal solutions were proven for 9 instances. The proposed branch-and-cut framework, on the other hand, obtained lower bounds for all instances. More importantly, the framework proved optimality for all 36 small instances. All branch-and-cut methodologies outperformed SRS with respect to optimality gap, and they were fairly robust among themselves; the worst performing ($CSP\text{-}I$) obtained an average 0.28% optimality gap, while the best performing ($CSP\text{-}I\&F_{vp}$ and $CSP\text{-}I\&F_{vp}\text{-}X$) with zero optimality gap, shows the exact separation prevails over the heuristic separation of fractional solutions for small instances.

With respect to medium-size instances created by Salari et al., no lower bound was known for any of the 12 instances in the literature. The branch-and-cut framework obtained the first lower bounds for all these instances. Furthermore, optimality was proven for all instances except one ($kroA200\text{-}7$), which remains with an optimality gap of 1.35%. The performance among the branch-and-cut methodologies varied more significantly this time. The best-performing methodology was $CSP\text{-}I\&F_h\text{-}X$, with an average gap of 0.11%. The heuristic separation overcomes the exact separation, mainly due to the reduction in computational effort. The worst-performing methodology ($CSP\text{-}I$) obtained an average gap of 6.12%, showing that integral cuts alone perform poorly for more challenging instances.

Regarding the medium-size instances generated in this work, the proposed branch-and-cut framework obtained lower bounds for all instances. Among the 39 instances, the framework proved optimality for 2 instances. The best methodology was $CSP\text{-}I\&F_h\text{-}X$, with an average lower bound of 20419, followed by methodologies $CSP\text{-}I\&F_{vp}\text{-}X$, $CSP\text{-}I\&F_h$ and $CSP\text{-}I\&F_{vp}$ with average lower bounds of 20151, 20146, and 19586, respectively. Moreover, $CSP\text{-}I\&F_h\text{-}X$ obtained the best lower bound among all methodologies for 14 instances, followed by $CSP\text{-}I\&F_{vp}\text{-}X$ with 13 instances, $CSP\text{-}I\&F_h$ with 8 instances, and $CSP\text{-}I\&F_{vp}$ with 1 instance.

The effect of the CI inequalities (10) in the performance of the methodologies was also examined. Regarding the small and medium instances created by Salari et al., the average gaps of $CSP\text{-}I\&F_{vp}$ and $CSP\text{-}I\&F_{vp}\text{-}X$ were both zero for small instances, but for the medium instances the CI inequalities reduced the average gap from 1.98% to 0.86%. Furthermore, comparing $CSP\text{-}I\&F_h$ and $CSP\text{-}I\&F_h\text{-}X$, the CI inequalities reduced the average gaps from 0.08% to 0.03% for small instances and from 0.31% to 0.11% for medium instances. It is worth mentioning that even for the medium instances where the methodologies $CSP\text{-}I\&F_{vp}$ and $CSP\text{-}I\&F_h$ obtained the same optimality gaps of $CSP\text{-}I\&F_{vp}\text{-}X$ and $CSP\text{-}I\&F_h\text{-}X$, the CI inequalities reduced the execution time on average. For example, comparing the methodologies that use heuristic separation, the difference in execution time was substantial. The average execution times of $CSP\text{-}I\&F_h$ and $CSP\text{-}I\&F_h\text{-}X$ were 1220.67 and 808.42, respectively, representing a reduction of more than 33%.

Considering the new medium instances, from a total of 39 instances, $CSP\text{-}I\&F_{vp}\text{-}X$ and $CSP\text{-}I\&F_h\text{-}X$ obtained, together, the best lower bounds for 27 instances, while $CSP\text{-}I\&F_{vp}$ and $CSP\text{-}I\&F_h$ obtained the best lower bounds for only 9 instances. Furthermore, only $CSP\text{-}I\&F_h\text{-}X$ was able to obtain optimal solutions. Therefore, the results show that the CI inequalities have a significant impact on reducing the optimality gaps and obtaining the best lower bounds.

2.6 Final Remarks

The proposed branch-and-cut framework for the CSP uses existing valid inequalities for the GTSP, by Fischetti et al. [12], and a new family of valid inequalities, *CI inequalities*, to improve on the state-of-the-art exact methodology for the CSP. Exact and heuristic separation routines for integer and fractional solutions are investigated.

The branch-and-cut framework is composed of five methodologies using distinct families of inequalities and separation routines. Computational experiments conducted on a benchmark of 48 instances from literature and 39 new instances delves into the effectiveness of the framework. From the 48 small and medium instances from literature, only 9 optimal solutions were known. Our branch-and-cut framework, by borrowing meaningful valid inequalities from GTSP and proposing new valid inequalities for CSP, was able to obtain optimal solutions for all instances except one, thus 47 instances were proven optimal (among them, 38 instances for the first time). With respect to the new instances, our methodology obtained the highest number of best lower bounds and number of proven optimal solutions. Finally, the experiments also show that the CI inequalities had a significant role in the performance of the methodologies.

The ideas presented in this work can support the exact solution of many future developments of the CSP. The heuristics and metaheuristics approaches proposed to solve the CSP in the literature can be combined with our branch-and-cut framework to develop hybrid methodologies, such as matheuristics, in order to improve the upper and lower bounds of the CSP. Future works may consider, for example, CSP with multiple vehicles, capacity constraints, time constraints, green vehicles, uncertainty on the covering neighborhood, and other generalizations of the CSP which better approximate practical routing problems. The new family of valid inequalities proposed in this work should be considered on the exact solution for any of these generalizations.

2.7 Acknowledgment

This work was supported by CAPES, CNPq, and Fapesp (grants 140960/2017-1, 314384/2018-9, 435520/2018-0, 2015/11937-9).

References

[1] Ravindra K Ahuja, Thomas L Magnanti, and James B Orlin. *Network Flows: Theory, Algorithms, and Applications*. Prentice Hall, 1993.

[2] David L. Applegate, Robert E. Bixby, Vasek Chvatal, and William J. Cook. *The Traveling Salesman Problem: A Computational Study (Princeton Series in Applied Mathematics)*. Princeton University Press, Princeton, NJ, USA, 2007.

[3] Behnam Behdani and J Cole Smith. An integer-programming-based approach to the close-enough traveling salesman problem. *INFORMS Journal on Computing*, 26(3):415–432, 2014.

[4] Walton Pereira Coutinho, Roberto Quirino do Nascimento, Artur Alves Pessoa, and Anand Subramanian. A branch-and-bound algorithm for the close-enough traveling salesman problem. *INFORMS Journal on Computing*, 28(4):752–765, 2016.

[5] John Current, Hasan Pirkul, and Erik Rolland. Efficient algorithms for solving the shortest covering path problem. *Transportation Science*, 28(4):317–327, 1994.

[6] John R Current and David A Schilling. The covering salesman problem. *Transportation science*, 23(3):208–213, 1989.

[7] John R Current and David A Schilling. The median tour and maximal covering tour problems: Formulations and heuristics. *European Journal of Operational Research*, 73(1):114–126, 1994.

[8] Mauro Dell'Amico, Roberto Montemanni, and Stefano Novellani. Algorithms based on branch and bound for the flying sidekick traveling salesman problem. *Omega*, 104:102493, 2021.

[9] Balázs Dezső, Alpár Jüttner, and Péter Kovács. Lemon–an open source c++ graph template library. *Electronic Notes in Theoretical Computer Science*, 264(5):23–45, 2011.

[10] Jing Dong, Ning Yang, and Ming Chen. Heuristic approaches for a tsp variant: The automatic meter reading shortest tour problem. In *Extending the Horizons: Advances in Computing, Optimization, and Decision Technologies*, pages 145–163. Springer, 2007.

[11] Adrian Dumitrescu and Joseph SB Mitchell. Approximation algorithms for tsp with neighborhoods in the plane. *Journal of Algorithms*, 48(1):135–159, 2003.

[12] Matteo Fischetti, Juan José Salazar González, and Paolo Toth. A branch-and-cut algorithm for the symmetric generalized traveling salesman problem. *Operations Research*, 45(3):378–394, 1997.

[13] Michel Gendreau, Gilbert Laporte, and Frédéric Semet. The covering tour problem. *Operations Research*, 45(4):568–576, 1997.

[14] Bruce Golden, Zahra Naji-Azimi, S Raghavan, Majid Salari, and Paolo Toth. The generalized covering salesman problem. *INFORMS Journal on Computing*, 24(4):534–553, 2012.

[15] Damon J Gulczynski, Jeffrey W Heath, and Carter C Price. The close enough traveling salesman problem: A discussion of several heuristics. In *Perspectives in Operations Research*, pages 271–283. Springer, 2006.

[16] Yongliang Lu, Una Benlic, and Qinghua Wu. A highly effective hybrid evolutionary algorithm for the covering salesman problem. *Information Sciences*, 564:144–162, 2021.

[17] Venkatesh Pandiri, Alok Singh, and André Rossi. Two hybrid metaheuristic approaches for the covering salesman problem. *Neural Computing and Applications*, pages 1–21, 2020.

[18] Gerhard Reinelt. Tsplib—a traveling salesman problem library. *ORSA journal on computing*, 3(4):376–384, 1991.

[19] Majid Salari and Zahra Naji-Azimi. An integer programming-based local search for the covering salesman problem. *Computers & Operations Research*, 39(11):2594–2602, 2012.

[20] Majid Salari, Mohammad Reihaneh, and Mohammad S Sabbagh. Combining ant colony optimization algorithm and dynamic programming technique for solving the covering salesman problem. *Computers & Industrial Engineering*, 83:244–251, 2015.

[21] Robert Shuttleworth, Bruce L Golden, Susan Smith, and Edward Wasil. Advances in meter reading: Heuristic solution of the close enough traveling salesman problem over a street network. In *The Vehicle Routing Problem: Latest Advances and New Challenges*, pages 487–501. Springer, 2008.

[22] Gizem Ozbaygin Tiniç, Oya E Karasan, Bahar Y Kara, James F Campbell, and Aysu Ozel. Exact solution approaches for the minimum total cost traveling salesman problem with multiple drones. *Transportation Research Part B: Methodological*, 168:81–123, 2023.

[23] Sebastián A Vásquez, Gustavo Angulo, and Mathias A Klapp. An exact solution method for the tsp with drone based on decomposition. *Computers & Operations Research*, 127:105127, 2021.

[24] Pandiri Venkatesh, Gaurav Srivastava, and Alok Singh. A multi-start iterated local search algorithm with variable degree of perturbation for the covering salesman problem. In *Harmony Search and Nature Inspired Optimization Algorithms*, pages 279–292. Springer, 2019.

[25] Xiaoning Zang, Li Jiang, Mustapha Ratli, and Bin Ding. A parallel variable neighborhood search for solving covering salesman problem. *Optimization Letters*, pages 1–16, 2020.

[26] Huili Zhang and Yinfeng Xu. Online covering salesman problem. *Journal of Combinatorial Optimization*, pages 1–14, 2018.

Chapter 3

Exact algorithms and heuristics for capacitated covering salesman problems

---

The text presented below is a preprint prepared for submission and it is co-authored with Fábio Luiz Usberti and Celso Cavellucci. In this manuscript we propose a capacitated variant of the covering salesman problem in which the objective is to find a set of routes of minimum cost such that each vertex is visited or covered by a route, and that the total demand serviced by any vehicle does not exceed its capacity. We present integer linear programming formulations to the variant, and to tackle large instances that arise from real applications, a biased random-key genetic algorithm and a matheuristic to intensify the search are proposed. We also extend our integer linear programming formulation for the Multi-depot covering tour vehicle routing problem. Computational experiments show that our formulation outperformed the state-of-the-art exact methodology from the literature concerning optimality gaps.

This paper introduces the Capacitated Covering Salesman Problem (CCSP), approaching the notion of service by coverage in capacitated vehicle routing problems. In CCSP, locations where vehicles can transit are provided, some of which have customers with demands. The objective is to service customers through a fleet of vehicles based in a depot, minimizing the total distance traversed by the vehicles. CCSP is unique in the sense that customers, to be serviced, do not need to be visited by a vehicle. Instead, they can be serviced if they are within a coverage area of the vehicle. This assumption is motivated by applications in which some customers are unreachable (e.g., forbidden access to vehicles) or visiting every customer is impractical. In this work, optimization methodologies are proposed for the CCSP based on ILP (Integer Linear Programming) and BRKGA (Biased Random-Key Genetic Algorithm) metaheuristic. Computational experiments conducted on a benchmark of instances for the CCSP evaluate the performance of the methodologies with respect to primal bounds. Furthermore, our ILP formulation is extended in order to create a novel MILP (Mixed Integer Linear Programming) for the Multi-Depot Covering Tour Vehicle Routing Problem (MDCTVRP). Computational experiments show that the extended MILP formulation outperformed the previous state-of-the-art exact approach with respect to optimality gaps. In particular, optimal solutions were obtained for several previously unsolved instances.

3.1 Introduction

The Capacitated Vehicle Routing Problem (CVRP), initially proposed by Dantzig and Ramser [8], is one of the most well-known problems in combinatorial optimization. The goal of CVRP is to service the demands of a set of customers through a set of vehicles located in a depot, minimizing the total distance travelled. Each vehicle must depart and return to the depot, and cannot service more than its capacity [5].

The CVRP encompasses many variants with restrictions of time constraints, resources availability, and even customers accessibility, for example, regions of difficult means of entry to vehicles [12, 2]. The latter can be addressed by considering service by covering. A notion by which a customer can be serviced remotely as long as the customer is in the covering range of the vehicle. For example, in Figure 3.1 customers b and d are within the covering range of customer a. Also, customers a and e can be remotely serviced by vertex c, if there is enough remaining capacity in the corresponding vehicle.

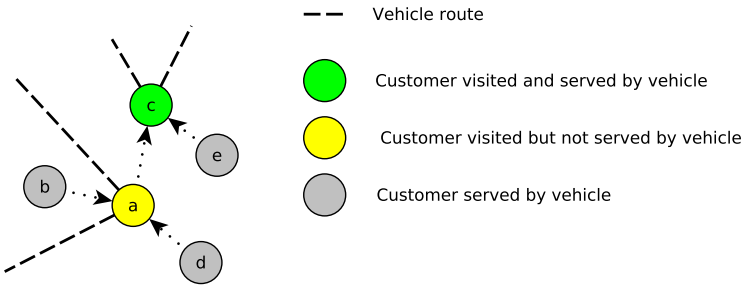


Figure 3.1: Example of covering ranges.

The first problem using the concept of servicing by coverage is the Covering Salesman Problem (CSP), by Current and Schilling [7], stated as follows. Given an undirected graph with cost attributed to the edges, the objective is to determine a minimum cost cycle such that every vertex out of the cycle is covered by at least one vertex in the cycle. The CSP generalizes the Travelling Salesman Problem (TSP) [3] in the case where each vertex only covers itself, from which follows that CSP is NP-hard.

Generalizations of the CSP were investigated in literature. Golden et al. [11] proposed a generalization of the CSP in which each vertex has a covering demand referring to the number of times it must be covered by the tour. Also, each vertex has a fixed cost that incurs from visiting it. The authors developed a heuristic with local search that explores exchange, removal, and insertion neighborhoods.

Gendreau et al. [10] investigated the Covering Tour Problem (CTP), a problem where the vertices are categorized by those that can be visited V, must be visited $T \subseteq V$, and cannot be visited W. The goal of the CTP is to obtain a minimum cost Hamiltonian cycle over a set of vertices $S \subseteq V$ containing all vertices in T and no vertices in W, and each vertex of W is covered by at least one vertex in S. Exact and heuristic methodologies were proposed to solve the problem.

Hachicha et al. [15] introduced the Multi-Vehicle Covering Routing Problem (m-CTP). It generalizes the CTP in the sense that there are multiple vehicles, and each route cannot exceed predefined length and number of vertices. The m-CTP was used as the basis to formulate a problem of locating distribution centers for humanitarian aid in disaster areas [24]. Methodologies to solve the m-CTP include branch-and-cut [14], column generation

[23], branch-and-price [17], constructive heuristics [15], evolutionary metaheuristic [14], variable neighborhood descent[18].

Allahyari et al. [2] proposed the the Multi-Depot Covering Tour Vehicle Routing Problem (MDCTVRP). The MDCTVRP is a combination of the Multi-Depot Vehicle Routing Problem (MDVRP) [27] and CSP. In the MDCTVRP, the demand of each customer can be served either by visiting the customer directly or by covering, i.e, the customer location is within a covering range of at least one visited customer. The authors developed two mixed integer programming formulations and a hybrid metaheuristic, combining Greedy Randomized Adaptive Search Procedure (GRASP), Iterated Local Search (ILS) and Simulated Annealing (SA).

It is worth noticing that the CSP does not have a multi-vehicle variant, as does the CTP. This work fills this gaps by proposing the Capacitated Covering Salesman Problem (CCSP), a NP-hard problem generalizing both the CVRP and the CSP. Vertices with non-negative demands must be covered by a set of capacitated vehicles, based at the depot. The goal is to find a minimum cost set of vehicle routes servicing all the demands. The CCSP represents a straightforward extension of the CSP where the service employed by the vehicles comes at a limited supply. At the same time, the CCSP generalizes the CVRP since covering provides an additional way to service each demand. It is worth pointing out the differences between m-CTP and CCSP:

- CCSP considers demands on the vertices;
- m-CTP forces some vertices to be visited ($T \subseteq V$);
- m-CTP constrains the routes by their lengths and number of vertices, while in CCSP the vehicle is capacitated by the amount of serviced demand.
- CCSP is a natural generalization of the CSP and the m-CTP generalizes the CTP.

Our contributions Two combinatorial optimization problems, the CSP and the VRP, are combined into a general framework to address routing problems with multiple vehicles and limited capacity in the context of service by covering. Mathematical formulations, using integer linear programming, are provided to represent these problems as a CCSP. The complexity of solving these problems optimally asks for heuristic methodologies to tackle large instances that arise from real applications. This work answers this demand by proposing a biased random-keys genetic algorithm to solve the CCSP, and a matheuristic to intensify the search. Furthermore, we extended our ILP to solve the MDCTVRP and conducted computational experiments on a benchmark of instances, comparing our formulation with the state-of-the-art exact methodology from literature. The proposed formulation outperformed the previous approach with respect to optimality gaps. Moreover, optimal solutions were proven for several previously unsolved instances.

The CCSP and MDCTVRP share common concepts of covering and vehicle capacity. Consequently, both can be modeled in a similar manner concerning serving remotely customers and demands served by the vehicle. Nevertheless, notable distinctions exist between these problems. In CCSP, a vehicle is not obligated to serve a customer during its visit, unlike in MDCTVRP, where the vehicle is required to serve a customer during each visit. Additionally, in CCSP there are customers with no specified demand, providing an opportunity for vehicles to use them to serve other customers remotely which have demand. In contrast, in MDCTVRP every customer is associated with a specific demand.

This paper is organized as follows. Section 3.2 formally defines the CCSP and MDCTVRP, presenting ILP formulations for the CCSP and a MILP formulation for the MDCTVRP. Section 3.3 describes the BRKGA, and the intra-route and inter-route intensification procedures. In Section 3.4, computational experiments are conducted on a representative set of instances, and results are analyzed and discussed. Section 3.5 gives the concluding remarks.

3.2 Mathematical Formulations

3.2.1 Models for the CCSP

Consider a complete undirected graph $G(V, E)$, where each vertex $v \in V$ has a demand d_v, each edge $e \in E$ has a metric cost c_e, and a depot vertex is denoted by v_0. Let $V_0 = V \setminus \{v_0\}$ and $V_d = \{v \in V : d_v > 0\}$. There are M homogeneous vehicles with capacity Q that must service all vertices with positive demand.

For each vertex $v \in V$, $C(v)$ is the set of vertices that covers v and $D(v)$ is the set of vertices that are covered by v. It is assumed that $v \in C(v)$ and $v \in D(v)$, $\forall v \in V$.

A route is a nonempty subset $R \subseteq E$ of edges for which the induced subgraph $G[R]$ is a simple cycle containing v_0. The goal of CCSP is to find M routes of minimum cost with the following constraints:

- each vertex is visited no more than once;
- each demand $d_v : v \in V_d$ is serviced by a route R, which implies in v or some vertex in $C(v)$ being visited by R, and the demand d_v being deducted from the capacity of the vehicle;
- the total demand serviced by any vehicle must not exceed its capacity Q.

Figure 3.2 shows an optimal solution for a CCSP instance. Routes are depicted with black lines; the blue triangle is v_0; red squares are vertices with positive demand; green points are visited vertices with no demand; arrows show which route serviced each demand.

The following ILP formulation $CCSP_1$ is proposed for the CCSP. We denote $\delta(v)$ as the edge cut-set of vertex v, and $\delta(S)$ the edge cut-set of a subset $S \subseteq V$. The formulation includes the following decision variables: $x_e \in \mathbb{Z}^+$ gives the number of times edge $e \in E$ is traversed; $y_v \in \{0, 1\}$ denotes if vertex $v \in V$ is visited (1) or not (0); $z_{uv} \in \{0, 1\}$ shows if vertex $u \in V_d$ is serviced through vertex $v \in C(u)$ (1) or not (0); $K \in \mathbb{Z}^+$ is the number of vehicles.

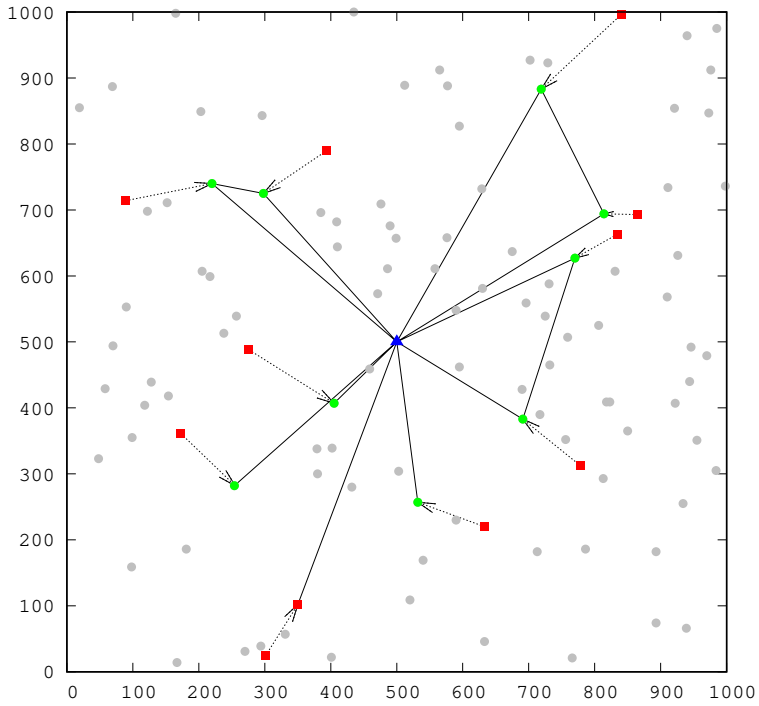


Figure 3.2: Optimal solution for the CCSP instance X-$n115$-$w11$-$c7$.

$(CCSP_1)$

$$MIN \quad \sum_{e \in E} c_e x_e, \tag{1}$$

subject to

$$\sum_{e \in \delta(v_0)} x_e = 2K, \tag{2}$$

$$\sum_{e \in \delta(v)} x_e = 2y_v \qquad \forall v \in V_0, \tag{3}$$

$$\sum_{v \in C(u)} y_v \geqslant 1 \qquad \forall u \in V_d, \tag{4}$$

$$z_{uv} \leqslant y_v \qquad \forall u \in V_d, \forall v \in C(u), \tag{5}$$

$$\sum_{v \in C(u)} z_{uv} = 1 \qquad \forall u \in V_d, \tag{6}$$

$$\sum_{e \in \delta(S)} x_e \geqslant \frac{2}{Q} \sum_{u \in V_d} \sum_{v \in (S \cap C(u))} d_u z_{uv} \qquad \forall S \subseteq V_0, \tag{7}$$

$$x_e \in \{0,1\} \qquad \forall e \notin \delta(v_0), \tag{8}$$

$$x_e \in \{0,1,2\} \qquad \forall e \in \delta(v_0), \tag{9}$$

$$y_v \in \{0,1\} \qquad \forall v \in V_0, \tag{10}$$

$$z_{uv} \in \{0,1\} \qquad \forall u \in V_d, \forall v \in C(u), \tag{11}$$

$$K \in \mathbb{Z}^+. \tag{12}$$

The objective function (1) minimizes the total cost of the routes. Constraints (2) state that the vertex depot is visited by all K routes. Constraints (3) ensure that the number of edges incident to a vertex $v \in V_0$ is 2 if v is visited or 0 otherwise. Constraints (4) impose that each vertex in V_d must be covered by at least one route. Constraints (5) state that if a vertex $u \in V_d$ is serviced by a vertex $v \in C(u)$, then v is visited. Constraints (6) ensure that every vertex $u \in V_d$ is serviced by a vertex $v \in C(u)$. Constraints (7) impose both the connectivity and the vehicle capacity by forcing into the solution a sufficient number of edges to each subset of vertices.

A second formulation, denominated $CCSP_2$, can be derived by eliminating variables y through variable substitution using constraints (3).

$(CCSP_2)$

$$MIN \quad \sum_{e \in E} c_e x_e,$$

subject to

$$\sum_{e \in \delta(v)} x_e \geqslant 2 z_{uv} \qquad \forall v \in V_0, \forall u \in V_d, \tag{13}$$

$$\sum_{e \in \delta(v)} x_e \leqslant 2 \sum_{u \in V_d} z_{uv} \qquad \forall v \in V_0, \tag{14}$$

$$\sum_{e \in \delta(v)} x_e \leqslant 2 \qquad \forall v \in V_0, \tag{15}$$

$(2), (6), (7), (8), (9), (11), (12).$

Constraints (13), (14), and (15) impose the correct number of edges incident to a vertex $v \in V_0$ (2 if visited or 0 otherwise).

Preliminary experiments have shown that, even though the $CCSP_2$ has fewer variables than the $CCSP_1$, the overall quality of the upper and lower bounds obtained by $CCSP_1$ is better than $CCSP_2$. Therefore, only the $CCSP_1$ formulation will be considered in the computational experiments.

3.2.2 Models for a Multi-depot Variant

The Multi-Depot Covering Tour Vehicle Routing Problem (MDCTVRP), proposed by Allahyari et al. [2], is defined next. Given a directed graph $G = (N, A)$, with vertices $N = N_c \cup N_d$, and arcs A. Each customer $i \in N_c = \{1, 2, ..., n_c\}$ has a positive demand d_i. Set $N_d = \{1, ..., n_d\}$ contains the depots. Each arc $(i, j) \in A$ has a positive traversing cost c_{ij}. Each customer has to be covered by a route. Set $C(v)$ represents the vertices that covers v. A cost $c'_{ij} > 0$ is attributed for servicing customer i through j. A set of identical vehicles $P = \{1, 2, ..., p\}$ is available, and Q is the vehicle capacity. Each depot $k \in N_d$ has a limited capacity H. Finally, to each depot is attributed a unique set $P_k = \{1, \ldots, p_k\}$ of vehicles. The objective of MDCTVRP is to find a minimum cost set of routes, such that all demands are covered, the vehicles and depots capacities are satisfied, and each vehicle starts and ends its route in the same depot.

Borrowing ideas from model $CCSP_1$ and from the flow-based formulation by Allahyari et al. [2], we propose a new MILP formulation for the MDCTVRP. The formulation,

henceforth denominated $MDCTVRP_m$, includes the following decision variables: x_{ij} denotes if arc $(i,j) \in A$ is traversed (1) or not (0); y_v represents if vertex $v \in N_c$ is visited (1) or not (0); z_{uv} shows if vertex $u \in N_c$ is serviced by vertex $v \in N_c$ (1) or not (0); f_{ij} gives the vehicle load while traversing arc (i,j). We represent SP as the set containing all simple paths connecting depots. Specifically, $SP_{(st)} \in SP$ denotes the set of all simple paths between depots s and t.

$(MDCTVRP_m)$

$$MIN \quad \sum_{(i,j)\in A} c_{ij}x_{ij} + \sum_{i\in N_c}\sum_{j\in N_c} c'_{ij}z_{ij}, \tag{16}$$

subject to

$$\sum_{j\in N_c} x_{jk} = \sum_{j\in N_c} x_{kj} \qquad \forall k \in N_d, \tag{17}$$

$$\sum_{j\in N_c} x_{kj} \leqslant |P_k| \qquad \forall k \in N_d, \tag{18}$$

$$\sum_{j\in N} x_{jv} = \sum_{j\in N} x_{vj} = y_v \qquad \forall v \in N_c, \tag{19}$$

$$\sum_{v\in C(u)} y_v \geqslant 1 \qquad \forall u \in N_c, \tag{20}$$

$$z_{uv} \leqslant y_v \qquad \forall u \in N_c, \forall v \in C(u), \tag{21}$$

$$\sum_{j\in N} x_{vj} \leqslant z_{vv} \qquad \forall v \in N_c, \tag{22}$$

$$\sum_{v\in C(u)} z_{uv} = 1 \qquad \forall u \in N_c, \tag{23}$$

$$\sum_{(i,j)\in SP_{(st)}} x_{ij} \leqslant \left|SP_{(st)}\right| - 1 \qquad \forall s,t \in N_d, s \neq t, \forall SP_{(st)} \in SP, \tag{24}$$

$$\sum_{j\in N} f_{ji} = \sum_{j\in N_c} d_j z_{ji} + \sum_{j\in N} f_{ij} \qquad \forall i \in N_c, \tag{25}$$

$$\sum_{i\in N_c} f_{ik} = 0 \qquad \forall k \in N_d, \tag{26}$$

$$f_{ij} \leqslant (Q - d_i)x_{ij} \qquad \forall (i,j) \in A : i \in N, j \in N_c, \tag{27}$$

$$d_j x_{ij} \leqslant f_{ij} \qquad \forall (i,j) \in A : i \in N, j \in N_c, \tag{28}$$

$$\sum_{i\in N_c} f_{ki} \leqslant H \qquad \forall k \in N_d, \tag{29}$$

$$x_{i,j} \in \{0,1\} \qquad \forall (i,j) \in A, \tag{30}$$

$$y_v \in \{0,1\} \qquad \forall v \in N_c, \tag{31}$$

$$z_{uv} \in \{0,1\} \qquad \forall u \in N_c, \forall v \in C(u), \tag{32}$$

$$f_{ij} \in \mathbb{R}^+ \qquad \forall (i,j) \in A. \tag{33}$$

The objective function (16) minimizes the total cost of the routes and allocations costs. For each depot $k \in N_d$, constraints (17) impose that the number of vehicles arriving k

must be equal to the number of vehicles leaving k. Constraints (18) bound the amount of vehicles arriving each depot. For each customer $v \in N_c$, constraints (19) state that the number of arcs arriving and leaving v is 1 if v is visited, or 0 otherwise. Constraints (20) impose that each vertex in N_c must be covered by at least one route. Constraints (21) state that a vertex $u \in N_c$ can only be serviced through a vertex $v \in C(u)$ if v is visited. Constraints (22) require that if a customer $v \in N_c$ is visited by a vehicle, then its demand is serviced by itself. Constraints (23) ensure that every vertex $u \in N_c$ is serviced by a vertex $v \in C(u)$. Constraints (24) prevent simple paths between depots, forcing each route to start and end in the same depot. Constraints (25) impose the flow conservation on each customer i. Constraints (26) ensure that the vehicle load is zero when returning to the depot. Constraints (27) and constraints (28) bound the vehicle load when traversing arc (i, j). Constraints (29) impose that the capacity of each depot is at most H.

The number of decision variables used in the new MILP formulation is $O(V^2)$ and the number of decision variables used in the model by Allahyari et al. [2] is $O(V^3)$. An $O(V^2)$ algorithm can separate constraints (24) for integer solutions using a lazy constraint strategy. Given a graph induced by an integer solution, the separation of constraints (24) is performed using Depth-First-Search (DFS) [6]. For every pair (i, j) such that $i, j \in N_d$, a DFS is performed starting from i. If j is reached, the edges from the path between i and j are retrieved and then a constraint (24) is added to the formulation.

3.3 BRKGA for Capacitated Covering Salesman Problem

Guided by the Darwinian principle of the survival of the fittest, the Biased Random Key Genetic Algorithm (BRKGA) [13] is an evolutionary metaheuristic in which a population of individuals, representing solutions of a combinatorial optimization problem, evolves towards the optimal.

Each individual is represented by a chromosome encoded as a vector, in which each allele is a random key uniformly drawn over the interval $[0, 1)$. The decoder method is the problem-specific component of the BRKGA which is responsible for mapping a chromosome into a solution.

An initial population of random chromosomes is created and forced into a selective pressure environment in which the best individuals are more likely to survive throughout the generations producing offsprings.

The BRKGA partitions the population into *elite* and *non-elite* sets, with sizes determined by fixed parameters. The elite set is composed by the best individuals; all the remaining individuals form the non-elite set, including the *mutants*, i.e., random chromosomes introduced into the population as a form of diversification.

In each generation, the BRKGA executes the following steps:

1. Decode the chromosomes, evaluating their fitness;
2. Identify the best individuals to form the elite set;
3. Preserve the elite set into the population for the next generation;
4. Introduce the mutants in the next generation;

5. Generate offsprings through the crossover of elite and non-elite chromosomes, inserting them in the next generation.

The BRKGA has demonstrated its efficacy as a robust method for addressing various routing problems [25, 21, 1, 20, 9]. In this sense, the following sections describe how the BRKGA can be employed to solve the CCSP.

3.3.1 Solution encoding

The solution is encoded as a vector $\mathcal{X} = (x_1, ..., x_n)$ of size $n = |V_d|$, where x_i is a random number in the interval $[0, 1)$, for $i = 1, \ldots, n$. Each element of $\mathcal{X}$ represents a vertex of V_d.

3.3.2 Decoder function

The decoder function takes as input a vector $\mathcal{X}$ and returns a feasible solution for the CCSP represented by a set of routes $\mathcal{R}$. Let $\mathcal{X}'$ be the vector resulting by sorting the keys of $\mathcal{X}$ in non-decreasing order.

The proposed decoder for the CCSP has two phases, described in the following sections.

First phase - Best Fit Algorithm

The minimum number of vehicles required to service all demands can be determined by solving a *Bin Packing Problem* (BPP) [19], which is NP-hard. Our decoder assigns vertices from V_d to vehicles by solving the BPP approximately, using the Best Fit Algorithm (BFA). The BFA assigns each vertex to a vehicle with the least residual capacity that can still service the vertex; if no such vehicle exists, a new one is assigned (Figure 3.3).

Algorithm 5 presents the BFA pseudo-code applied in the decoder, which can be implemented using self-balancing search trees leading to a worst-time complexity of $O(n \lg n)$.

Second phase - route construction

Consider that route R_m of vehicle m is a sequence of vertices represented as $R_m = \{R_m(0), \ldots, R_m(r_m + 1)\}$, where $R_m(i)$ and r_m are, respectively, the i-th vertex visited by vehicle m and the number of vertices in V_0 visited by vehicle m. The route starts and ends at the depot, i.e., $R_m(0) = R_m(r_m + 1) = v_0$.

The second phase of the decoder creates a route for each vehicle by using the following insertion cost function,

$$g(v, R_m) = \min_{i=\{0,\ldots,r_m\}} \big\{c_{(u,v)} + c_{(v,w)} - c_{(u,w)} : u = R_m(i), w = R_m(i+1)\big\},$$

which gives the minimum cost of inserting a vertex v into a route R_m.

For each vehicle $m \in \mathcal{M}$ and each vertex $u \in A_m$, all unvisited vertices $v \in C(u)$ are considered to be included in route R_m by checking the value of $g(v, R_m)$ and taking the vertex resulting in the least cost increment.

Algorithm 6 presents the route construction pseudo-code used in our decoder.

A vertex v is called *redundant* if when removed from a route it does not change the solution feasibility. It occurs when the vertices serviced by v can all be serviced by other vertices in the solution without violating the capacity of any involved vehicle. After

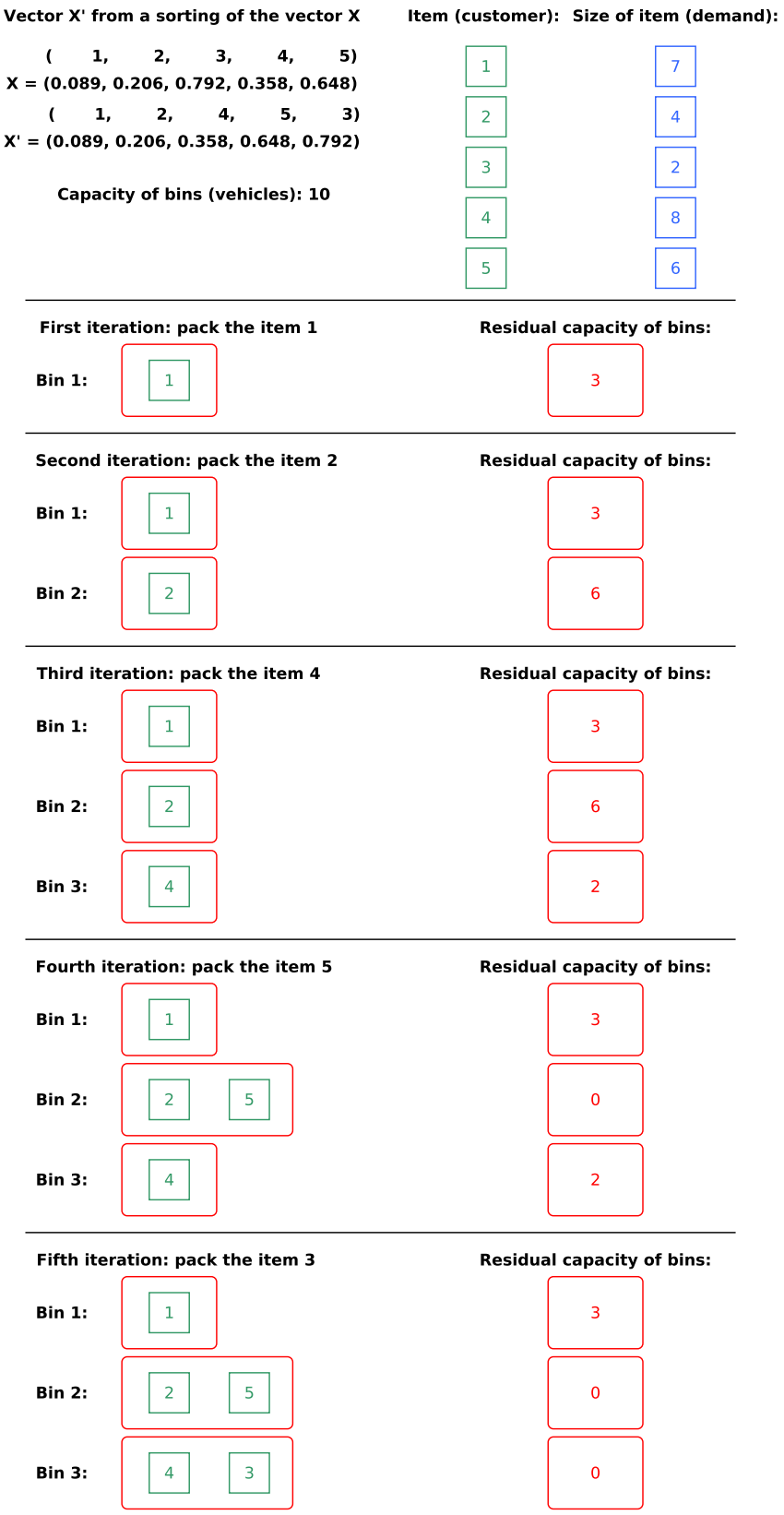


Figure 3.3: Example of Best Fit Algorithm.

Algorithm 5 Best fit algorithm

Input: a vector $\mathcal{X}^{'}$.
Output: a set of vehicles $\mathcal{M} = \{1, \ldots, M\}$ and their assigned vertices $\mathcal{A} = \{A_1, \ldots, A_M\}$.

1: $m \leftarrow 0; \mathcal{M} \leftarrow \{\emptyset\}; \mathcal{A} \leftarrow \{\emptyset\};$
2: **for** each $x'_i \in \mathcal{X}^{'}$ **do**
3: $v \leftarrow getVertex(i)$ – returns the vertex associated to element x'_i;
4: **if** $\left(\exists m \in \mathcal{M} : \sum_{u \in A_m} d_u \leqslant Q - d_v \right)$ **then**
5: $m \leftarrow \arg\max_{m' \in \mathcal{M}} \left\{ \sum_{u \in A'_m} d_u : \sum_{u \in A'_m} d_u \leqslant Q - d_v \right\};$
6: $A_m \leftarrow A_m \cup \{v\};$
7: **else**
8: $m \leftarrow m + 1;$
9: $\mathcal{M} \leftarrow \mathcal{M} \cup \{m\};$
10: $A_m \leftarrow \{v\};$
11: $\mathcal{A} \leftarrow \mathcal{A} \cup \{A_m\}$

Algorithm 6 Construction of Routes

Input: a set of vehicles $\mathcal{M} = \{1, \ldots, M\}$ and their assigned vertices $\mathcal{A} = \{A_1, \ldots, A_M\}$.
Output: a set of routes $\mathcal{R} = \{R_1, \ldots, R_m\}$.

1: $\mathcal{R} \leftarrow \emptyset;$
2:
3: **for** each vehicle $m \in \mathcal{M}$ **do**
4: $R_m(0) \leftarrow \{v_0\};$
5:
6: $r_m \leftarrow 0$
7:
8: **for** each $v \in A_m$ **do**
9: $S \leftarrow \left\{ u \in C(v) : u \neq R_{m'}(u'), \forall m' \in \{1, \ldots, m\}, \forall u' \in \{1, \ldots, r_{m'}\} \right\}$ – set of unvisited candidates to service v
10:
11: $u \leftarrow \arg\min_{u' \in S} \left\{ g(u', R_m) \right\}$ – find best candidate
12:
13: $insert(u, R_m)$ – insert u in the best position of route R_m
14:
15: $r_m \leftarrow r_m + 1$
16:
17: $R_m(r_m) \leftarrow \{v_0\};$
18:
19: $\mathcal{R} \leftarrow \mathcal{R} \cup \{R_m\};$
20:

applying Algorithm 6, the decoder greedily removes redundancies by considering their cost decrease, until a maximal set of redundancies is removed.

3.3.3 Intra-Route

Once the BRKGA stopping criteria is triggered, the Lin-Kernighan (LK) heuristic [22] performs the final intra-routes improvements. The LK heuristic is based on k-opt neighborhood, which consists in applying up to k edge exchanges, and it is considered one of the best local searches for the Traveling Salesman Problem.

3.3.4 Inter-Route Intensification

Following the ideas of Sartori and Buriol [26], this paper proposes a matheuristic for the CCSP using a formulation with covering and packing constraints. Let F be the set of all CCSP feasible routes, a_{if} the covering matrix where for each pair (i, f), $i \in V_d$ and $f \in F$, $a_{if} = 1$ if and only if vertex i is serviced by route f, and b_{if} the visiting matrix where for each pair (i, f), $i \in V_0$ and $f \in F$, $b_{if} = 1$ if and only if vertex i is visited by route f. The formulation includes the binary variable λ_f, which denotes whether the feasible route $f \in F$ is used (1) or not (0).

The formulation reads as follows:

$$
\begin{aligned}
&(Matheuristic)\\
&MIN \quad \sum_{f \in F} c_f \lambda_f, && && (34)\\
&\text{subject to}\\
&\sum_{f \in F} a_{if} \lambda_f \geqslant 1 && \forall i \in V_d && (35)\\
&\sum_{f \in F} b_{if} \lambda_f \leqslant 1 && \forall i \in V_0 && (36)\\
&\lambda_f \in \{0, 1\} && \forall f \in F.
\end{aligned}
$$

The objective function (34) minimizes the costs of the routes. Constraints (35) ensure that each vertex in V_d must be serviced by at least one route, while constraints (36) impose that every vertex in V_0 must be visited by at most one route. Considering that the cardinality of F grows exponentially, in this work we generate a pool of routes $F^{'}$ in the Matheuristic formulation. The pool of routes contains a set of CCSP feasible routes generated as follows. First, an exhaustive search is conducted to find every optimal route that services up to three vertices. All of these routes are included into $F^{'}$. The remaining routes of $F^{'}$ are filled with the elite individuals from the BRKGA generations, starting from the last generation and continuing until either the size limit of $F^{'}$ is reached or all elite individuals from each BRKGA generation are added to $F^{'}$.

3.4 Computational Experiments

3.4.1 Instances Benchmark

The instances for the CCSP were obtained from the CVRP instances created by Christofides and Eilon [4] and Uchoa et al. [29], containing between 101 and 303 vertices, and named as *E-nA-kB* and *X-nA-kB*, respectively. The symbol A gives the number of vertices (with the depot), and B represents the number of vehicles required to service all the demands.

To generate CCSP instances, the following parameters were used:

- $|V_d|$: number of vertices with demand;
- $\left|D(v)\right|$: covering size, where $v \in V_0$.

The $|V_d|$ parameter varied in 10%, 20%, and 40% of n. The vertices with demand consist of the first $|V_d|$ vertices, excluding the depot, of the CVRP instance. It is worth mentioning that all demands associated with V_d remain unchanged in relation to CVRP instance. Similarly, the vehicle capacity Q in the CCSP instance remains the same as in the CVRP instance.

For each vertex $v \in V_0$, the set $D(v)$ is defined by the closest vertices from v. The cardinality of $D(v)$ varied in 7, 9, and 11.

For each CVRP instance, all pairings of $|V_d|$ and $\left|D(v)\right|$ were considered, resulting in nine combinations and a benchmark of 495 instances. A pre-processing was conducted in the set of instances to remove any vertex $v \in V_0$ such that $D(v) \cap V_d = \emptyset$.

The MDCTVRP instances were created by Allahyari et al. [2], and they are divided in small (120 instances) and large (160 instances). The small instances contain up to 30 vertices, and the large instances have up to 90 vertices. In small instances, the vehicle capacity fluctuates between 140 and 150, while in large instances, the vehicle capacity alternates between 160 and 170. The small instances are divided into three categories, and the large instances are divided into four categories. Each category has eight different groups of instances. The instances are named InputXYZT, where "X", "Y", "Z", and "T" give the category, number of depots, vehicle capacity, and the coverage coefficient (which defines the cost of serving a vertex), respectively. Five instances were generated for each group of instances.

3.4.2 Computational Settings

The ILP and MILP formulations were implemented and solved using Gurobi 8.1.1 version. The execution time limit were set to a one hour, except for the $MDCTVRP_m$ formulation, which was set to two hours following Allahyari et al. [2]. The experiments were conducted on a PC under Ubuntu 10.12, and CPU Intel Xeon(R) Silver 3114 2.20 GHz, with 32GB of RAM. The BRKGA developed for the CCSP used the C++ framework from Resende and Toso [28]. The parameters used by the BRKGA are listed in Table 3.1. The implementation of the LK heuristic proposed by Helsgaun [16] was employed. The matheuristic adopted a size limit for the pool F' of 1 million routes.

3.4.3 Evaluated Methodologies

Five methodologies were implemented and evaluated in the computational experiments:

Table 3.1: BRKGA parameters.

Parameter	Value
Population size	1000
Fraction of population to be elite individuals	40%
Fraction of population to be replaced by mutants	20%
Crossover probability	70%

- $\mathbb{CCSP}_1$: solution of the $CCSP_1$ model, initially ignoring Constraints (7), and later including them in the formulation using a ***lazy constraint*** strategy;
- $\mathbb{BRKGA}$: implementation of the BRKGA for the CCSP described in Section 3.3;
- $\mathbb{CCSP}_{1s}$: same as $\mathbb{CCSP}_1$, however the solution obtained by the BRKGA is given as warm start for the $CCSP_1$ model;
- *Matheuristic*: solution of the matheuristic described in Subsection 3.3.4;
- $\mathbb{MDCTVRP}_m$: solution of the $MDCTVRP_m$ formulation, initially ignoring Constraints (24), and including them on-demand in the formulation using a ***lazy constraint*** strategy.

The $\mathbb{MDCTVRP}_m$ was compared with the flow-based formulation (F_{flow}) and node-based formulation (F_{node}), both proposed by Allahyari et al. [2], which are the state-of-the-art exact methodologies for the MDCTVRP, to the best of our knowledge.

3.4.4 Results for the CCSP

Full experimental data, results, instances, and source codes are available on-line[1]. The results of the computational experiments show that the $\mathbb{CCSP}_1$ methodology was able to obtain upper bounds for 430 out of 495 instances. In addition, the $\mathbb{CCSP}_1$ methodology proved optimality for 71 instances size up to 101 vertices. Analyzing the results of the $\mathbb{BRKGA}$ methodology, we can note that for 407 instances, the obtained solutions were better than the upper bounds obtained by $\mathbb{CCSP}_1$.

With respect to methodologies $\mathbb{CCSP}_{1s}$ and *Matheuristic*, the results show that the matheuristic was more effective to improve the solutions obtained from $\mathbb{BRKGA}$. *Matheuristic* methodology improved the BRKGA solutions for 187 out of 495 instances, while $\mathbb{CCSP}_{1s}$ improved for 88 instances. The average cost of improvements made by $\mathbb{CCSP}_{1s}$ and *Matheuristic* on BRKGA solutions were approximately 1.69% and 2.3%, respectively.

Figure 3.4 shows, for each methodology, the percentage of solved instances in function of the deviation from the best upper bound ($(\frac{UB-BestUB}{UB}) * 100$). The performance profile clearly shows the $\mathbb{BRKGA}$ dominating $\mathbb{CCSP}_1$ with respect to upper bounds. The $\mathbb{BRKGA}$ obtained the best solutions for approximately 48% of instances, while the $\mathbb{CCSP}_1$ obtained for approximately 16% of instances. The $\mathbb{CCSP}_{1s}$ was able to improve the warm start $\mathbb{BRKGA}$ solution for several instances, obtaining the best solutions for approximately 58% of instances. Finally, comparing methodologies *Matheuristic* and $\mathbb{CCSP}_{1s}$, *Matheuristic* was more effective in improving the BRKGA solutions. The *Matheuristic* methodology obtained the best solution for approximately 74% of instances, outperforming $\mathbb{CCSP}_{1s}$.

[1] `http://www.ic.unicamp.br/~fusberti/problems/ccsp`

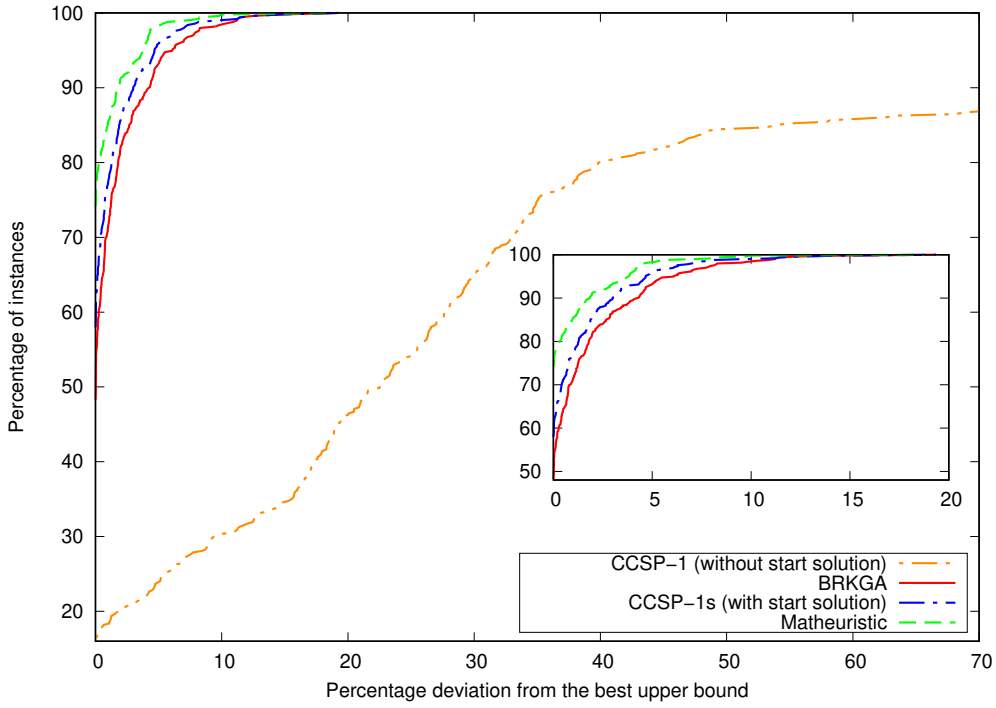


Figure 3.4: Performance profiles in terms of deviation (%) from the best upper bound.

3.4.5 Results for the MDCTVRP

Tables 3.2 and 3.3 report the results for the small and large instances, respectively. Table 3.2 reports for each methodology and for each group of small-size instances, the average gap (Avg.gap), the number of optimal solutions (#Opt), and the average running time (Avg.time).

For small instances, the overall optimality gaps were 0.10%, 8.99%, and 30.28% for the $\mathbb{MDCTVRP}_m$, F_{flow}, and F_{node} methodologies, respectively. From 120 small instances, 117, 10, and 4 optimal solutions were obtained by $\mathbb{MDCTVRP}_m$, F_{flow}, and F_{node}, respectively.

Table 3.3 gives the results from $\mathbb{MDCTVRP}_m$ for each group of large-size instances. The column group $\mathbb{MDCTVRP}_m$ reports the averages upper bound (Avg.ub), lower bound (Avg.lb), optimality gap (Avg.gap), running time (Avg.time), and number of optimal solutions (#Opt). The column group GRASP x ILS [2] reports the results obtained by the hybrid meta-heuristic proposed by Allahyari et al. [2]. "Avg" gives the average cost obtained over five executions for each instance. Avg.gap$_{LP}$ and Avg.gap $_{\mathbb{MDCTVRP}_m}$ give the "Avg" gap from the linear relaxation of F_{flow} formulation, and the average lower bound (Avg.lb) of $\mathbb{MDCTVRP}_m$, respectively.

It should be noticed that, due to the large number of variables, Allahyari et al. [2] have not executed experiments on large instances with F_{flow}. The computational results have shown that $\mathbb{MDCTVRP}_m$ achieved good lower bounds for large. Previously, the GRASP x ILS overall gap with respect to the linear programming relaxation of F_{flow} was 20.98%, while with the new lower bounds obtained by $\mathbb{MDCTVRP}_m$ the GRASP x ILS overall gap was improved to 7.98%. It is worth noting that even though $\mathbb{MDCTVRP}_m$ is an exact methodology, the upper bounds were close to the solutions cost obtained by the GRASP x ILS. More specifically, the solutions cost obtained by GRASP x ILS are, on average, only 1.37% apart from the upper bounds obtained by $\mathbb{MDCTVRP}_m$.

Table 3.2: Results of computational experiments for the small-size instances.

		$\mathbb{MDCTVRP}_m$			F_{flow}			F_{node}		
Category	Group	Avg.gap	#Opt	Avg.time	Avg.gap	#Opt	Avg.time	Avg.gap	#Opt	Avg.time
1	Input1000	0.00	5	8.00	6.37	0	7200.00	22.25	0	7200.00
	Input1001	0.00	5	8.80	6.38	0	7200.00	21.29	0	7200.00
	Input1010	0.00	5	6.60	1.74	2	6015.00	17.71	0	7200.00
	Input1011	0.00	5	5.20	2.48	1	7066.00	18.63	0	7200.00
	Input1100	0.00	5	7.00	4.96	2	5621.00	14.19	1	7200.00
	Input1101	0.00	5	6.80	4.87	1	5848.00	12.44	1	7200.00
	Input1110	0.00	5	5.80	3.14	2	5504.00	13.85	1	7200.00
	Input1111	0.00	5	6.20	2.03	2	5220.00	11.57	1	7200.00
2	Input2000	0.00	5	140.20	10.56	0	7200.00	42.30	0	7200.00
	Input2001	0.00	5	184.60	10.81	0	7200.00	38.64	0	7200.00
	Input2010	0.00	5	70.20	10.39	0	7200.00	37.45	0	7200.00
	Input2011	0.00	5	115.40	11.14	0	7200.00	33.94	0	7200.00
	Input2100	0.00	5	66.60	7.98	0	7200.00	31.33	0	7200.00
	Input2101	0.00	5	138.20	7.63	0	7200.00	27.85	0	7200.00
	input2110	0.00	5	73.20	9.02	0	7200.00	28.45	0	7200.00
	Input2111	0.00	5	44.60	9.55	0	7200.00	27.43	0	7200.00
3	Input3000	1.01	4	2653.20	14.48	0	7200.00	50.29	0	7200.00
	Input3001	0.71	4	2853.20	14.93	0	7200.00	44.15	0	7200.00
	Input3010	0.00	5	1865.80	15.23	0	7200.00	44.46	0	7200.00
	Input3011	0.58	4	1923.60	14.29	0	7200.00	41.20	0	7200.00
	Input3100	0.00	5	288.20	12.95	0	7200.00	39.75	0	7200.00
	Input3101	0.00	5	246.40	11.83	0	7200.00	35.90	0	7200.00
	Input3110	0.00	5	383.60	12.25	0	7200.00	36.70	0	7200.00
	Input3111	0.00	5	492.20	10.85	0	7200.00	35.00	0	7200.00
Average		0.10		483.07	8.99		6869.75	30.28		7200.00

Table 3.3: Results of computational experiments for the large-size instances.

		$\mathbb{MDCTVRP}_m$					GRASP x ILS [2]		
Category	Group	Avg.ub	Avg.lb	Avg.gap	#Opt	Avg.time	Avg	Avg.gap$_{LP}$	Avg.gap$_{\mathbb{MDCTVRP}_m}$
4	Input4000	799.45	751.98	6.22	0	7200.00	796.21	19.30	5.88
	Input4001	808.23	764.91	5.55	0	7200.00	806.95	18.57	5.50
	Input4010	787.18	729.22	7.86	0	7200.00	775.66	19.68	6.37
	Input4011	792.13	744.53	6.26	0	7200.00	787.64	19.12	5.79
	Input4100	737.27	692.92	6.24	0	7200.00	733.72	21.90	5.89
	Input4101	750.94	701.21	6.98	0	7200.00	745.62	20.91	6.33
	Input4110	721.12	678.30	6.09	1	6650.80	711.94	21.26	4.96
	Input4111	730.01	689.79	5.72	0	7200.00	728.17	21.03	5.56
5	Input5000	899.58	815.45	10.27	0	7200.00	880.01	19.80	7.92
	Input5001	903.58	829.00	8.95	0	7200.00	895.99	18.93	8.08
	Input5010	869.08	784.04	10.82	0	7200.00	857.27	21.41	9.34
	Input5011	886.07	800.62	10.63	0	7200.00	873.99	20.46	9.16
	Input5100	810.75	732.59	10.65	0	7200.00	796.88	22.12	8.78
	Input5101	827.35	748.43	10.52	0	7200.00	813.70	20.85	8.72
	Input5110	785.86	711.22	10.51	0	7200.00	777.99	23.23	9.39
	Input5111	804.80	728.69	10.37	0	7200.00	794.78	21.71	9.07
6	Input6000	1005.58	922.85	8.88	0	7200.00	997.18	20.01	8.05
	Input6001	1037.50	936.43	10.75	0	7200.00	1016.10	19.43	8.51
	Input6010	987.04	898.72	9.63	0	7200.00	968.81	20.59	7.80
	Input6011	998.51	916.29	8.83	0	7200.00	989.60	20.16	8.00
	Input6100	955.65	875.82	9.07	0	7200.00	946.88	20.63	8.11
	Input6101	984.79	893.52	10.09	0	7200.00	967.23	20.11	8.25
	Input6110	936.80	848.97	10.27	0	7200.00	920.24	21.58	8.40
	Input6111	953.62	867.56	9.88	0	7200.00	939.51	20.77	8.29
7	Input7000	1031.77	930.53	10.79	0	7200.00	1013.92	21.35	8.96
	Input7001	1047.06	935.57	11.88	0	7200.00	1019.34	20.89	8.95
	Input7010	1012.71	905.58	11.78	0	7200.00	989.22	21.90	9.24
	Input7011	1018.47	910.80	11.83	0	7200.00	995.08	21.47	9.25
	Input7100	933.58	842.19	10.81	0	7200.00	918.91	22.90	9.11
	Input7101	948.35	849.03	11.65	0	7200.00	924.61	22.39	8.90
	Input7110	917.68	830.41	10.51	0	7200.00	904.92	23.56	8.97
	Input7111	919.48	832.89	10.37	0	7200.00	915.05	23.24	9.86
Average		893.81	815.63	9.40			881.35	20.98	7.98

3.5 Final Remarks

This work proposes the Capacitated Covering Salesman Problem (CCSP), a problem that approaches the notion of coverage in vehicle routing problems. Two ILP formulations and a BRKGA are proposed to solve the CCSP. From the set of instances for the CVRP [4, 29], a benchmark of instances for CCSP was generated.

Computational experiments conducted on a benchmark of 198 instances for CCSP evaluated the ILP formulation and the BRKGA. The results show the effectiveness of the BRKGA in obtaining upper bounds for all instances. The $CCSP_1$ obtained optimal solutions for 71 instances, with up to 101 vertices.

Furthermore, a new MILP formulation is proposed for the Multi-Depot Covering Tour Vehicle Routing Problem (MDCTVRP). Computational experiments were conducted on a benchmark of 280 instances from literature. The overall results show unequivocally the new formulation outperforming the best known exact methodology from literature, obtaining 118 new optimal solutions and improving all known lower bounds.

Future works should focus on valid inequalities and a branch-and-cut framework for the solution of CCSP and MDCTVRP. Another promising field of research is to consider multi-objective vehicle routing problem with covering range being the additional objective function.

3.6 Acknowledgments

This work was supported by CAPES, CNPq, and Fapesp (grants 140960/2017-1, 314384/2018-9, 435520/2018-0, 2015/11937-9).

References

[1] Levi R Abreu, Roberto F Tavares-Neto, and Marcelo S Nagano. A new efficient biased random key genetic algorithm for open shop scheduling with routing by capacitated single vehicle and makespan minimization. *Engineering Applications of Artificial Intelligence*, 104:104373, 2021.

[2] Somayeh Allahyari, Majid Salari, and Daniele Vigo. A hybrid metaheuristic algorithm for the multi-depot covering tour vehicle routing problem. *European Journal of Operational Research*, 242(3):756–768, 2015.

[3] David L. Applegate, Robert E. Bixby, Vasek Chvatal, and William J. Cook. *The Traveling Salesman Problem: A Computational Study (Princeton Series in Applied Mathematics)*. Princeton University Press, Princeton, NJ, USA, 2007.

[4] Nicos Christofides and Samuel Eilon. An algorithm for the vehicle-dispatching problem. *Journal of the Operational Research Society*, 20(3):309–318, 1969.

[5] Jean-François Cordeau, Gilbert Laporte, Martin WP Savelsbergh, and Daniele Vigo. Vehicle routing. *Handbooks in operations research and management science*, 14:367–428, 2007.

[6] Thomas H Cormen, Charles E Leiserson, Ronald L Rivest, and Clifford Stein. *Introduction to algorithms*. MIT press, 2022.

[7] John R Current and David A Schilling. The covering salesman problem. *Transportation science*, 23(3):208–213, 1989.

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

[9] Saúl Domínguez-Casasola, José Luis González-Velarde, Yasmín Á Ríos-Solís, and Kevin Alain Reyes-Vega. The capacitated family traveling salesperson problem. *International Transactions in Operational Research*, 2023.

[10] Michel Gendreau, Gilbert Laporte, and Frédéric Semet. The covering tour problem. *Operations Research*, 45(4):568–576, 1997.

[11] Bruce Golden, Zahra Naji-Azimi, S Raghavan, Majid Salari, and Paolo Toth. The generalized covering salesman problem. *INFORMS Journal on Computing*, 24(4):534–553, 2012.

[12] Bruce L Golden, Subramanian Raghavan, and Edward A Wasil. *The vehicle routing problem: latest advances and new challenges*, volume 43. Springer Science & Business Media, 2008.

[13] José Fernando Gonçalves and Mauricio GC Resende. Biased random-key genetic algorithms for combinatorial optimization. *Journal of Heuristics*, 17(5):487–525, 2011.

[14] Minh Hoang Ha, Nathalie Bostel, André Langevin, and Louis-Martin Rousseau. An exact algorithm and a metaheuristic for the multi-vehicle covering tour problem with a constraint on the number of vertices. *European Journal of Operational Research*, 226(2):211–220, 2013.

[15] Mondher Hachicha, M John Hodgson, Gilbert Laporte, and Frédéric Semet. Heuristics for the multi-vehicle covering tour problem. *Computers & Operations Research*, 27(1):29–42, 2000.

[16] Keld Helsgaun. An effective implementation of the lin–kernighan traveling salesman heuristic. *European Journal of Operational Research*, 126(1):106–130, 2000.

[17] Nicolas Jozefowiez. A branch-and-price algorithm for the multivehicle covering tour problem. *Networks*, 64(3):160–168, 2014.

[18] Manel Kammoun, Houda Derbel, Mostapha Ratli, and Bassem Jarboui. An integration of mixed vnd and vns: the case of the multivehicle covering tour problem. *International Transactions in Operational Research*, 24(3):663–679, 2017.

[19] H Kellerer, U Pferschy, and D Pisinger. *Knapsack Problems*, volume 1. Springer-Verlag Berlin Heidelberg, 2004.

[20] Alberto F Kummer, Olinto CB de Araújo, Luciana S Buriol, and Mauricio GC Resende. A biased random-key genetic algorithm for the home health care problem. *International Transactions in Operational Research*, 2022.

[21] Alberto F Kummer N, Luciana S Buriol, and Olinto CB de Araújo. A biased random key genetic algorithm applied to the vrptw with skill requirements and synchronization constraints. In *Proceedings of the 2020 Genetic and Evolutionary Computation Conference*, pages 717–724, 2020.

[22] Shen Lin and Brian W Kernighan. An effective heuristic algorithm for the traveling-salesman problem. *Operations research*, 21(2):498–516, 1973.

[23] Keisuke Murakami. A column generation approach for the multi-vehicle covering tour problem. In *Automation Science and Engineering (CASE), 2014 IEEE International Conference on*, pages 1063–1068. IEEE, 2014.

[24] Zara Naji-Azimi, Jacques Renaud, Angel Ruiz, and Majid Salari. A covering tour approach to the location of satellite distribution centers to supply humanitarian aid. *European Journal of Operational Research*, 222(3):596–605, 2012.

[25] Efrain Ruiz, Valeria Soto-Mendoza, Alvaro Ernesto Ruiz Barbosa, and Ricardo Reyes. Solving the open vehicle routing problem with capacity and distance constraints with a biased random key genetic algorithm. *Computers & Industrial Engineering*, 133:207–219, 2019.

[26] Carlo S Sartori and Luciana S Buriol. A matheuristic approach to the pickup and delivery problem with time windows. In *International Conference on Computational Logistics*, pages 253–267. Springer, 2018.

[27] Frank A Tillman. The multiple terminal delivery problem with probabilistic demands. *Transportation Science*, 3(3):192–204, 1969.

[28] Rodrigo F Toso and Mauricio GC Resende. A c++ application programming interface for biased random-key genetic algorithms. *Optimization Methods and Software*, 30(1):81–93, 2015.

[29] Eduardo Uchoa, Diego Pecin, Artur Pessoa, Marcus Poggi, Thibaut Vidal, and Anand Subramanian. New benchmark instances for the capacitated vehicle routing problem. *European Journal of Operational Research*, 257(3):845–858, 2017.

Chapter 4

Routing Electric Vehicles with Remote Servicing

The paper presented next is a full chapter published in Modeling and Optimization in Green Logistics book in 2020 and it is co-authored by Rafael Kendy Arakaki, Lucas Porto Maziero, Matheus Diógenes Andrade, Vitor Mitsuo Fukushigue Hama, and Fábio Luiz Usberti (DOI: `https://doi.org/10.1007/978-3-030-45308-4`). In this text we introduce a new variant of the Vehicle Routing Problem with covering constraints and electric vehicles. Exact and heuristic algorithms are proposed to solve the problem. The efficacy of the suggested approaches is demonstrated through computational experiments, offering valuable insights for decision-making concerning electric vehicle-based transportation.

This paper introduces a problem called Electric Capacitated Covering Tour Problem (ECCTP), a variant of the Vehicle Routing Problem that allows customers demands to be serviced remotely by electric vehicles with limited battery range and that recharge at Alternative Fuel Stations (AFSs). The ECCTP integrates two research areas: vehicle routing problems and green logistics. We propose a Mixed Integer Linear Programming (MILP) mathematical formulation and a Biased Random-Key Genetic algorithm (BRKGA) metaheuristic for the ECCTP. A set of benchmark instances from literature is adapted for the problem. Computational experiments show the effectiveness of the proposed methods while providing useful information for the decision-making on transportation operated by electric vehicles.

4.1 Introduction

In the Vehicle Routing Problem (VRP) [12] a set of customers must be serviced by a homogeneous fleet of vehicles with limited capacity, each vehicle starting from a common depot. The objective is to minimize the total cost of vehicle routes. Many variants of the VRP were studied in the literature where additional constraints related to scheduling, budget, fleet heterogeneity and others are considered [28].

In another research direction, green logistics have emerged as a discipline focused on the planning and operation of logistic systems aiming high energy efficiency and low levels of carbon emission [17]. Among VRP variants within this discipline, there is the Green Vehicle Routing Problem (G-VRP), comprehending additional challenges associated with

operating a fleet of Alternative Fuel Vehicles (AFV) and incorporating stops at Alternative Fuel Stations (AFS) [8]. One of the biggest challenges modeled by G-VRP is the reduced number of alternative fuel stations and the low range of alternative fuel vehicles. A G-VRP solution example is depicted in Figure 4.1, in which each customer is visited by a route and some AFS are visited more than once for recharging.

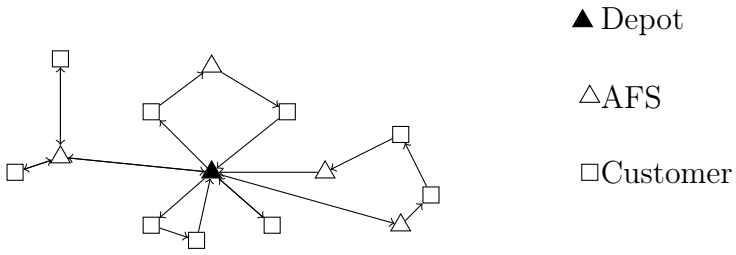


Figure 4.1: G-VRP solution example

In another branch of VRP variants, [1] investigated the covering tour VRP, a problem where some customers may be located in regions of difficult access but can be serviced by covering: the vehicle visits one of the neighboring customers instead of directly visiting the customer. This problem has application in determining post-boxes locations within a set of candidate locations, finding optimal collection routes [16]. Another application is to design routes for mobile health care delivery teams where services are rendered at a number of locations by medical teams, and the population living outside these locations must travel on foot to reach them [10, 26].

This work merges two branches of vehicle routing problems, the G-VRP and the covering tour VRP, resulting in a new problem called Electric Capacitated Covering Tour Problem (ECCTP). In the ECCTP, a homogeneous fleet of electric vehicles are used to service the demands of customers. A customer can be serviced by any vehicle visiting a node inside the neighborhood of that customer. An empty neighborhood implies that the corresponding customer must be visited in order to be serviced. Figure 4.2 presents an ECCTP example in which some vertices are customers and others are traffic vertices. A traffic vertex does not have demand and may be visited or not. Both traffic and customer vertices can be used to service demands of neighboring customers by covering.

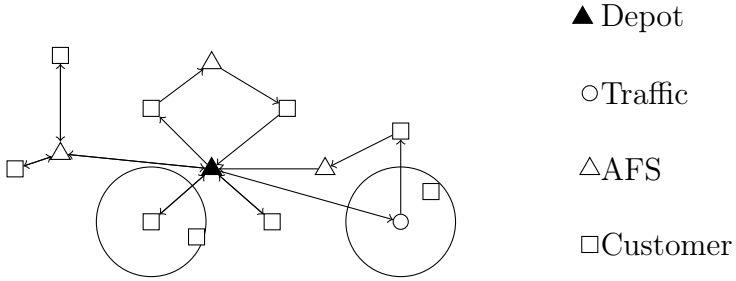


Figure 4.2: ECCTP solution example

Our contribution We introduce a new VRP variant called Electric Capacitated Covering Tour Problem (ECCTP) and present a Mixed Integer Linear Programming (MILP) model for the problem. A Biased Random-Key Genetic Algorithm (BRKGA) is also proposed for the ECCTP. Furthermore, a set of benchmark instances for the problem was

created based on public benchmark instances originally proposed for the VRP. Computational experiments were conducted with the proposed methods to evaluate the effectiveness of the approaches and to extract useful information for transportation decision-making.

Section 4.2 presents a literature review of papers related to this work, classified in two subsections: (1) covering tour problems and (2) green vehicle routing problems. In Section 4.3 the ECCTP is formally defined. In Section 4.4 a mathematical formulation for the problem is presented. Section 4.5 describes the BRKGA metaheuristic. Section 4.6 contains the results and analysis of the computational experiments. Finally, Section 4.7 presents the final remarks.

4.2 Literature Review

4.2.1 Covering Routing Problems

The Covering Salesman Problem (CSP), proposed by J. R. Current and D. A. Schilling [6], addresses the following question: considering a set of sites scattered in the plane that must be covered by a single vehicle tour and knowing that each site covers some of its neighbors, what is the minimum length of an enclosed vehicle tour in which all sites are covered? More formally, given an undirected graph, the CSP objective is to find the shortest Hamiltonian cycle on a subset of vertices that covers the graph. The special case where each vertex covers strictly itself is the Traveling Salesman Problem (TSP) [2], which follows that CSP is also NP-hard.

Some solution methodologies were proposed in the literature for the CSP. J. R. Current and D. A. Schilling [6], for example, developed a two-step heuristic to solve the CSP: the first step solves a set cover problem; the second step solves the TSP on the vertices determined by the first step. More than two decades later [21] revisited the problem by proposing a heuristic for the CSP embedded within an Integer Linear Programming (ILP) framework. First they employ constructive heuristics to find good initial solutions and then the tour vertices are rearranged by the use of ILP techniques in an attempt to reduce its length. More recently, [22] give a polynomial size formulation and a hybrid heuristic for the CSP. Their heuristic combines ant colony optimization and dynamic programming.

M. Gendreau et al. [11] studied the Covering Tour Problem (CTP), a variant of CSP. Let $G = (V \cup W, E)$ be an undirected graph, where $V \cup W$ is the set of vertices and E is the set of edges. Vertex v_0 is the depot, V is the set of vertices that can be visited, $T \subseteq V$ is the set of vertices that must be visited ($v_0 \in T$), and W is the set of vertices that must be covered but cannot be visited. The goal of the CTP is to determine a minimum length tour that visits a subset of vertices $S \subseteq V$ such that $T \subseteq S$ and each vertex of W is covered by some vertex in S. The authors proposed heuristics and a branch-and-cut algorithm to solve the CTP.

There are works in the literature that tackle the geometric version of CSP. In this version, a compact region of the plane containing each vertex is specified as a neighborhood set. The goal is to find a minimum length tour that starts from a depot and intercepts all neighborhood sets, thus covering all its corresponding vertices. Approximation algorithms, heuristics and methodologies based on ILP were developed for this version ([7], [14], [25], [5]).

Some works in the literature addresses variants of CTP considering multiple vehicles. [15] introduced the multi-vehicle covering tour problem (m-CTP). Given a graph $G = (V \cup W, E)$, where $V \cup W$ is the set of vertices and E the set of edges. Vertex v_0 is a

depot at which m identical vehicles start their routes, V is the set of vertices that can be visited, $T \subseteq V$ is the set of vertices that must be visited ($v_0 \in T$), and W is the set of vertices that must be covered but cannot be visited. The objective of m-CTP is to determine m routes of minimum total length satisfying following restrictions:

- there are at most m vehicle routes and each route starts and ends at vertex v_0;
- each vertex of T is visited exactly once, while each vertex of $V \backslash T$ is visited at most once;
- each vertex of W must be covered by a route in the sense that it must lie within a preset distance c of a vertex of V belonging to a route (assuming that v_0 does not cover all vertices of W);
- the number of vertices (excluding v_0) in each route is limited by a value p;
- the length of each route cannot exceed a value q.

The work of [19] investigated the location of distribution centers in the context of disaster relief. In these situations, support teams are unable to visit each affected area. Therefore, people need to go to a particular distribution center to obtain survival items, provided that these centers are not too far away. The locations of the distribution centers are defined for a car fleet departing from a depot. This application can be modeled through m-CTP. The authors proposed an Integer Linear Programming (ILP) formulation and developed a heuristic able to produce high quality solutions, even for large size instances.

Some works in the literature considered capacitated versions of covering vehicle routing problems. S. Allahyari et al. [1] proposed the Multi-Depot Covering Vehicle Routing Problem (MDCTVRP). Let $G = (N, A)$ be a directed graph, where $N = N_c \cup N_d$ is the set of vertices; $A = \{(i, j)|i, j \in N\}$ is the set of arcs; $N_c = \{1, 2, ..., n_c\}$ is the set of customers' vertices such that all $i \in N_c$ has a demand $d_i > 0$; $N_d = \{1, 2, ..., n_d\}$ is the set of depots of which the vehicles begin their routes. In the MDCTVRP is not necessary that each customer has to be visited by a vehicle, provided that they are within a maximum distance from at least one visited customer. The authors developed two formulations of Mixed Integer Linear Programming (MILP) and a Greedy Randomized Adaptive Search Procedure (GRASP) metaheuristic to solve the MDCTVRP.

In the work of [9] the multi-vehicle cumulative covering tour problem (m-CCTP) was proposed. Given a complete graph $G = (V \cup W, E)$, where $V = \{v_0, ..., v_{n+1}\}$ is the set of vertices. $V' = V \backslash \{v_0, v_{n+1}\}$ is the set of vertices that can be visited and v_0 and v_{n+1} are the two depots; W is the set of inaccessible vertices that must be covered; and E is the set of edges. Let $T \subset V'$ be the set of vertices that must be visited. The m-CCTP consists of finding a set of routes that visits all customers in T and some in $V' \backslash T$ such that the vertices in W are covered and the sum of arrival times at the visited vertices is minimized. An MILP formulation and a GRASP metaheuristic are proposed for m-CCTP. The computational experiments evaluated the performance of the methodologies for a set of instances and showed the effectiveness of the GRASP metaheuristic.

4.2.2 Green vehicle routing problems

S. Erdoğan and E. Miller-Hooks [8] proposed the G-VRP and developed two constructive heuristics: the modified heuristic of Clarke and Wright (savings) and the algorithm of

clustering based on density. The authors also proposed a heuristic custom improvement technique.

Ç. Koç and I. Karaoglan [31] proposed an approach for the G-VRP that utilizes a simulated annealing within a branch-and-cut framework. The Simulated Annealing was used to improve the initial solution and to find better primal bounds during the search. The authors evaluated their approach in terms of the number of optimal solutions obtained and the computational time required to find the best solution. The computational results have shown that 22 of 40 instances with 20 customers were solved optimally with a pre-defined time limit. Moreover, [31] proposed a new mathematical formulation with fewer variables and restrictions than previous works and without the need of network augmenting.

V. Leggieri and M. Haouari [17] proposed an improved model, achieving better results than the previous model by Ç. Koç and I. Karaoglan [31] and S. Erdoğan and E. Miller-Hooks [8]. Their formulation offers two significant advantages: compactness, due to a polynomial number of variables and restrictions, and flexibility, allowing it to be more easily adapted to other VRP variants. The main idea was to apply Reformulation-Linearization Technique (RLT), proposed by H. Sherali and W. Adams [23] and H. D. Sherali and W. P. Adams [24]. First a non-linear MILP formulation is proposed for the G-VRP, and then an equivalent MILP formulation is derived. V. Leggieri and M. Haouari [17] provide empirical evidence that the formulation and the reduction procedure were able to find optimal solutions for medium-sized instances using a general purpose solver.

A. Montoya et al. [18] propose a two-phase heuristic for the G-VRP. In the first phase, the heuristic constructs a set of routes using route-first cluster-second randomized heuristics with a procedure of optimal insertion of AFSs. In the second phase, a G-VRP solution is obtained from the resolution of set partitioning formulation over the set of routes constructed in the first phase. To test their approach, [18] execute experiments in a set of 52 literature instances. The results showed that the heuristic is competitive with other state-of-art methods.

Some very similar problems related to the G-VRP are proposed by R. G. Conrad and M. A. Figliozzi [4] and Y.-W. Wang et al. [30]. For example, they introduced the recharging vehicle routing problem, where vehicles with limited range are allowed to recharge at customer locations mid-tour. Their work addresses the problem in two versions, the capacitated (CRVRP) and the capacitated with time-windows (CRVRP-TW), for both presenting mathematical formulations and experimental results with solution bounds.

Another variant which connects routing problem to green logistics is the Pollution-Routing Problem (PRP), introduced by T. Bektas and G. Laporte [3]. The PRP has a broader and more comprehensive objective function that accounts not just for the travel distance, but also for the number of greenhouse emissions, fuel, travel times, and their costs. The results of [3] suggest that the PRP has the potential of yielding savings in total cost.

4.3 Problem Description

The ECCTP is defined next. Consider a complete graph $G(V, E)$, where V is the set of vertices and E is the set of edges. The set of vertices $V = W \cup T \cup F \cup \{v_0\}$ is composed of four disjoint sets of vertices: W contains customer vertices; T contains traffic vertices; F contains Alternative Fuel Stations (AFFs) vertices; and v_0 is the depot vertex. The

edges $(i, j) \in E$ have costs $c_{ij} \geqslant 0$. There is a set of M_{UB} homogeneous vehicles that start their routes in the depot vertex v_0 with a demand capacity Q and a battery level charged to the maximum of β units. Each customer $w \in W$ has an associated demand $d_w \geqslant 0$. For each customer $w \in W$, the set $C(w)$ is the subset of all vertices in $W \cup T$ that cover w. It is considered that $w \in C(w)$, $\forall w \in W$. The vehicles must service all customers in W. The goal of ECCTP consists of finding a set of $M \leq M_{UB}$ minimum cost routes satisfying the following constraints:

- Each route begins and ends at the vertex depot v_0;
- Each vertex $v \in (T \cup W)$ is visited at most once by only one route;
- Each vertex $f \in F$ can be visited multiple times by several routes;
- Each vertex $w \in W$ must be covered by at least one route, i.e., it must be serviced through some vertex $v \in C(w)$ that is visited by some route;
- The demand d_w of a vertex $w \in W$ is serviced exclusively by one route;
- The demand of a vertex $w \in W$ can be serviced by a route $k \in \{1, \ldots, M_{UB}\}$ only if there is a vertex v visited by route k such that $v \in C(w)$;
- The total demand serviced by a route must not exceed the capacity Q of the vehicle;
- The battery level of a vehicle decreases by c_{ij} when it travels along an edge $(i, j) \in E$. It is not allowed for a vehicle to visit an edge whose cost is higher than its current battery level. When the vehicle visits a station vertex $f \in F$, its battery level is restored to the maximum capacity of β units.

The VRP is a special case of the ECCTP where: (1) only depot and customer vertices are considered; (2) a very high battery capacity ($\beta = \infty$); (3) each customer covers only itself ($\forall w \in W : C(w) = \{w\}$). Since the VRP is a NP-hard problem [12], it follows that the ECCTP is also NP-hard.

4.4 Mathematical model

The ECCTP formulation is defined on a subgraph $G'(V, E')$ of the original graph $G(V, E)$. The subset of edges $E' = E \setminus E^{inf}$ is the original set E without some infeasible edges, where $E^{inf} = \{(v_i, v_j) \in E : c_{ij} > \beta\}$ is a set of edges that cannot be visited by any feasible solution because of the limited battery capacity.

The functions and parameters used in the MILP formulation for the ECCTP are:

- $\delta^+(i)$: is the set of directed edges in E' which *leave* a vertex $v_i \in V$;
- $\delta^-(i)$: is the set of directed edges in E' which *enter* a vertex $v_i \in V$;
- $\delta(i) = \delta^+(i) \cup \delta^-(i)$;
- $M_{set} = \{1, ..., M_{UB}\}$ is the set of available vehicles;
- $M_{LB} = \lceil \sum_{i \in W} d_i / Q \rceil$ is a lower bound for a feasible number of vehicles;

The sets of variables used in the MILP formulation for the ECCTP are given next:

- x_{ij}^k: is the number of times edge $(v_i, v_j) \in E'$ is visited by route $k \in M_{set}$.
- y_i : is the number of times vertex $v_i \in (W \cup T)$ is visited.
- w_f : is the number of times vertex $v_f \in F$ is visited.
- z_i^k : is 1 if vertex $v_i \in W$ is served by route $k \in M_{set}$; 0 otherwise.
- e_i : is the vehicle battery level at arriving in vertex $v_i \in V$;
- M : is the number of routes.

The objective function (1) minimizes the routes total cost. Constraints (2), (3) and (4) imply that the depot must belong to all routes and limit the number of routes. Constraints (5) guarantee that every vertex $v_i \in W$ has its demand serviced. Constraints (6) ensure that for each route the number of edges that enters and leaves a vertex is the same. Constraints (7) and (8) integrate edge variables with vertex degree variables of customer/traffic vertices and stations, respectively. Constraints (9) ensure that, for every vertex $v_w \in W$ whose demand is attended by vehicle $k \in M_{set}$, there at least one vertex $i \in C(w)$ visited by vehicle k, where $C(w)$ is the set of vertices that cover v_w. Constraints (10) ensure that the total attended demand by a vehicle must not exceed its capacity Q. Constraints (11) are the connectivity constraints which assure that all routes are connected to the depot. More specifically, it states that given a set of vertices S without depot vertex v_0 and a customer/traffic vertex $v_{i^*} \in S$, if vertex v_{i^*} is visited by route k then there must be at least one edge entering the set S in that route in order to v_{i^*} be connected to the depot.

Constraints (12-15) enforce that the vehicles can only visit vertices that are within the reach of their residual capacities. Constraints (12) define the minimum and maximum energy level. Constraints (13) restore the vehicle energy level when a vehicle is visiting a depot or a recharging station. Constraints (14) update the vehicle energy level when a vertex $j \in W \cup T$ is visited. Constraints (15) do not allow vehicles to recharge in stations or return to depot without the required energy. Constraints (16-22) define the domain for each set of variables.

$$\text{Min} \sum_{(i,j)\in E'} c_{ij}x_{ij} \tag{1}$$

s.t.

$$\sum_{k\in M_{set}} \sum_{j\in\delta^+(v_0)} x_{ij}^k = \sum_{k\in M_{set}} \sum_{j\in\delta^-(v_0)} x_{ji}^k = M \tag{2}$$

$$M_{LB} \leqslant M \leqslant M_{UB} \tag{3}$$

$$\sum_{j\in\delta^+(v_0)} x_{ij}^k \leqslant 1 \qquad \forall k \in M_{set} \tag{4}$$

$$\sum_{k\in M} z_i^k = 1 \qquad \forall v_i \in W \tag{5}$$

$$\sum_{j\in\delta^+(i)} x_{ij}^k = \sum_{j\in\delta^-(i)} x_{ji}^k \qquad \forall v_i \in V, \forall k \in M_{set} \tag{6}$$

$$\sum_{k\in M_{set}} \sum_{j\in\delta^+(i)} x_{ij}^k = y_i \qquad \forall v_i \in W \cup T \tag{7}$$

$$\sum_{k\in M_{set}} \sum_{j\in\delta^+(f)} x_{ij}^k = w_f \qquad \forall v_f \in F \tag{8}$$

$$\sum_{i\in C(w)} \sum_{j\in\delta^-(i)} x_{ji}^k \geqslant z_w^k \qquad \forall v_w \in W, k \in M_{set} \tag{9}$$

$$\sum_{i\in W} d_i z_i^k \leqslant Q \qquad \forall k \in M \tag{10}$$

$$\sum_{v_i\in(V\setminus S)} \sum_{v_j\in S} x_{ij}^k \geqslant \sum_{v_p\in\delta^-(v_{i^*})} x_{pi^*}^k \qquad \forall S \subseteq V \setminus \{v_0\}$$

$$\forall v_{i^*} \in S \cap (W \cup T), \forall k \in M_{set} \tag{11}$$

$$0 \leqslant e_i \leqslant \beta \qquad \forall v_i \in V \cup W \tag{12}$$

$$e_f = \beta \qquad \forall v_f \in F \cup \{v_0\} \tag{13}$$

$$e_j \leqslant e_i - c_{ij}(\sum_{k \in M_{set}} x_{ij}^k) + \beta(1 - (\sum_{k \in M_{set}} x_{ij}^k)) \qquad \forall v_j \in W \cup T, \forall v_i \in V \tag{14}$$

$$c_{ij}(\sum_{k \in M_{set}} x_{ij}^k) \leqslant e_i \qquad \forall v_j \in F \cup \{v_0\}, \forall v_i \in W \cup T \tag{15}$$

$$x_{ij}^k \in \{0,1\} \qquad \forall (v_i, v_j) \in E' : v_i \in W \cup T \text{ or } v_j \in W \cup T, \forall k \in M_{set} \tag{16}$$

$$x_{ij}^k \in \mathbb{Z}^+ \qquad \forall (v_i, v_j) \in E' : v_i, v_j \in F \cup \{v_0\}, \forall k \in M_{set} \tag{17}$$

$$y_i \in \{0,1\} \qquad \forall v_i \in W \cup T \tag{18}$$

$$w_f \in \mathbb{Z}^+ \qquad \forall v_f \in F \tag{19}$$

$$z_i^k \in \{0,1\} \qquad \forall v_i \in W, \forall k \in M_{set} \tag{20}$$

$$e_j \in \mathbb{R}^+ \qquad \forall v_j \in V \tag{21}$$

$$M \in \mathbb{Z}^+ \tag{22}$$

There are an exponential number of connectivity constraints (11); therefore, a complete enumeration of them is only possible for very small instances. The solution adopted was to consider an iterative algorithm that adds the connectivity constraints to the formulation as they are needed to progress the optimization. First the formulation is executed without these constraints and as soon as an integer solution (x^*, M) is obtained, the iterative algorithm is called. The process is shown in Algorithm 7. A connected component algorithm is executed M times for each route $k \in \{1, ..., M\}$ on undirected graphs $G_{x^*}^k(V_{x^*}^k, E_{x^*}^k)$ induced by the edges visited in each route k of the solution. From the connected components in each induced graph one can observe that excluding the one that contains the depot vertex v_0, all others make subset of vertices disconnected from the depot. Therefore, for each subset of vertices S associated to a component disconnected from v_0, a set of corresponding connectivity constraints (11) is added to the formulation, then the *solver* restarts the optimization. This process is repeated until an optimal solution that does not violate any of the connectivity constraints (11) is obtained.

Some of the advantages of our proposed formulation for ECCTP, comparing to other formulations for similar electric vehicle routing problems in literature, is: (1) the ability to handle any number of visits on each station vertex; (2) allowing solutions that have any number of consecutive visits of stations. For example, [8] proposed a formulation where dummy vertices are created for each possible visit to a station; [8, 31] and [17] proposed formulations that assume a vehicle never makes two consecutive visits in station vertices; and [30] proposed a formulation that assumed each station would be used at most once per vehicle.

4.5 BRKGA

In this section, we will describe a Biased Random Key Genetic Algorithm (BRKGA) metaheuristic developed for the ECCTP. The BRKGA [13] is a metaheuristic proposed

Algorithm 7 Iterative algorithm for ECCTP formulation.

Input: A feasible solution (x^*, M) for constraints (2-10) and (12-22).

Output: A set CC of connectivity constraints (11) violated by (x^*, M), if there is any.

1: **for** $k = 1$ to M **do**
2: Create a graph $G^k_{x^*}(V^k_{x^*}, E^k_{x^*})$ induced by the set of edges $E^k_{x^*} = \{(i,j) \in E' : \sum_{j \in \delta^+(v_i)} x^k_{ij} + \sum_{j \in \delta^-(v_i)} x^k_{ji} \geqslant 1\}$ and where $V_{x^*} = \{i \in V : \sum_{j \in \delta^+(v_i)} x^k_{ij} \geqslant 1\}$
3: Obtain the set of connected components in $G^k_{x^*}$
4: **for** each found connected component S such that $v_0 \notin S$ **do**
5: $v_{i^*} \leftarrow$ an arbitrarily chosen vertex from $S \cap (W \cup T)$
6: $CC_{new} \leftarrow$ set of connectivity constraints (11) given by (S, v_{i^*}) for each $k \in M_{set}$
7: $CC \leftarrow CC \cup CC_{new}$
8: **return** CC

to address combinatorial optimization problems. The key idea is developing a *decoder* function to map a sequence of fixed-length real-valued numbers (*random keys*) to represent a solution for the targeted problem. This section first gives an overview of this metaheuristic and then our method is described.

4.5.1 Biased Random Key Genetic Algorithm (BRKGA)

Similar to genetic algorithms, BRKGAs represent solutions by *chromosomes*. The method starts with an initial population of random *individuals*. The population is composed of two ranks: elite and non-elite individuals. The algorithm then processes the generations iteratively in four steps: (1) *crossover* of couples made by an elite and a non-elite individuals; (2) small fraction of pure random mutants are created; (3) decoding and ranking the new generation; and (4) a population selection procedure to keep the population at a fixed size.

The main characteristics that define the BRKGA, in contrast to the general class of genetic algorithm, are that in the BRKGA: (1) the solutions are represented by a fixed-length sequence of N random numbers, each in the interval $[0, 1)$; (2) elitist strategy is always adopted: best solutions pass to next generation without change; (3) the *crossover* is always between an elite and a non-elite individuals; and (4) no *mutation* method is considered: instead, some random individuals (*mutants*) are inserted in the population from time to time.

The BRKGA contains only one problem-specific procedure: a *decoder* function. This procedure starts from a vector of *random keys*, i.e. real-valued numbers in the interval of $[0, 1)$ and should output a solution for the problem that can be evaluated by a *fitness* function. Therefore, to define a BRKGA method, one just needs to define a corresponding decoding function and its problem specified parameters [27]. The parameters chosen for the BRKGA are described in Section 4.6. In the following subsection, the proposed decoder function for ECCTP is described.

4.5.2 Decoding an ECCTP solution from a vector of random keys

The decoder function for ECCTP is composed of the following steps:

1. construction of a single route with unlimited capacity and an unlimited electric battery that visit exactly once each customer and traffic vertex in $(W \cup T)$;

2. assign which customers will be served by covering each vertex in the route;

3. Split procedure: optimally split the single route with unlimited capacity into a set of routes with feasible capacity, each starting and ending in the depot vertex v_0.

4. InsertStations procedure: a dynamic programming approach to optimally insert the stations' visits in each route, making the route feasible;

In the first step, a chromosome S of length $N = |W \cup T|$ is given. The alleles in non-decreasing order keeping track of the original positions as shown in Fig. 4.3. Each position is then mapped to a unique element in $W \cup T$. The result is a single route R that visits each vertex in $W \cup T$ once as shown in Fig. 4.4. This route may have a demand greater than the limited capacity and/or may use more energy than the battery supports. These infeasibilities will be treated by the following steps.

Chromossome S	0.4	0.5	0.1	0.8	0.3	0.9
	1	2	3	4	5	6

Chromossome S'	0.1	0.3	0.4	0.5	0.8	0.9
Route R	3	5	1	2	4	6

Figure 4.3: Decoding a chromosome into a route.

Figure 4.4: A single route decoded from the chromosome.

In the second step, we assign to each vertex in R a subset of customers that will be serviced (by covering) when the route visits that vertex. Starting from the vertex r_1 at the beginning of the route $R = (r_1, r_2, \ldots, r_n)$, we assign to node r_i the following customers, in that order: (1) r_i itself if it is a customer vertex not yet assigned; (2) the customers, in lexicographic order of, that can be served by covering from r_i and which have not yet been assigned, until the total assigned demand does not exceed the vehicle capacity. If all customers coverable by r_i were assigned and there is still capacity remaining, we try to assign customers to the next vertex r_{i+1} and so on. If the demand was greater or equal to the capacity, we stop assigning customers to the vertex, reset the total demand, and restart assigning customers to the following vertices of the route. In the example shown in Fig. 4.5, vertex 3 services customers 3, 2 and 4, while vertices 5 and 1 only services themselves, and vertex 4 only serves customer 6. Vertices 2 and 6 do not service any client and will be removed in the next step. This guarantees that every customer will be serviced exactly once, but still does not assure the battery constraint.

Next, we remove from R all the vertices that have not been assigned to service any customer. Then the split procedure is applied in order to handle capacity constraints. This procedure was inspired by the split procedure originally proposed for VRP [20]. The

Customer/Transit node	Can cover		
1	1	5	3
2	2	1	4
3	3	2	4
4	4	2	3
5	5	1	2
6	5	4	3

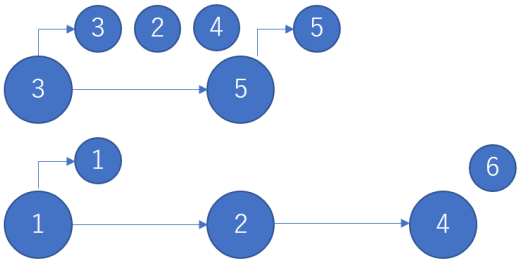


Figure 4.5: Assign customers to a vertex.

single route R is divided into several sub-routes such that each one attends the capacity constraints. This procedure is optimal for a given sequence of vertices. We create a weighted Direct Acyclic Graph (DAG) whose vertices are ordered as same as the route R. Edge (i, j) exists if and only if the sub-route $0 \to i \xrightarrow{\text{vertices between } i,j} , 0,$ (where 0 represents the depot), has a demand less or equal to the vehicle capacity Q. The weight of this edge is the total traveled distance of this route. By solving the shortest path problem in this DAG we can find the routes (edges) such that the total distance is minimum. An example is illustrated in Fig. 4.6, where the obtained shortest path cost is $2+5 = 7$. This algorithm is fast since the time complexity for the shortest path in DAGs is $O(|E|)$.

In the last step, InsertStations procedure, we solve the problem of deciding where the vehicle should proceed to recharge its battery for each route. The pseudocode of the dynamic programming is shown in Algorithm 8. One can observe that between two consecutive vertices i, j of a given route, we have two choices: go from i to a recharge station not farther than the current battery level and then visit j (line 8); or go straight from i to j (line 13). In the first case, if we have more than one feasible ways of going from a station to j, with the same remaining battery level, then the shortest way is chosen and saved in the table (lines 11 and 14). An example is illustrated in Fig. 4.7, where the chosen decision was to go from vertex 0 to vertex 3 directly and then from vertex 3 to recharge in station 3 and then arrive at vertex 0. The total route cost is 6. After InsertStations procedure is executed for each route, the obtained solution is feasible for ECCTP.

4.6 Computational Experiments

4.6.1 Benchmark of Instances

The set of instances for the ECCTP was generated from a set of CVRP instances proposed by E. Uchoa et al. [29], namely X-n110-k13. This instance is named in the format X-

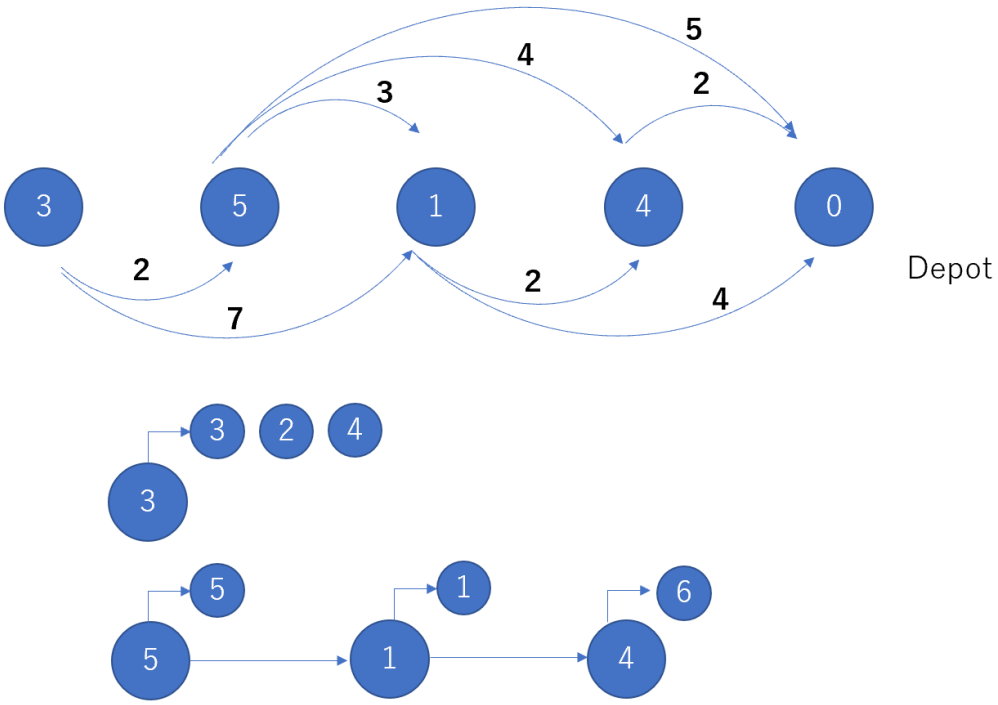
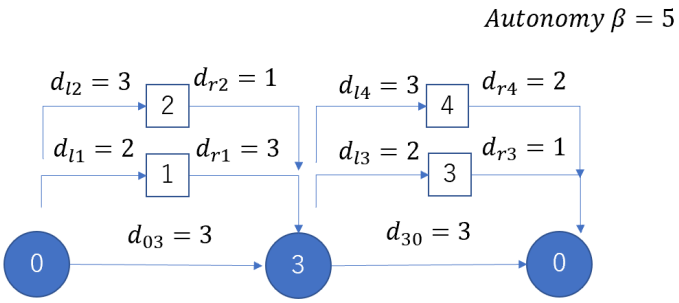


Figure 4.6: The split procedure.



	0	1	2	3	4	5
0	∞	∞	∞	∞	∞	0
3	∞	∞	$\min\{d_{03}, d_{l1} + d_{r1}\}$ $= d_{03} = 3$	∞	$d_{l2} + d_{r2}$ $= 4$	∞
0	∞	∞	∞	$4 + d_{l4} + d_{r4}$ $= 9$	$\min\{3 + d_{l3} + d_{r3}, 4 + d_{l3} + d_{r3}\}$ $= \mathbf{6}$	∞

Figure 4.7: Inserting recharging stations.

Algorithm 8 InsertStations

Input: A route R, vehicle battery range β, shortest distance between any two vertices $d(i, j)$.

Output: A feasible route.

```
Create a matrix M[1, ..., |R|][0, ..., β]
Initialize every value of M to ∞
Initialize M[1][β] := 0
for i = 1 to |R| − 1 do
    for j = 0 to β do
        if M[i][j]! = ∞ then
            dist = d(i, i + 1)
            for each station k such that d(i, k) <= j and d(k, i + 1) <= β do
                d_l = d(i, k)
                d_r = d(k, i + 1)
                if M[i][j] + d_l + d_r < M[i + 1][β − d_r] then
                    M[i + 1][β − d_r] = M[i][j] + d_l + d_r
            if j-dist>=0 then
                if M[i][j] + dist < M[i + 1][j − dist] then
                    M[i + 1][j − dist] = M[i][j] + dist
return The route corresponding to the min_j M[|R|][j]
```

nA-kB, where A represents the $n + 1$ vertices (including the depot vertex) and B the minimum number of vehicles needed to serve the demand for all vertices, calculated by solving the bin packing problem. Figure 4.8 illustrates the CVRP instance X-n110-k13 (Table 4.1).

From that X-n110-k13 instance, a total of 54 ECCTP instances were generated with varying parameters. Table 4.1 summarizes the parameters used in the generation of instances for the ECCTP. Each generated ECCTP instance considered the five following parameters:

Table 4.1: Parameters of generated instances for the ECCTP.

Parameter	Description	Values
$\|V\|$	total of vertices	$\{21, 31, 41, 51, 61, 71\}$
$\|W\|$	number of customers vertices	50% of $\|V\| - 1$
$\|T\|$	number of traffic vertices	30% of $\|V\| - 1$
$\|F\|$	number of stations vertices	20% of $\|V\| - 1$
β	vehicle battery capacity	{50%, 60%, 70%} of d_G
c	size of coverage	$\{5, 7, 11\}$
d_G - graph diameter of the ECCTP instance		

In the ECCTP instances the depot vertex is the same as the one in X-n110-k13 and the remaining vertices are the $|V| - 1$ closest vertices to the depot. The vehicle capacity Q of the ECCTP instances is the same as the X-n110-k13 instance.

The customer and station vertices were chosen randomly, with uniform distribution, from the set $V \backslash \{v_0\}$. The $|V| - 1 - |W| - |F|$ remaining vertices are traffic vertices. The customer vertices are always within a maximum distance of $\frac{\beta}{2}$ from some station vertex.

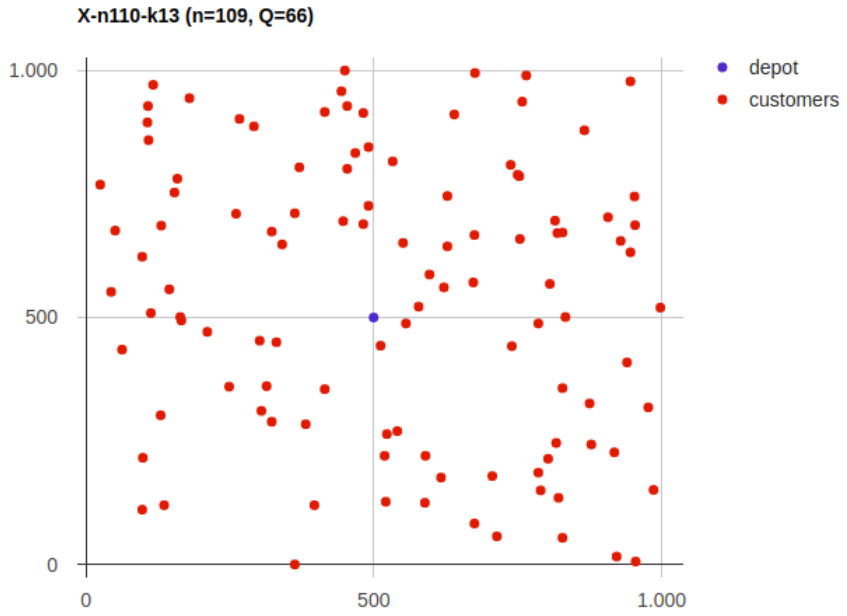


Figure 4.8: The CVRP instance X-$n110$-$k13$ used to generate the instances for ECCTP. [a]

[a]Image produced by Uchoa et al. [29] and extracted from the site http://vrp.atd-lab.inf.puc-rio.br/index.php/en/plotted-instances?data=X-n110-k13.

Besides, it is guaranteed there exists a tree spanning all stations and the depot using only edges of length β or less. Therefore, the vehicles can travel through all customer and station vertices can be served.

For each vertex $v \in (W \cup T) \setminus \{v_0\}$ a set $D(v)$, which represents the vertices covered by v, is formed by the c closest customers or traffic vertices to v. From that, the set $C(w)$, which represents the vertices that cover a customer $w \in W$, is generated by simply considering that $v \in C(w) \iff w \in D(v)$.

The instances for the ECCTP are named in the format wI-eaJ-cK, where I represents the number of customer vertices, J the vehicle battery capacity, and K the size of coverage.

4.6.2 Computational Settings

The computational experiments were executed in an Intel Xeon E3-1230 V2 3.3 GHz with 32 GB of RAM and Linux 64-bit operating system. The MILP formulation was implemented using solver Gurobi Optimizer 8.1.1, set with a 3600 s time limit for all instances. The BRKGA described in this paper was implemented using the C++ framework proposed by R. F. Toso and M. G. C. Resende [27], set with a $|W| * 10$ s time limit for each instance. Also, the parameters used by BRKGA are:

- Number of alleles per chromosome: $|W| + |T|$;
- Number of chromosomes in population: 100;
- Size of the elite set in population: 10% of the entire population;
- Number of mutants to be introduced in the population at each generation: 10% of the entire population;

- Probability that an allele is inherited from the elite parent: 80%.

4.6.3 Experiment Results

The results of the computational experiments for the benchmark of instances are reported in Tables 4.2, 4.3 and 4.4.

Table 4.2 reports the summary results of the experiments and for each $|W|$ the following data are shown:

- avg gap (%): average optimality gap;
- # opt: number of optimal solutions;
- avg time: average execution time in seconds.

The column group "MILP formulation" reports the summary results obtained by the MILP formulation, while the column group "BRKGA" reports the summary results obtained by the BRKGA metaheuristic. Note that the "avg time" column of the "MILP formulation" column group is the average total execution time of the MILP formulation, while the "avg time" column of the "BRKGA" column group represents the average execution time until the best upper bounds were obtained. The last row reports the average values of each column.

Table 4.2: Summary results of experiments.

	MILP formulation			BRKGA		
$\|W\|$	avg gap (%)	# opt	avg time	avg gap (%)	# opt	avg time
10	0.00	9	1.69	3.42	2	0.02
15	2.60	6	1433.40	6.09	1	6.26
20	13.01	1	3289.03	15.40	0	20.64
25	44.59	0	3600.00	44.63	0	24.33
30	52.69	0	3600.00	46.63	0	65.80
35	85.42	0	3600.00	59.99	0	101.65
overall	33.05	16	2587.35	29.36	3	36.45

From Table 4.2 one can observe that for the smaller instances ($|W| \leqslant 20$) the solution quality of the MILP formulation was superior, achieving a total of 16 optimal instances, while the BRKGA obtained 3. Conversely, BRKGA has a processing time much faster than the MILP formulation for instances of all sizes: in overall average, the BRKGA is approximately 70 times faster.

Regarding the hardest instances ($|W| \geqslant 30$) the performance of BRKGA is superior in both solution cost and processing time. For $|W| = 35$, for example, BRKGA obtained an average gap (%) of 59.99 in comparison to 85.42 of the MILP formulation. Moreover, for these instances, the BRKGA average time was approximately 35 times faster. This shows that the BRKGA is the most suitable method for the hardest ECCTP instances, with the advantage of also being faster. Conversely, the MILP formulation can be used to obtain good solutions for small instances and to obtain lower bounds.

Tables 4.3 and 4.4 report the full results of the experiments and for each instance, the following data are shown:

- UB: best upper bound obtained;
- LB: best lower bound obtained;
- gap (%): optimality gap $(\frac{UB-LB}{UB}) * 100$;
- time: execution time in seconds;
- M: number of vehicles used in the solution.

Column "M_{UB}" represents an upper bound of the number of vehicles required to the demand of all customers. The column group "MILP formulation" reports the results obtained by the MILP formulation proposed for ECCTP. In this column group, "time" represents the total execution time of MILP formulation and the symbol "-" means that no upper bounds were obtained. The column group "BRKGA" reports the results obtained by the BRKGA metaheuristic proposed for ECCTP. In this column group, "time" represents the execution time until the best UB was obtained. For all instances, the last row of each variation of β reports the average values of each column. Similarly, the last row of each variation of $|W|$ reports the average values of each column.

The value of parameter M_{UB} can be quite important regarding the optimization difficulty of instances by the proposed methods. If the value is too low, it may be difficult to find any feasible solution. Conversely, a big value can turn the instances unnecessarily difficult to be solved by the MILP formulation since the number of variables relies on M_{UB}. Therefore, we considered a moderate value of M_{UB} for each instance computed by the following formula: $M_{UB} = \lfloor \frac{3}{2} * M_{LB} \rfloor + 1$, where $M_{LB} = \lceil \sum_{i \in W} d_i/Q \rceil$ is a lower bound on the number of vehicles for any feasible solution.

Table 4.3 shows that parameter c is crucial for the difficulty of the instances and cost of optimal solutions. The MILP method addressed more easily instances with $c = 11$, both in solution quality and processing time, when compared to $c \in \{5, 7\}$. For example, the only optimal solution obtained for $|W| = 20$ is the instance with $c = 11$ and $\beta = 70\% * d_G$.

Parameter β also showed a substantial impact on solution costs. The decrease of solution cost when increasing the vehicle batteries capacity from $\beta = 50\% * d_G$ to $\beta = 60\% * d_G$ is substantially greater than from $\beta = 60\% * d_G$ to $\beta = 70\% * d_G$. For example, observing the optimal costs of $|W| = 10$ instances, the average cost decrease from the first increment in battery capacity was approximately 8%, while the second increment incurred in approximately 9% cost reduction. The same pattern was observed for $|W| = 15, 20, 25$. The economic interpretation of these results is that the increase in the battery capacity has diminishing returns concerning transportation costs. This kind of analysis could support the decision-making process for which vehicles model should be used given their capacities and potential cost reductions.

Table 4.4 shows the difficulty of optimizing the largest instances $|W| = 30, 35$. For many of these instances, the MILP method did not obtain any feasible solution. This probably is related to the energy constraints since this happened especially more frequently for $\beta = 50\% * d_G$ and less frequently for $\beta = 70\% * d_G$. This suggests that reformulating the problem using a stronger set of constraints to model the battery capacities may be a promising research topic. Conversely, BRKGA was shown to be an effective approach to obtain feasible solutions.

Table 4.3: Results for instances with $10 \leqslant |W| \leqslant 20$.

				MILP formulation					BRKGA			
$\|W\|$	β	c	M_{UB}	UB	LB	gap (%)	time	M	UB	gap (%)	time	M
10	50%	5	4	911	911	0.00	0.90	2	911	0.00	0.01	2
		7	4	711	711	0.00	0.80	2	715	0.56	0.02	2
		11	4	278	278	0.00	0.10	2	294	5.44	0.04	2
		overall		633.33	633.33	0.00	0.60	2.00	640.00	2.00	0.02	2.00
	60%	5	4	819	819	0.00	3.70	2	819	0.00	0.01	2
		7	4	669	669	0.00	1.60	2	688	2.76	0.02	2
		11	4	278	278	0.00	0.10	2	294	5.44	0.02	2
		overall		588.67	588.67	0.00	1.80	2.00	600.33	2.73	0.02	2.00
	70%	5	4	768	768	0.00	50.00	3	819	6.23	0.01	2
		7	4	654	654	0.00	2.90	2	688	4.94	0.03	2
		11	4	278	278	0.00	0.10	2	294	5.44	0.01	2
		overall		566.67	566.67	0.00	2.67	2.33	600.33	5.54	0.02	2.00
	overall			596.22	596.22	0.00	1.69	2.11	613.56	3.42	0.02	2.00
15	50%	5	4	1129	1085	3.90	3600.00	2	1160	6.47	44.47	2
		7	4	942	942	0.00	1070.40	2	948	0.63	7.54	2
		11	4	587	587	0.00	6.10	2	634	7.41	0.10	2
		overall		886.00	871.33	1.30	1558.83	2.00	914.00	4.84	17.37	2.00
	60%	5	4	1046	938	10.33	3600.00	2	1076	12.83	0.23	2
		7	4	894	894	0.00	707.00	2	926	3.46	0.44	3
		11	4	587	587	0.00	6.70	2	634	7.41	0.04	2
		overall		842.33	806.33	3.44	1437.90	2.00	878.67	7.90	0.24	2.33
	70%	5	4	1013	920	9.18	3600.00	2	1013	9.18	3.31	2
		7	4	833	833	0.00	283.10	2	833	0.00	0.01	2
		11	4	587	587	0.00	27.30	2	634	7.41	0.19	2
		overall		811.00	780.00	3.06	1303.47	2.00	826.67	5.53	1.17	2.00
	overall			846.44	819.22	2.60	1433.40	2.00	873.11	6.09	6.26	2.11
20	50%	5	5	1657	1321	20.28	3600.00	3	1657	20.28	8.87	3
		7	5	1285	1184	7.86	3600.00	3	1366	13.32	149.82	3
		11	5	1061	955	9.99	3600.00	3	1068	10.58	0.72	3
		overall		1334.33	1153.33	12.71	3600.00	3.00	1363.67	14.73	53.14	3.00
	60%	5	5	1607	1241	22.78	3600.00	3	1607	22.78	20.01	3
		7	5	1259	1033	17.95	3600.00	3	1263	18.21	1.38	3
		11	5	926	873	5.72	3600.00	3	962	9.25	0.02	3
		overall		1264.00	1049.00	15.48	3600.00	3.00	1277.33	16.75	7.14	3.00
	70%	5	5	1573	1229	21.87	3600.00	3	1573	21.87	4.60	3
		7	5	1259	1125	10.64	3600.00	3	1341	16.11	2.08	3
		11	5	902	902	0.00	801.30	3	962	6.24	9.99	3
		overall		1244.67	1085.33	10.84	2667.10	3.00	1292.00	14.74	5.56	3.00
	overall			1281.00	1095.89	13.01	3289.03	3.00	1311.00	15.40	20.64	3.00

Table 4.4: Results for instances with $25 \leqslant |W| \leqslant 35$.

				MILP formulation					BRKGA			
$\|W\|$	β	c	M_{UB}	UB	LB	gap (%)	time	M	UB	gap (%)	time	M
25	50%	5	5	2458	1358	44.75	3600.00	3	2462	44.84	39.10	3
		7	5	2066	1134	45.11	3600.00	3	2067	45.14	2.76	3
		11	5	1715	1042	39.24	3600.00	3	1715	39.24	0.99	3
		overall		2079.67	1178.00	43.04	3600.00	3.00	2081.33	43.07	14.28	3.00
	60%	5	5	2431	1324	45.54	3600.00	3	2461	46.20	30.93	3
		7	5	2027	1041	48.64	3600.00	3	2027	48.64	41.72	3
		11	5	1604	872	45.64	3600.00	3	1640	46.83	94.10	3
		overall		2020.67	1079.00	46.61	3600.00	3.00	2042.67	47.22	55.58	3.00
	70%	5	5	2418	1358	43.84	3600.00	3	2368	42.65	2.97	3
		7	5	2044	1094	46.48	3600.00	3	1994	45.14	1.69	3
		11	5	1552	899	42.07	3600.00	3	1577	42.99	4.73	3
		overall		2004.67	1117.00	44.13	3600.00	3.00	1979.67	43.59	3.13	3.00
	overall			2035.00	1124.67	44.59	3600.00	3.00	2034.56	44.63	24.33	3.00
30	50%	5	7	-	1610	-	3600.00	-	3183	49.42	38.95	4
		7	7	-	1456	-	3600.00	-	2586	43.70	28.77	4
		11	7	2243	1262	43.74	3600.00	4	2169	41.82	14.73	4
		overall		2243.00	1442.67	43.74	3600.00	4.00	2646.00	44.98	27.48	4.00
	60%	5	7	-	1593	-	3600.00	-	3149	49.41	93.49	4
		7	7	2585	1525	41.01	3600.00	4	2476	38.41	80.56	4
		11	7	3272	1053	67.82	3600.00	6	2180	51.70	7.56	4
		overall		2928.50	1390.33	54.41	3600.00	5.00	2601.67	46.51	60.54	4.00
	70%	5	7	-	1559	-	3600.00	-	3102	49.74	161.09	4
		7	7	3375	1229	63.59	3600.00	5	2514	51.11	39.63	4
		11	7	2104	1109	47.29	3600.00	4	1992	44.33	127.46	4
		overall		2739.50	1299.00	55.44	3600.00	4.50	2536.00	48.39	109.39	4.00
	overall			2715.80	1377.33	52.69	3600.00	4.60	2594.56	46.63	65.80	4.00
35	50%	5	7	-	1561	-	3600.00	-	4254	63.31	192.99	5
		7	7	-	1454	-	3600.00	-	3553	59.08	11.33	4
		11	7	-	1236	-	3600.00	-	3132	60.54	14.42	4
		overall		-	1417.00	-	3600.00	-	3646.33	60.97	72.91	4.33
	60%	5	7	-	1655	-	3600.00	-	4009	58.72	47.37	4
		7	7	-	1295	-	3600.00	-	3592	63.95	65.35	4
		11	7	-	1121	-	3600.00	-	2805	60.04	4.27	4
		overall		-	1357.00	-	3600.00	-	3468.67	60.90	39.00	4.00
	70%	5	7	10139	1577	84.45	3600.00	6	3651	56.81	234.11	4
		7	7	-	1341	-	3600.00	-	3376	60.28	339.96	5
		11	7	9105	1239	86.39	3600.00	6	2893	57.17	5.02	4
		overall		9622.00	1385.67	85.42	3600.00	6.00	3306.67	58.09	193.03	4.33
	overall			9622.00	1386.56	85.42	3600.00	6.00	3473.89	59.99	101.65	4.22

4.7 Conclusion

We proposed a new problem called Electric Capacitated Covering Tour Problem (ECCTP), which joins two branches in vehicle routing, the green vehicle routing and the covering tour vehicle routing.

The ECCTP was formulated as a Mixed Integer Linear Programming (MILP) model. In contrast to other formulations of electric vehicle routing problems, our model does not make any assumptions or restrictions concerning the number of times a vehicle visits the recharging station.

A Biased Random Key Genetic Algorithm (BRKGA) metaheuristic was proposed as a methodology to obtain high-quality solutions for large instances. The BRKGA decoder's main features are: (1) a split procedure, which optimally splits a single non-capacitated route into capacitated routes; (2) a dynamic programming procedure that optimally inserts stations visits in each vehicle route.

A set of benchmark instances, adapted from the CVRP literature, was proposed for the ECCTP. Computational experiments showed the effectiveness of the proposed methods. The MILP formulation solved the small instances ($|W| \leqslant 15$) in reasonable processing time (less than 1 h). For bigger instances ($|W| \geqslant 30$), the BRKGA outperformed the MILP, obtaining substantially better solutions while taking less computational time.

The computational experiments also allowed analyzing how the parameters of electric vehicle battery capacity and coverage neighborhood size could impact on the optimal solution cost and the difficulty to solve the instances. The proposed methods were found to be sensitive to the neighborhood size concerning the solution gap. Moreover, the analysis emphasized the importance of vehicle battery capacity when addressing the economic feasibility of a transportation plan.

Future studies of the ECCTP should investigate stronger mathematical formulations and also the development of local search methods as well as intensification and diversification strategies.

References

[1] Somayeh Allahyari, Majid Salari, and Daniele Vigo. A hybrid metaheuristic algorithm for the multi-depot covering tour vehicle routing problem. *EJOR*, 242(3):756–768, 2015.

[2] David L. Applegate, Robert E. Bixby, Vasek Chvatal, and William J. Cook. *The Traveling Salesman Problem: A Computational Study (Princeton Series in Applied Mathematics)*. Princeton University Press, 2007.

[3] Tolga Bektaş and Gilbert Laporte. The pollution-routing problem. *Transportation Research Part B: Methodological*, 45(8):1232 – 1250, 2011. Supply chain disruption and risk management.

[4] Ryan G. Conrad and Miguel A. Figliozzi. The recharging vehicle routing problem. In *Proceedings of the 2011 Industrial Engineering Research Conference*, 2011.

[5] Walton Pereira Coutinho, Roberto Quirino do Nascimento, Artur Alves Pessoa, and Anand Subramanian. A branch-and-bound algorithm for the close-enough traveling salesman problem. *INFORMS Journal on Computing*, 28(4):752–765, 2016.

[6] John R. Current and David A. Schilling. The covering salesman problem. *Transportation science*, 23(3):208–213, 1989.

[7] Adrian Dumitrescu and Joseph SB Mitchell. Approximation algorithms for tsp with neighborhoods in the plane. *Journal of Algorithms*, 48(1):135–159, 2003.

[8] S. Erdoğan and E. Miller-Hooks. A green vehicle routing problem. *Transportation Research Part E: Logistics and Transportation Review*, 48(1):100 – 114, 2012. Select Papers from the 19th International Symposium on Transportation and Traffic Theory.

[9] David A Flores-Garza, M Angélica Salazar-Aguilar, Sandra Ulrich Ngueveu, and Gilbert Laporte. The multi-vehicle cumulative covering tour problem. *Annals of Operations Research*, 258(2):761–780, 2017.

[10] Frances Foord. Gambia: evaluation of the mobile health care service in west kiang district. *World health statistics quarterly 1995; 48 (1): 18-22*, 1995.

[11] Michel Gendreau, Gilbert Laporte, and Frédéric Semet. The covering tour problem. *Operations Research*, 45(4):568–576, 1997.

[12] Bruce Golden. Vehicle routing problems. Technical report, M.I.T. Operations Research Center, Cambridge, 1975.

[13] José Fernando Gonçalves and Mauricio GC Resende. Biased random-key genetic algorithms for combinatorial optimization. *Journal of Heuristics*, 17(5):487–525, 2011.

[14] Damon J Gulczynski, Jeffrey W Heath, and Carter C Price. The close enough traveling salesman problem: A discussion of several heuristics. *Perspectives in Operations Research: Papers in Honor of Saul Gass' 80 th Birthday*, pages 271–283, 2006.

[15] Mondher Hachicha, M John Hodgson, Gilbert Laporte, and Frédéric Semet. Heuristics for the multi-vehicle covering tour problem. *Computers & Operations Research*, 27(1):29–42, 2000.

[16] Martine Labbé and Gilbert Laporte. Maximizing user convenience and postal service efficiency in post box location. *Belgian Journal of Operations Research, Statistics and Computer Science*, 26:21–35, 1986.

[17] Valeria Leggieri and Mohamed Haouari. A practical solution approach for the green vehicle routing problem. *Transportation Research Part E: Logistics and Transportation Review*, 104:97 – 112, 2017.

[18] Alejandro Montoya, Christelle Guéret, Jorge E. Mendoza, and Juan G. Villegas. A multi-space sampling heuristic for the green vehicle routing problem. *Transportation Research Part C: Emerging Technologies*, 70:113 – 128, 2016.

[19] Zara Naji-Azimi, Jacques Renaud, Angel Ruiz, and Majid Salari. A covering tour approach to the location of satellite distribution centers to supply humanitarian aid. *EJOR*, 222(3):596–605, 2012.

[20] Christian Prins, Nacima Labadi, and Mohamed Reghioui. Tour splitting algorithms for vehicle routing problems. *International Journal of Production Research*, 47(2):507–535, 2009.

[21] Majid Salari and Zahra Naji-Azimi. An integer programming-based local search for the covering salesman problem. *Computers & Operations Research*, 39(11):2594–2602, 2012.

[22] Majid Salari, Mohammad Reihaneh, and Mohammad S. Sabbagh. Combining ant colony optimization algorithm and dynamic programming technique for solving the covering salesman problem. *Computers & Industrial Engineering*, 83:244–251, 2015.

[23] H. Sherali and W. Adams. A hierarchy of relaxations between the continuous and convex hull representations for zero-one programming problems. *SIDMA*, 3(3):411–430, 1990.

[24] Hanif D. Sherali and Warren P. Adams. A hierarchy of relaxations and convex hull characterizations for mixed-integer zero—one programming problems. *Discrete Applied Mathematics*, 52(1):83 – 106, 1994.

[25] Robert Shuttleworth, Bruce L Golden, Susan Smith, and Edward Wasil. Advances in meter reading: Heuristic solution of the close enough traveling salesman problem over a street network. *The vehicle routing problem: Latest advances and new challenges*, pages 487–501, 2008.

[26] Witaya Swaddiwudhipong, Chaveewan Chaovakiratipong, PATCHREE NGUNTRA, Pranee Mahasakpan, Ploenjai Lerdlukanavonge, and Supawan Koonchote. Effect of a mobile unit on changes in knowledge and use of cervical cancer screening among rural thai women. *International journal of epidemiology*, 24(3):493–498, 1995.

[27] Rodrigo F Toso and Mauricio GC Resende. A c++ application programming interface for biased random-key genetic algorithms. *Optimization Methods and Software*, 30(1):81–93, 2015.

[28] Paolo Toth and Daniele Vigo. *Vehicle routing: problems, methods, and applications*. SIAM, 2014.

[29] Eduardo Uchoa, Diego Pecin, Artur Pessoa, Marcus Poggi, Thibaut Vidal, and Anand Subramanian. New benchmark instances for the capacitated vehicle routing problem. *EJOR*, 257(3):845–858, 2017.

[30] Ying-Wei Wang, Chuan-Chih Lin, and Tsung-Ju Lee. Electric vehicle tour planning. *Transportation Research Part D: Transport and Environment*, 63:121–136, 2018.

[31] Çağrı Koç and Ismail Karaoglan. The green vehicle routing problem: A heuristic based exact solution approach. *Applied Soft Computing*, 39:154 – 164, 2016.

Chapter 5

Discussion

This thesis comprises two articles presented in Chapters 2 and 3, along with one book chapter presented in Chapter 4. The objective of this research was to develop exact and heuristic algorithms to address covering routing problems. The aim was to explore algorithmic methodologies and provide effective solutions for NP-hard problems. The exact methodologies were based on Integer Linear Programming (ILP) and Mixed Integer Linear Programming (MILP) formulations, with a focus on obtaining optimal solutions and improving dual bounds. On the other hand, heuristic methodologies were employed to effectively solve large-sized instances within a short execution time, aiming to achieve high-quality solutions. Both methodologies were executed and rigorously evaluated using significant benchmarks from the literature and works presented in this thesis.

The scientific questions *"Is there an exact method more effective than the state-of-the-art for CSP?"* and *"Is there a way to obtain specific valid inequalities for the CSP?"* of Table 1.1 are addressed in the Chapter 2, in which an exact methodology for the Covering Salesman Problem (CSP) is presented. The approach involves the development of a branch-and-cut framework, which combines a novel set of inequalities called Cover Intersection (CI) inequalities, along with the usage of valid inequalities from the Generalized Traveling Salesman Problem (GTSP) proposed by Fischetti et al. [11]. The study addresses exact and heuristic separation routines for both integer and fractional solutions.

In order to assess the effectiveness of the branch-and-cut framework, a comparison was made with the ILP formulation proposed by Salari et al. [32] for the CSP. Extensive computational experiments were conducted on a benchmark of 48 instances from literature and 39 new instances. Prior to this study, only 9 optimal solutions were known among the 48 instances from literature. The branch-and-cut framework was able to obtain optimal solutions for 47 instances, with 38 of them being solved optimally for the first time. Furthermore, when it comes to the new instances, the results show substantial impact of the CI inequalities on reducing the optimality gap and obtaining the best lower bounds. Notably, the only optimal solutions achieved among the 39 instances were solely possible when using the CI inequalities.

Chapter 3 introduces a new covering routing problem denominated as the Capacitated Covering Salesman Problem (CCSP). The objective of CCSP is to determine a set of routes with the minimum cost, ensuring that each vertex is visited or covered by at least one route and the total demand serviced by any vehicle does not exceed its capacity. For the purpose of answering the scientific question *"Is it possible to develop effective methodologies for a Multi-capacitated-vehicle CSP variant?"*, two ILP formulations are proposed for the CCSP, and to tackle large size instances, a Biased Random-Key Genetic Algo-

rithm (BRKGA) and a matheuristic to intensify the search are proposed. Furthermore, the scientific question *"Is there an exact method more robust than the state-of-the-art for MDCTVRP?"* is verified by extending a CCSP ILP formulation to address the Multi-Depot Covering Tour Vehicle Routing Problem (MDCTVRP).

With the objective of evaluating the performance of the two ILP formulations, BRKGA and matheuristic, a benchmark of CCSP instances was created based on the Capacitated Vehicle Routing Problem (CVRP) instances proposed by Christofides and Eilon [5] and Uchoa et al. [43]. Additionally, the new MILP formulation proposed for the MDCTVRP was compared to the flow-based and node-based formulations presented by Allahyari et al.[1], using a benchmark of 280 instances from the literature. The computational experiments demonstrated that the new formulation consistently outperformed the best known exact methodology for the MDCTVRP from literature. Out of the 280 instances, 108 were solved optimally for the first time. Furthermore, the new formulation was able to improve all known lower bounds.

In an effort to assess the scientific question *"Is it reasonable to combine covering concept with green logistic in a new problem and develop methodologies to solve it?"*, Chapter 4 introduces the Electric Capacitated Covering Tour Problem (ECCTP). This problem presents a combination of a covering routing problem and the Green Vehicle Routing Problem (G-VRP), in which the customers demands are serviced by covering using electric vehicles. The goal of ECCTP is to find a set of routes with the minimum cost, ensuring that each costumer is visited or covered, the total demand serviced by any electric vehicle does not exceed its capacity and the battery level of an electric vehicle during a route does not run out. Two methodologies are proposed to solve the ECCTP: a MILP formulation and a BRKGA algorithm.

For the purpose of measuring the performance for both methodologies, a benchmark of 54 instances for the ECCTP was proposed. These instances were derived from the CVRP instances created by Uchoa et al. [43]. In the case of small instances, the results revealed that the MILP formulation achieved 16 optimal solutions, while BRKGA obtained only 3. Regarding the more challenging instances, BRKGA outperformed MILP in terms of optimality gaps. The average gap achieved by BRKGA for the hardest instances was approximately 59.99%, while MILP obtained a corresponding value of 85.42%. Hence, these results confirm that BRKGA is the most appropriate method for the hardest ECCTP instances.

Chapter 6

Conclusions

Routing problems have captured the interest of many researchers due to their numerous real-world applications. When it comes to real-world applications, novel variations of routing problems have been introduced in the literature, encompassing restrictions such as time constraints, green logistics, resources availability and customers accessibility. Among these, the latter gives rise to the covering routing problems, designed to address real-world scenarios where there are hard-to-reach regions containing customers that need to be serviced by coverage. In this thesis, we study and present computational methodologies for four distinct covering routing problems: (i) the Covering Salesman Problem (CSP); (ii) the Capacitated Covering Salesman Problem (CCSP); (iii) Multi-Depot Covering Tour Vehicle Routing Problem (MDCTVRP); and (iv) the Electric Capacitated Covering Tour Problem (ECCTP).

Different computational methodologies were presented in this thesis in order to address the covering routing problems under study. The exact approaches were based on Integer Linear Programming (ILP) and Mixed Integer Linear Programming (MILP) formulations. Furthermore, valid inequalities and branch-and-cut algorithms were developed, encompassing both exact and heuristic separation routines for different inequalities. On the other hand, metaheuristics based on the Biased Random-Key Genetic Algorithm (BRKGA) were investigated with the purpose of tackling large size instances. Moreover, a hybrid methodology was employed to improve heuristic solutions obtained through BRKGA. This methodology was based on an ILP formulation with covering and packing constraints, where an intensification in the search space was made from the solutions of BRKGA.

The computational methodologies and results presented in this thesis show several contributions to the advancement of studies on covering routing problems and their state of the art. Various instances in the literature were solved optimally for the first time by employing the developed methodologies. Besides, the state-of-the-art methodologies for different covering routing problems were outperformed. Through the results obtained in this thesis, we believe that the proposed computational methodologies can be applied to tackle several real applications, including collect and delivery routing systems with bimodal distribution, routing of health care services in developing countries and meter reading automation by using trucks equipped with sensors capable of read data remotely.

For the CSP, future works should investigate the integration of heuristics methodologies into the branch-and-cut framework proposed in this thesis. This approach aims to develop hybrid methodologies for the CSP, such as matheuristics. Thus, the upper and lower bounds for the CSP can be improved by using hybrid methodologies. Furthermore,

the new family of valid inequalities should be assessed in new CSP variants, for example, CSP with multiple vehicles, capacity constraints, time constraints, green vehicles, uncertainty on the covering neighborhood, and other generalizations of the CSP which better approximate practical routing problems.

Regarding the CCSP and MDCTVRP, investigations into valid inequalities and branch-and-cut algorithms can be significantly effective. The proposed valid inequalities and the branch-and-cut framework for the CSP can be extended in order to tackle both of these problems. Furthermore, it is essential to consider novel variants of routing problems, such as multi-objective vehicle routing problem. In this variant, the covering range of the problem is considered within the objective function.

In relation to the ECCTP, new ILP and MILP formulations, valid inequalities and branch-and-cut algorithms should be investigated in order to obtain new optimal solutions and improve the lower bounds. On the other hand, heuristic methodologies should be explored with the goal of obtaining high-quality solutions for the ECCTP. This can be achieved by the development of new metaheuristics, local search procedures and hybrid methodologies, such as matheuristics in which apply intensification and diversification strategies in the search space of the problem.

References

[1] Somayeh Allahyari, Majid Salari, and Daniele Vigo. A hybrid metaheuristic algorithm for the multi-depot covering tour vehicle routing problem. *European Journal of Operational Research*, 242(3):756–768, 2015.

[2] David L. Applegate, Robert E. Bixby, Vasek Chvatal, and William J. Cook. *The Traveling Salesman Problem: A Computational Study (Princeton Series in Applied Mathematics)*. Princeton University Press, Princeton, NJ, USA, 2007.

[3] Dimitris Bertsimas and John N Tsitsiklis. *Introduction to linear optimization*, volume 6. Athena Scientific Belmont, MA, 1997.

[4] Bülent Çatay and İhsan Sadati. An improved matheuristic for solving the electric vehicle routing problem with time windows and synchronized mobile charging/battery swapping. *Computers and Operations Research*, page 106310, 2023.

[5] Nicos Christofides and Samuel Eilon. An algorithm for the vehicle-dispatching problem. *Journal of the Operational Research Society*, 20(3):309–318, 1969.

[6] John R Current and David A Schilling. The covering salesman problem. *Transportation science*, 23(3):208–213, 1989.

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

[8] Rodrigo De la Fuente, Maichel M Aguayo, and Carlos Contreras-Bolton. An optimization-based approach for an integrated forest fire monitoring system with multiple technologies and surveillance drones. *European Journal of Operational Research*, 2023.

[9] Karl F Doerner and Richard F Hartl. Health care logistics, emergency preparedness, and disaster relief: new challenges for routing problems with a focus on the austrian situation. *The vehicle routing problem: latest advances and new challenges*, pages 527–550, 2008.

[10] João Luís dos Santos, João Francisco Gonçalves Antunes, Júlio César Dalla Mora Esquerdo, Alexandre Camargo Coutinho, and Lucas Porto Maziero. Otimização de um banco de dados geográficos utilizando postgis. In: SIMPÓSIO BRASILEIRO DE SENSORIAMENTO REMOTO, 18., 2017, Santos. Anais ..., 2017.

[11] Matteo Fischetti, Juan José Salazar González, and Paolo Toth. A branch-and-cut algorithm for the symmetric generalized traveling salesman problem. *Operations Research*, 45(3):378–394, 1997.

[12] Michel Gendreau, Gilbert Laporte, and Frédéric Semet. The covering tour problem. *Operations Research*, 45(4):568–576, 1997.

[13] Fred Glover. Tabu search—part i. *ORSA Journal on computing*, 1(3):190–206, 1989.

[14] José Fernando Gonçalves and Mauricio GC Resende. Biased random-key genetic algorithms for combinatorial optimization. *Journal of Heuristics*, 17(5):487–525, 2011.

[15] Mondher Hachicha, M John Hodgson, Gilbert Laporte, and Frédéric Semet. Heuristics for the multi-vehicle covering tour problem. *Computers & Operations Research*, 27(1):29–42, 2000.

[16] M John Hodgson, Gilbert Laporte, and Frederic Semet. A covering tour model for planning mobile health care facilities in suhumdistrict, ghama. *Journal of Regional Science*, 38(4):621–638, 1998.

[17] John H Holland. *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence*. MIT press, 1992.

[18] Lei Jiao, Zhihong Peng, Lele Xi, Miao Guo, Shuxin Ding, and Yue Wei. A multi-stage heuristic algorithm based on task grouping for vehicle routing problem with energy constraint in disasters. *Expert Systems with Applications*, 212:118740, 2023.

[19] Scott Kirkpatrick, C Daniel Gelatt Jr, and Mario P Vecchi. Optimization by simulated annealing. *science*, 220(4598):671–680, 1983.

[20] Nikolaos A Kyriakakis, Stylianos Aronis, Magdalene Marinaki, and Yannis Marinakis. A grasp/vnd algorithm for the energy minimizing drone routing problem with pickups and deliveries. *Computers and Industrial Engineering*, page 109340, 2023.

[21] David Lai, Yijun Li, Emrah Demir, Nico Dellaert, and Tom Van Woensel. Self-adaptive randomized constructive heuristics for the multi-item capacitated lot sizing problem. *Computers and Operations Research*, page 105928, 2022.

[22] Yuchen Luo, Bruce Golden, and Rui Zhang. The hot spot coverage patrol problem: Formulations and solution approaches. *INFORMS Journal on Computing*, 2023.

[23] Yong Ma, Zhengwen He, Nengmin Wang, and Erik Demeulemeester. Tabu search for proactive project scheduling problem with flexible resources. *Computers and Operations Research*, 153:106185, 2023.

[24] Shruti Maheshwari, Pramod Kumar Jain, and Ketan Kotecha. Route optimization of mobile medical unit with reinforcement learning. *Sustainability*, 15(5):3937, 2023.

[25] Lucas Porto Maziero, Rafael Kendy Arakaki, Matheus Diógenes Andrade, and Fábio Luiz Usberti. The electric capacitated covering tour problem. SBPO, Anais do LI Simpósio Brasileiro de Pesquisa Operacional, 2019.

[26] Lucas Porto Maziero, Fábio Luiz Usberti, and Celso Cavellucci. Um algoritmo branch-and-cut para o problema do ciclo dominante. pages 2036–2047. SBPO, Anais do XLVIII Simpósio Brasileiro de Pesquisa Operacional, 2016.

[27] Zara Naji-Azimi, Jacques Renaud, Angel Ruiz, and Majid Salari. A covering tour approach to the location of satellite distribution centers to supply humanitarian aid. *European Journal of Operational Research*, 222(3):596–605, 2012.

[28] Rojin Nekoueian, Tom Servranckx, and Mario Vanhoucke. Constructive heuristics for selecting and scheduling alternative subgraphs in resource-constrained projects. *Computers and Industrial Engineering*, page 109399, 2023.

[29] Christos H Papadimitriou and Kenneth Steiglitz. *Combinatorial Optimization: Algorithms and Complexity*. Courier Corporation, 1998.

[30] Mauricio GC Resende. Greedy randomized adaptive search procedures (grasp). *Encyclopedia of Optimization*, 2:373–382, 2001.

[31] Lucas Borges Rondon, Lucas Porto Maziero, Geraldo Pereira Rocha Filho, Augusto José Venâncio Neto, Maycon Leone Maciel Peixoto, and Leandro Aparecido Villas. Protocolo baseado em geometria computacional para descoberta de cache em redes veiculares de dados nomeados. In *Anais do XXXVIII Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos*, pages 980–993. SBC, 2020.

[32] Majid Salari, Mohammad Reihaneh, and Mohammad S Sabbagh. Combining ant colony optimization algorithm and dynamic programming technique for solving the covering salesman problem. *Computers & Industrial Engineering*, 83:244–251, 2015.

[33] Rosemarie Santa González, Marilène Cherkesly, Teodor Gabriel Crainic, and Marie-Ève Rancourt. Multi-period location routing: An application to the planning of mobile clinic operations in iraq. *Computers & Operations Research*, page 106288, 2023.

[34] Weishi Shao, Zhongshi Shao, and Dechang Pi. Effective constructive heuristics for distributed no-wait flexible flow shop scheduling problem. *Computers and Operations Research*, page 105482, 2021.

[35] Samuel E Silva, Celso C Ribeiro, and Uéverton dos Santos Souza. A biased random-key genetic algorithm for the chordal completion problem. *RAIRO-Operations Research*, 57(3):1559–1578, 2023.

[36] Rui Sun, Jieyu Wu, Chenghou Jin, Yiyuan Wang, Wenbo Zhou, and Minghao Yin. An efficient local search algorithm for minimum positive influence dominating set problem. *Computers and Operations Research*, 154:106197, 2023.

[37] Nour ElHouda Tellache and Laoucine Kerbache. A genetic algorithm for scheduling open shops with conflict graphs to minimize the makespan. *Computers and Operations Research*, 156:106247, 2023.

[38] Xinliang Tian, Dantong Ouyang, Rui Sun, Huisi Zhou, and Liming Zhang. Two efficient local search algorithms for the vertex bisection minimization problem. *Information Sciences*, 609:153–171, 2022.

[39] Xinliang Tian, Dantong Ouyang, Yiyuan Wang, Huisi Zhou, Luyu Jiang, and Liming Zhang. Combinatorial optimization and local search: A case study of the discount knapsack problem. *Computers and Electrical Engineering*, 105:108551, 2023.

[40] Frank A Tillman. The multiple terminal delivery problem with probabilistic demands. *Transportation Science*, 3(3):192–204, 1969.

[41] Kevin Tole, Rashad Moqa, Jiongzhi Zheng, and Kun He. A simulated annealing approach for the circle bin packing problem with rectangular items. *Computers and Industrial Engineering*, 176:109004, 2023.

[42] Paolo Toth and Daniele Vigo. *The vehicle routing problem*. SIAM, 2002.

[43] Eduardo Uchoa, Diego Pecin, Artur Pessoa, Marcus Poggi, Thibaut Vidal, and Anand Subramanian. New benchmark instances for the capacitated vehicle routing problem. *European Journal of Operational Research*, 257(3):845–858, 2017.

[44] Laurence A Wolsey. *Integer programming*, volume 42. Wiley New York, 1998.

Appendix A

EDP Sciences copyright policy

As per the EDP Sciences copyright policy, accessible at the internet address `https://www.edpsciences.org/fr/authors/copyright-and-licensing`, authors are allowed to incorporate the contents of their published papers into their institution's web site: "Authors can make their article, published by EDP Sciences, available on their personal site, their institution's web site and Open Archive Initiative sites, provided the source of the published article is cited and the ownership of the copyright clearly mentioned.". We express our appreciation to EDP Sciences for permitting the inclusion of our article in this thesis.

Appendix B

Springer copyright policy

The Springer copyright policy, which can be accessible at the internet address `https://www.springer.com/gp/rights-permissions/obtaining-permissions/882`, grants authors the rights to use the contents of their published papers in their own thesis: "Authors have the right to reuse their article's Version of Record, in whole or in part, in their own thesis. Additionally, they may reproduce and make available their thesis, including Springer Nature content, as required by their awarding academic institution.". We thank Springer publishing company for allowing us the inclusion of our paper in this thesis.