Pontifícia Universidade Católica
do Rio de Janeiro

Ítalo Gomes Santana

Exploring the frontier of Combinatorial Optimization and Machine Learning: Applications to Vehicle Routing and Support Vector Machines

Tese de Doutorado

Thesis presented to the Programa de Pós–graduação em Informática of PUC-Rio in partial fulfillment of the requirements for the degree of Doutor em Ciências - Informática.

Advisor: Prof. Thibaut Victor Gaston Vidal

Rio de Janeiro
September 2022

Ítalo Gomes Santana

Exploring the frontier of Combinatorial Optimization and Machine Learning: Applications to Vehicle Routing and Support Vector Machines

Thesis presented to the Programa de Pós–graduação em Informática of PUC-Rio in partial fulfillment of the requirements for the degree of Doutor em Ciências - Informática. Approved by the Examination Committee:

Rio de Janeiro, September 9th, 2022

Ítalo Gomes Santana

Ítalo Gomes Santana holds a M.Sc. degree (2018) from the Institute of Computing of Universidade Federal Fluminense (UFF) and B.Sc. in Computer Science (2015) from Universidade Federal do Tocantins (UFT). He held a WRIP scholarship from Polytechnique Montréal as remote intern for four months during the doctorate. He also held a scholarship from Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq) during B.Sc. for 18 months at the University of Western Australia in Perth.

Bibliographic data

Santana, Ítalo Gomes

Exploring the frontier of Combinatorial Optimization and Machine Learning: Applications to Vehicle Routing and Support Vector Machines / Ítalo Gomes Santana; advisor: Thibaut Victor Gaston Vidal. – 2022.

93 f: il. color. ; 30 cm

Tese (doutorado) - Pontifícia Universidade Católica do Rio de Janeiro, Departamento de Informática, 2022.

Inclui bibliografia

1. Informática – Teses. 2. Otimização combinatória. 3. Aprendizado de máquina. 4. Meta-heurísticas. 5. Problema de roteamento de veículos. 6. Cortes combinatórios de Benders. I. Vidal, Thibaut Victor Gaston. II. Pontifícia Universidade Católica do Rio de Janeiro. Departamento de Informática. III. Título.

CDD: 004

Acknowledgments

I would like to thank my family: my parents, Mariza, Jaci, and Emilio, and to my girlfriend, Daniella, and her family. Thanks for all the fantastic support, comprehension, and inspiration that helped to keep me focused on my research.

I also would like to express my deep gratitude to my advisor Thibaut Vidal for all the outstanding support, lessons, guidance, car rides to university, funny moments, and friendship. I learned so much from his lectures, conversations, attitude, and example.

I will never be thankful enough to all the professors, co-authors, and friends who helped me finish this journey in so many different ways. I am especially grateful to Professors Alexandre Plastino, Isabel Rossetti, Rafael Lima, and Warley Grammacho for helping me to enroll in UFF for my Master's degree and in PUC-Rio for my Doctorate. To my co-authors, Andrea Lodi, Breno Serrano, Florian Arnold, Kenneth Sörensen, Maximilian Schiffer, and Thibaut Vidal. To all my friends in the ADAT group.

This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001, and by the support provided by Calcul Québec and Compute Canada.

Abstract

Santana, Ítalo Gomes; Vidal, Thibaut Victor Gaston (Advisor). **Exploring the frontier of Combinatorial Optimization and Machine Learning: Applications to Vehicle Routing and Support Vector Machines**. Rio de Janeiro, 2022. 93p. Tese de Doutorado – Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

Combinatorial optimization (CO) is ubiquitous in myriad practical applications (e.g., production planning, scheduling, logistics, etc.). Over the years, CO and machine learning (ML) have emerged, together, as a prospective area of research for improving decision-making processes. There is interest to harness ML algorithms to improve existing CO methods. Conversely, since many ML tasks can be reformulated as optimization problems, there is broad interest in leveraging state-of-the-art CO methods for them. In this thesis, we conduct three studies that connect CO and ML around two important applications: the capacitated vehicle routing problem (CVRP) and support vector machines with hard-margin loss (SVM-HML). Our first study proposes a strategy to explore high-order local-search neighborhoods by pattern mining into two state-of-the-art metaheuristics for the CVRP. In a second study, also in the context of the CVRP, we exploit relatedness criteria for customer nodes using predictions from graph neural networks. We show that relatedness measures can be exploited to steer local search and extend crossover operators in a state-of-the-art genetic algorithm. Lastly, in a third study, we propose an efficient mixed-integer programming approach based on Combinatorial Benders cuts and sampling strategies for optimally training the SVM-HML.

Keywords

Combinatorial optimization; Machine learning; Metaheuristics; Vehicle routing problem; Combinatorial Benders cut.

Resumo

Santana, Ítalo Gomes; Vidal, Thibaut Victor Gaston. **Explorando a fronteira de Otimização Combinatória e Aprendizado de Máquina: Aplicações para Roteamento de Veículos e Máquinas de Vetores de Suporte**. Rio de Janeiro, 2022. 93p. Tese de Doutorado – Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

A otimização combinatória (OC) está presente em inúmeras aplicações práticas (por exemplo, planejamento de produção, logística, etc.). Ao longo dos anos, OC e aprendizado de máquina (AM) surgiram, juntas, como uma área prospectiva de pesquisa para melhorar processos de tomada de decisão. Nesse contexto, há interesse em utilizar algoritmos de AM para melhorar métodos de OC. Por outro lado, como muitas tarefas de AM podem ser reformuladas como problemas de otimização, há um amplo interesse em utilizar métodos de OC para resolver esses problemas. Nesta tese, três estudos que conectam OC e AM em torno de duas aplicações importantes são conduzidos: o problema de roteamento de veículos capacitado (PRVC) e máquinas de vetores de suporte com perda em margem rígida (SVM-HML – do inglês *support vector machines with hard-margin loss*). No primeiro estudo, uma estratégia para explorar vizinhanças de busca local de alta ordem por mineração de padrões em duas meta-heurísticas estado da arte para o PRVC é proposta. Em um segundo estudo, também no contexto do PRVC, critérios de relacionamento para nós de clientes baseados em saídas de redes neurais em grafos são explorados. Com base nessas saídas, medidas de relação podem ser exploradas para orientar a busca local e estender operadores de cruzamento em um algoritmo genético estado da arte. Por fim, no terceiro estudo, uma abordagem eficiente de programação inteira mista baseada em cortes combinatórios de Benders e estratégias de amostragem são utilizadas para treinar modelos de SVM-HML de maneira mais eficiente.

Palavras-chave

Otimização combinatória; Aprendizado de máquina; Meta-heurísticas; Problema de roteamento de veículos; Cortes combinatórios de Benders.

Table of contents

1 Introduction 11

2 PILS: Exploring high-order neighborhoods by pattern mining and injection 14
2.1 Introduction 14
2.2 Literature review 16
2.3 Pattern Injection Local Search 19
2.3.1 Pattern extraction 20
2.3.2 Pattern injection 21
2.3.3 Design choices and parameters 24
2.4 Application of PILS in two CVRP metaheuristics 25
2.5 Computational Experiments 26
2.5.1 Impact of PILS parameters 27
2.5.2 Pattern frequency versus solution quality 29
2.5.3 Performance impact of PILS 31
2.6 Conclusions 34

3 Neural Networks for Local Search and Crossover in Vehicle Routing: A Possible Overkill? 36
3.1 Introduction 36
3.2 Methodology 39
3.2.1 Hybrid Genetic Search 40
3.2.2 Local Search using Relatedness Measures 40
3.2.3 Crossover using Relatedness Measures 43
3.3 Experimental Analyses 45
3.3.1 Computational Environment 45
3.3.2 Benchmark Instances 46
3.3.3 Parametrization and Training of the GNN 46
3.3.4 Calibration of the Local Search 47
3.3.5 Experimental Results – Set XML 48
3.3.6 Experimental Results – Set X 50
3.4 Conclusions 52

4 Support Vector Machines with the Hard-Margin Loss: Optimal Training via Combinatorial Benders' Cuts 53
4.1 Introduction 53
4.2 Related Literature 55
4.3 Methodology 57
4.3.1 Descriptive formulations 57
4.3.2 Combinatorial Benders' cuts 58
4.3.3 Combinatorial Benders' cuts and the SVM-HML 60
4.3.4 General Solution Approach 61
4.4 Computational Experiments 63
4.4.1 Data and Experimental Setup 64

4.4.2 Performance of CB-SVM-HML 65
4.4.3 Impact of the time dedicated to cut separation 67
4.5 Conclusion 69

5 Conclusions 71

Bibliography 73

A Detailed Experimental Results – Set XML 86

B Detailed Experimental Results – Set X 90

C Publication Status 93

List of figures

Figure 2.1 Most frequent patterns for a CVRP instance with 560 delivery locations 21
Figure 2.2 Illustration of the pattern injection process 22
Figure 2.3 Pattern frequency and appearance in optimal solutions 30
Figure 2.4 Pattern frequency and quality of the associated solutions 30
Figure 2.5 Proportion of applied PILS moves per "move order" 33
Figure 2.6 Proportion of applied PILS moves per pattern size 33
Figure 2.7 Proportion of applied PILS moves per number of involved routes 33
Figure 2.8 Proportion of CPU time spent in different components of the algorithms 33
Figure 2.9 Convergence of HGS, HGS-PILS, KGLS and KGLS-PILS solutions over time 34

Figure 3.1 Sets of related customers according to ϕ_{D} and ϕ_{N}, on instance X-n247-k50 42
Figure 3.2 Illustration of the ordered crossover (OX) 43
Figure 3.3 Illustration of the crossover using relatedness measures (NOX and DOX) 44
Figure 3.4 Convergence plots for all HGS variants on set XML (upper graph = complete run, lower graph = last 1.5 seconds) 49
Figure 3.5 Convergence plots for all HGS variants on set X 51

Figure 4.1 Hinge loss function 54
Figure 4.2 Hard-margin loss function 54
Figure 4.3 Optimality gaps on Type-A and Type-B instances as a function of the number of samples 67
Figure 4.4 Optimality gaps achieved by CB-SVM-HML with different values of T_B and T_S 68

Figure A.1 Convergence plots on set XML for all GNN configurations (for each configuration, left graph = complete run, right graph = last 1.5 seconds of the run time) 87
Figure A.2 Boxplots of the percentage error gaps achieved by the methods, for different subsets of the XML instances, using $\Gamma = 15$ and the ORIGINAL configuration of the GNN 89

Figure B.1 Convergence plots on set X for all GNN configurations (for each configuration, left graph = complete run, right graph = last 25 % of the run time) 91
Figure B.2 Boxplots of the percentage error gaps achieved by the methods, for different subsets of the X instances, using $\Gamma = 15$ and the ORIGINAL configuration of the GNN 92

List of tables

Table 2.1 Parameter sensitivity analysis 28
Table 2.2 Impact of PILS on solution quality for HGS and KGLS different subsets of instances 31
Table 2.3 Impact of PILS at different stages of the search – Statistical significance 34

Table 3.1 GNN configurations 47
Table 3.2 Impact of Γ on solution quality and CPU time 48
Table 3.3 Results of all methods and GNN configurations for the instances of set XML 49
Table 3.4 Results of all methods and GNN configurations for the instances of set X 50

Table 4.1 Characteristics of the UCI datasets 64
Table 4.2 Performance comparison on the UCI instances 65
Table 4.3 Performance comparison on the Type-A instances – grouped by number of samples n 65
Table 4.4 Performance comparison on the Type-B instances – grouped by number of samples n 66
Table 4.5 Performance comparison on the Type-A instances – grouped by number of features m 66
Table 4.6 Performance comparison on the Type-B instances – grouped by number of features m 66
Table 4.7 Significance results of paired-samples Wilcoxon tests 69

Table A.1 Results for set XML for all values of $\Gamma \in \{15, 20, 30, 50\}$ 88

Table B.1 Results for set X for all values of $\Gamma \in \{15, 20, 30, 50\}$ 90

1 Introduction

Combinatorial optimization (CO) is a well-developed area that includes a long list of real-world problems. To cite a few examples, CO applications are seen in contexts of logistics, sales and manufacturing performance, workload placement, staff rostering, and portfolio management, among many others. Generally, a CO problem requires finding a combination of decisions that optimize a given objective subject to a set of constraints. Despite their broad presence in real-life scenarios, CO problems are usually NP-hard, and solving them to optimality typically takes a time that grows exponentially with the number of decisions to take.

To deal with these problems effectively, many studies focus on improving existing algorithms by reducing solutions' search space or making the search process faster. Recently, there has been increasing interest in employing machine learning (ML) techniques to improve these algorithms even further. As a result, CO and ML areas have emerged, together, as a prospective area of research since ML training can also benefit from CO techniques.

From the viewpoint of ML in CO, one can use ML techniques to mitigate limitations or strengthen components of CO solvers by learning from past decisions. In light of this, an emerging line of research embraces neural networks (NN) as (sub)routines to solve CO problems. For example, [146] presented a graph NN that predicts the probability of each edge participating in a high-quality solution. All edges whose probabilities are below a threshold are then used to prune the search space of solutions in applications for the capacitated vehicle routing problem (CVRP) and the traveling salesperson problem (TSP). For a complete review of studies in this viewpoint, we refer the interested reader to the surveys of [1, 154, 155, 157].

Conversely, from the viewpoint of CO in ML, there has been broad interest in leveraging CO methods to enhance ML training tasks, which are often cast as optimization problems. These training tasks usually optimize a given objective (e.g., minimize training error) constrained over some restrictions of the learning problem [158]. On this subject, [98] and [108] presented various mathematical optimization approaches for training support vector machines with hard-margin loss (SVM-HML), an ML model that requires a solution to an NP-hard problem. The presence of CO problems in training ML models is widespread in many surveys in the literature, such as [95, 159, 160, 158].

In this thesis, we conduct three studies that explore these viewpoints. From the viewpoint of ML in CO, we present two studies that employ ML techniques to enhance the overall performance of state-of-the-art metaheuristics for the CVRP. From the other viewpoint, we employ efficient solution techniques from the mathematical optimization domain to improve the training performance of SVM-HML. Each study, presented in an individual chapter, is summarized as follows. The final remarks of this thesis are presented in the last chapter (Chapter 5).

– In the first study, we propose a strategy to explore high-order local-search neighborhoods by pattern mining, described in Chapter 2. We demonstrate that this strategy complements classical search procedures by identifying useful high-order moves, illustrated in two state-of-the-art metaheuristics for the capacitated vehicle routing problem (CVRP). Our approach relies on an effective recursive algorithm that optimally rebuilds solutions from a set of route fragments and a pattern. At the evaluation step, we extensively analyze our proposal's ability to find new moves and contribute to the search performance.
– In the second study, also developed in the CVRP domain, we introduce the concept of relatedness criteria for customer nodes and their exploitation of local search and crossover operators of a state-of-the-art metaheuristic. We demonstrate that this relatedness based on predictions of graph neural networks (GNNs) or simply distance-based measures can enhance state-of-the-art results. Moreover, we circumvent a limitation in these GNNs, which are designed to predict outputs only for fixed-size graphs, to perform on larger instances without retraining the GNN model. We instead decompose the original instance into a sequence of fixed-size subproblems and aggregate the resulting outputs. This technique has the benefit of only requiring a single trained GNN. The complete content of this study is presented in Chapter 3.
– Finally, the third study comprises an efficient mixed-integer programming approach based on Combinatorial Benders (CB) cuts and sampling strategies for optimally training the support vector machines with hard-margin loss (SVM-HML), an SVM variant that requires the solution to an NP-hard problem. In this study, we demonstrate that these cuts, generated on a variant of the linear separator problem that fits in the CB framework, lead to useful inequalities for SVM-HML. Furthermore, a simple sampling strategy is utilized to promote a better diversification in the

pool of CB cuts by solving smaller subproblems. In the reported results, we observed that our methodological approach significantly increased the ability to reach optimal solutions in less computational time and smaller optimality gaps in the same computational time than baseline results. We detail this study in Chapter 4.

2
PILS: Exploring high-order neighborhoods by pattern mining and injection

2.1
Introduction

Recent research has demonstrated that discovering the structural properties of high-quality solutions, i.e., what differentiates high-quality from low-quality solutions, can be instrumental in developing state-of-the-art heuristics for hard combinatorial optimization problems [5, 75]. In this study, we investigate whether *pattern mining*, i.e., the discovery of frequently used patterns or structures in high-quality solutions, can similarly improve the performance of a heuristic optimization algorithm.

Pattern mining is a well-established technique to detect correlations and substructures in datasets. It is traditionally used in market-data analysis to identify sets of products that are frequently acquired together, and many other applications exist, such as DNA analysis and fraud detection [2]. In all of these cases, the extraction of patterns reveals insightful associations and can guide strategic decisions.

We focus this study on the *capacitated vehicle routing problem* (CVRP). This problem belongs to the class of optimization problems known as *vehicle routing problems*. These problems seek to find least-cost delivery routes to visit a geographically dispersed customer set, therefore generalizing the classical traveling salesman problem (TSP) with multiple vehicles and other side constraints [49, 55]. Almost all vehicle routing problems are NP-hard as an extension of the classical TSP, but most are notoriously more difficult to solve in practice. Despite 60 years of research and published research papers numbering in the thousands, the best exact algorithms for vehicle routing problems remain unable to consistently solve instances with around 300 customers in a reasonable amount of computation time [56, 38]. In contrast, real-life TSP instances of a similar size generally take a few seconds to be solved to proven optimality [76]. Due to both their computational difficulty and practical interest, vehicle routing problems have therefore emerged as one of the most important benchmarks for metaheuristics, designed to produce high-quality approximate solutions in a controlled time.

It is well known that high-quality solutions of a vehicle routing problem tend to be structurally close to the global optima, with which they share a large number of common edges [12]. Moreover, during a typical search, several sequences of consecutive visits regularly re-appear in high-quality solutions. A few studies have attempted to exploit such *patterns* heuristically, either by guiding the search towards frequently occurring customer sequences or by building new initial solutions from them as a starting point for the local search operators. However, state-of-the-art heuristics for vehicle routing problems generally rely on efficient local search operators to a far greater extent than on iterative solution construction procedures. We therefore posit that a careful adaptation of the local search components using a set of high-quality patterns (i.e., customer sequences that frequently occur in high-quality solutions) could be a promising avenue in the design of high-quality of heuristics for vehicle routing problems. This research path, however, remains mostly unexplored.

To fill this gap, we introduce a technique to effectively exploit discovered patterns in a local search heuristic. We have called this technique *pattern injection local search* (PILS). PILS is a generic move generator that efficiently finds high-order moves (i.e., moves in which more than two visits are affected simultaneously) based on patterns frequently occurring in high-quality solutions.

In a nutshell, PILS consists of two algorithmic steps: pattern collection and pattern injection. *Pattern collection* is the process of collecting patterns (i.e., sequences of consecutive visits) that frequently occur in high-quality solutions. Then, a subset of the most frequent patterns can be introduced in an incumbent solution in a three-step process called *pattern injection.* (1) Incompatible edges (i.e., edges adjacent to nodes in the pattern, but not occurring in the pattern itself) are disconnected. (2) The edges defined by the pattern are reconnected. This yields a set of disconnected route fragments, which are (3) optimally reconnected. In PILS, a pattern injection move is only accepted if it improves the incumbent solution.

The ability of PILS to find high-order pattern injection moves can be easily used to complement other local searches and is independent of the metaheuristic paradigm used (e.g., population- or trajectory-based methods). We demonstrate this generality by applying PILS in the framework of two state-of-the-art metaheuristics for the CVRP: the hybrid genetic search of [52] and the guided local search of [5].

In summary, the contributions of this study are threefold:

1) We introduce a new optimization technique called *pattern injection*

local search (PILS) that can be used to generate high-order moves by introducing patterns frequently discovered in high-quality solutions in an incumbent solution. We discuss the major design decisions and implementation strategies related to this new technique. To the best of our knowledge, this study presents the first attempt to use pattern mining to generate and enumerate specialized large neighborhoods.

2) As part of the PILS approach, we describe a simple algorithm to optimally reconnect the route fragments that occur during the pattern injection phase.
3) Finally, we conduct extensive experiments to measure the effectiveness of PILS using two state-of-the-art metaheuristics for the CVRP. We also evaluate how pattern frequency and quality are correlated, and measure the sensitivity of the approach to the number of selected patterns and the number of pattern-insertion attempts, thereby providing a deep analysis of the role of pattern mining in local search-based metaheuristics.

The remainder of this study is organized as follows. Section 2.2 reviews the related literature. Section 2.3 describes the PILS methodology, while Section 2.4 discusses the integration of PILS within two state-of-the-art metaheuristics for the CVRP. Section 2.5 presents our computational experiments, and Section 2.6 concludes.

2.2 Literature review

Pattern mining and metaheuristics If we (informally) define a pattern as *a set of solution characteristics*, then pattern extraction and exploitation is, at least indirectly, a founding principle of most modern metaheuristics. According to [25], the success of crossover-based genetic algorithms is largely because they promote the survival and propagation of high-quality building blocks. Similarly, path relinking algorithms [41] iteratively guide the search towards the characteristics of an elite solution, while ant colony optimization (ACO) [18] learns and reinforces promising decisions. This modus operandi comes from the fact that, for most combinatorial optimization problems of interest, high-quality solutions are structurally close to the global optimum in the solution space [12].

Other recent metaheuristics have more directly exploited pattern information, for two general purposes: (1) information exchange and cooperation between the various operators used in the metaheuristic, and (2) to generate new initial solutions. [31] rely on frequent patterns to coordinate and guide the search of several metaheuristics. Patterns are extracted from a *solution warehouse* and used to temporarily fix or prohibit edges in cooperating tabu searches and genetic algorithms. [19] and [28] have extended this methodology into an integrative cooperative search (ICS) for multi decision-attribute optimization problems, relying on structural problem decompositions and *integrations* of partial elite solutions to form complete solutions.

While studies on pattern guidance remain few and far between, contributions in which patterns are exploited to generate new initial solutions are more widespread. Adaptive memory programming (AMP – [46]) is a methodological paradigm that represents this strategy well. It generalizes most of the classical metaheuristics (tabu search, scatter search, genetic algorithms, and ACO) within a unified framework, based on the premises that all these methods "memorize solutions or characteristics of solutions generated during the search process" and "include a procedure that creates an initial solution with the information stored in memory". BoneRoute [48] successfully applies the AMP strategy to the CVRP within a population-based approach. New partial solutions are regularly built from solution components and completed heuristically. Similarly, [43] proposes a genetic algorithm, in which new solutions are generated via a multi-parent crossover or a construction procedure combining elite patterns. Set-covering-based *matheuristics* [35, 45] also regularly combine solution elements (e.g., routes, bins, clusters) into complete solutions using integer programming solvers. Several studies have also focused on identifying frequent sequences or visits to construct new initial solutions for the TSP, leading to methods known as backbone search [27, 44], tabu search with vocabulary building [23], and fixed set search [26].

It is tempting to combine multiple promising solution fragments into new solutions. However, search methods based on this principle face a major problem: even though several *individual* decisions may be found in a large number of high-quality solutions, *combinations* of these decisions may not. For example, even though there may exist edges that appear in a large number of high-quality solutions of a vehicle routing problem, this does not in any way guarantee that any combination of these promising edges can be used to form a high-quality feasible solution. In other words, the pattern built from promising decisions is generally not *supported* in any high-quality solution. For this reason, [11, 42] and other related studies opted to use a single — large and supported — pattern during each solution construction.

To summarize, previous studies have, either directly or indirectly, exploited pattern mining to enhance metaheuristics. Coined *parts*, *fragments* or *backbones*, these patterns capture frequent structures from elite solutions.

To the best of our knowledge, patterns have been mainly used to guide the search and drive solution construction rather than local improvement, a surprising fact given that local searches play the most critical role in most modern metaheuristics. We therefore aim to design new strategies to exploit pattern information at the local search level, through specialized, enumerable moves whose evaluation complexity remain controllable, leading to a new local search paradigm.

The capacitated vehicle routing problem Due to its importance for transportation logistics and its rich combinatorial structure, the CVRP currently stands as one of the main benchmarks for research on combinatorial optimization algorithms. In its canonical form, it is defined on a complete undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ such that $\mathcal{V} = \{0, 1, \dots, n\}$. Vertex 0 stands for a depot where a vehicle fleet is based, and each other vertex $i \in \{1, \dots, n\}$ represents a customer with demand q_i. Each edge $(i, j) \in \mathcal{E}$ represents the possibility of traveling from i to j with distance cost $c_{ij} \in \mathbb{R}^+$. The goal of the CVRP is to design vehicle routes starting and ending at the depot, in such a way that each customer is visited once, that the total demand transported on each route does not exceed the vehicle capacity Q, and that the total cost measured as the sum of the route distances is minimized [49]. Some authors assume the number of routes is fixed to a minimum number of possible routes in a solution.

The CVRP is NP-hard as a generalization of the TSP. Despite the considerable progress of mathematical programming techniques for NP-hard combinatorial optimization problems, current exact methods for the CVRP can only solve instances with a few hundred customers in a reasonable amount of time [56, 38]. Since this size is insufficient for recent applications, e.g., for e-commerce or mobility-on-demand, extensive research has been conducted on metaheuristics in an attempt to generate approximate solutions in a more controlled computational effort. Similarly, metaheuristics have regularly appeared in the pattern recognition and machine learning domains, for optimization tasks in clustering [66, 67, 78], feature selection [70, 68], and a variety of other applications [69, 72, 71, 77].

All of the classical metaheuristic paradigms have been tested on the CVRP. In the early 2000s, state-of-the-art algorithms were primarily based on tabu search and other single-trajectory metaheuristics [65]. This status-quo changed with the proposal of effective hybrid genetic searches (HGS) for this problem [36, 40, 51, 52], combining the exploration abilities of crossover- and population-based search with the improvement potential of specialized local searches to achieve a fine balance between diversification and intensification

[61, 60]. The algorithm of [51] has been holding the best-known results for the CVRP for nearly a decade. Recently, two other methods have achieved high-quality results for some instance classes: the adaptive large neighborhood search with *slack induction* of [15] and the knowledge-guided local search (KGLS) of [5]. As visible at `http://vrp.galgos.inf.puc-rio.br/index.php/en/updates`, research remains very active on the topic, and new best solutions are still regularly reported for the classical instances of [50].

2.3 Pattern Injection Local Search

Nearly all successful CVRP metaheuristics rely on some local search-based optimization component, which is iteratively applied on multiple solutions throughout the method. Given an incumbent solution s, a local search (LS) explores a neighborhood $\mathcal{N}(s)$ including all solutions reachable from s by small changes, called *moves*, with the goal of finding an improving neighbor, which is used as a new incumbent solution. This process is repeated until reaching a *local minimum* state, where no more improving neighbor exists. It can be said that LS performs a mapping of a set of initial solutions onto a set of local minima, which can be seen as a discrete analogy to gradient descent in continuous space.

Neighborhood size is typically exponential in the number of vertices or edges that are jointly modified in a move. For example, there exist $\Theta(k!\, n^k)$ solutions in the k-OPT neighborhood, obtained by deleting k edges and reconnecting the resulting solution fragments. For this reason, most CVRP metaheuristics use simple $O(n^2)$-sized neighborhoods based on single-vertex relocations (RELOCATE), pairwise exchanges (SWAP), or replacements of two edges (2-OPT, and 2-OPT*) [60]. Due to their larger computational requirements, higher-order neighborhoods are only rarely considered.

This is where the proposed technique PILS provides a meaningful alternative. Instead of exhaustively exploring high-order k-OPT neighborhoods, it relies on the information of frequent patterns to select and consider fewer — targeted — moves that insert a pattern in the incumbent solution and optimally reconstruct the remaining edges to avoid large disruptions. We now describe the two algorithmic steps involved in this process, pattern extraction and pattern injection, and then discuss important design choices when using PILS.

2.3.1 Pattern extraction

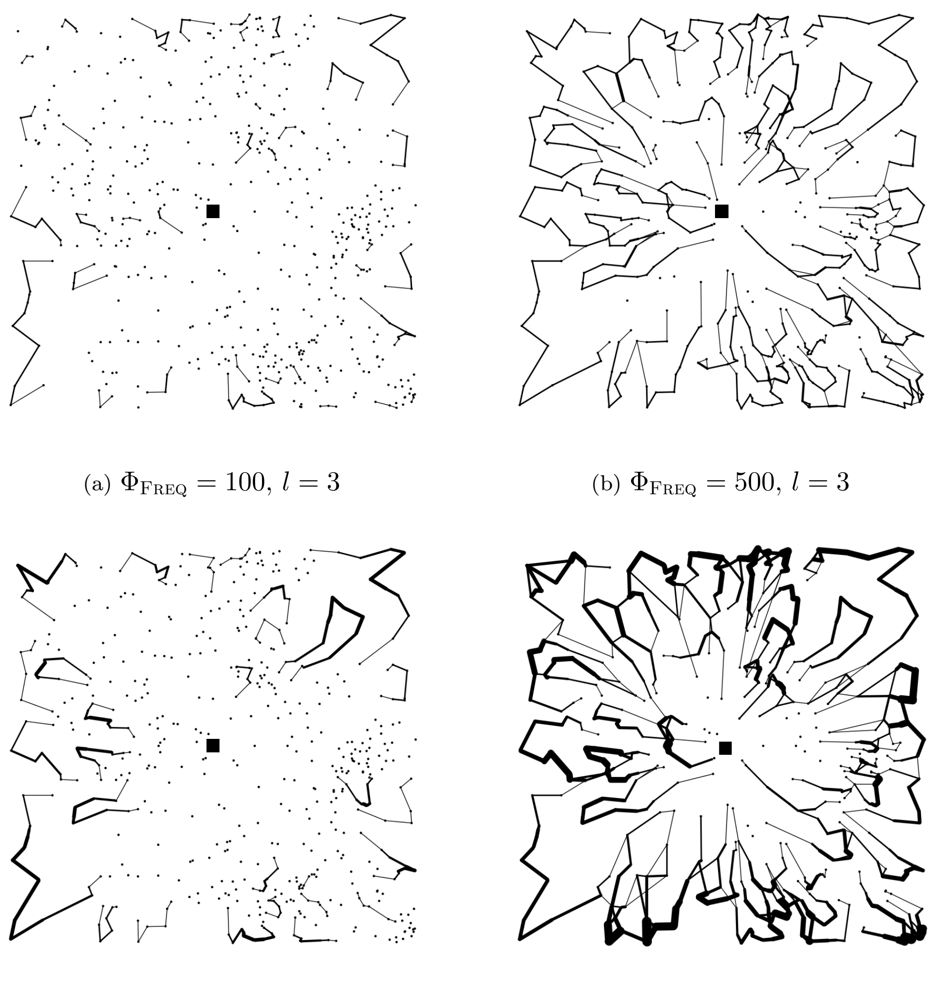
Pattern extraction, in the case of the CVRP, consists of monitoring historical solutions to obtain the frequency of patterns, i.e., the appearance of sequences of consecutive customer visits of a certain size range $\{L_{\text{MIN}}, \dots, L_{\text{MAX}}\}$. Consider a route $\sigma = (\sigma_1, \dots, \sigma_{|\sigma|})$ starting and ending at the depot, such that $\sigma_1 = 0$ and $\sigma_{|\sigma|} = 0$. In this route, each contiguous subsequence of $(\sigma_2, \dots, \sigma_{|\sigma|-1})$ represents a (supported) pattern, which could contain either all the visited customers or a part thereof. Route $(0, 1, 3, 5, 2, 6, 0)$, for example, contains two patterns of size four: $(1, 3, 5, 2)$ and $(3, 5, 2, 6)$. As we will conduct experiments on symmetric CVRP datasets, mirrored subsequences will be considered as identical, e.g., $(1, 3, 5, 2) = (2, 5, 3, 1)$. In these conditions, any route σ contains $\max\{0, |\sigma| - 1 - l\}$ patterns of size l, and any solution contains $O(n)$ patterns of a given size, such that pattern extraction can be done by simple inspection. This process is described in Algorithm 1. The resulting patterns and their associated frequency are stored in an associative array $\mathcal{A}$.

Algorithm 1: Extraction of patterns of size $l \in \{L_{\text{MIN}}, \dots, L_{\text{MAX}}\}$ from a solution s

```
1 for each pattern size $l \in \{L_{\text{MIN}}, \dots, L_{\text{MAX}}\}$ do
2     for each route $\sigma$ of $s$ do
3         for $i \in \{l + 1, \dots, |\sigma| - 1\}$ do
4             $p = (\sigma_{i-l+1}, \dots, \sigma_i)$;
5             if $p \in \mathcal{A}$ then
6                 Increment the frequency of $p$ by one unit
7             else
8                 Add new pattern in $p$ in $\mathcal{A}$
```

Depending on the size of the problem instance and the number of solutions from which patterns are extracted, the number of uniquely encountered patterns in $\mathcal{A}$ can be large. Since we aim to focus on a limited subset of frequent patterns during injections, we use an additional min-heap data structure to track the Φ_{FREQ} most frequent patterns of each given length $l \in \{L_{\text{MIN}}, \dots, L_{\text{MAX}}\}$. This data structure allows $O(1)$ access to the root to verify if the least frequent element of the heap needs to be replaced, and $O(\log(|\Phi_{\text{FREQ}}|))$ updates whenever the frequency of an element of the heap is incremented. It also allows the method to efficiently iterate over the most frequent patterns during the pattern injection phase.

Figure 2.1 illustrates the 100 and 500 most frequent patterns of size 3 and 6 for a CVRP instance with 560 delivery locations (X-n561-k42). The depot is located at the center of the figure. The thickness of the edges is proportional to their occurrence frequency in the patterns. As visible in this figure, small patterns are often contained in larger patterns. Moreover, frequent patterns usually involve customers that are more distant from the depot, since the number of relevant visit sequences for such customers tends to be smaller.


(a) $\Phi_{\text{FREQ}} = 100$, $l = 3$ (b) $\Phi_{\text{FREQ}} = 500$, $l = 3$

(c) $\Phi_{\text{FREQ}} = 100$, $l = 6$ (d) $\Phi_{\text{FREQ}} = 500$, $l = 6$

Figure 2.1: Most frequent patterns for a CVRP instance with 560 delivery locations

2.3.2 Pattern injection

During the injection phase, frequent patterns are tentatively inserted in the incumbent solution to define high-order local search moves. These moves are accepted in case of improvement. A pattern p is injected into a solution by connecting the vertices of p and rigorously removing all other interfering edges. This leads to a set of route fragments that need to be reconnected to obtain a feasible solution. Given an incumbent solution s and a subset $\mathcal{P}$ of frequent patterns, neighborhood $\mathcal{N}_{\text{PILS}}(s, \mathcal{P})$ is therefore defined as the set of all solutions obtained by:

1) selecting a pattern $p \in \mathcal{P}$ that does not currently appear in s,
2) disconnecting in s all edges adjacent to the vertices of p,
3) inserting edges to form the pattern p, and
4) optimally inserting additional edges to obtain a complete solution.

Figure 2.2 illustrates the injection process. In this example, a pattern of size six has been selected. The pattern injection step leads to a new solution in which eight edges (represented with dashed lines) have been replaced, i.e., a 8-OPT move.

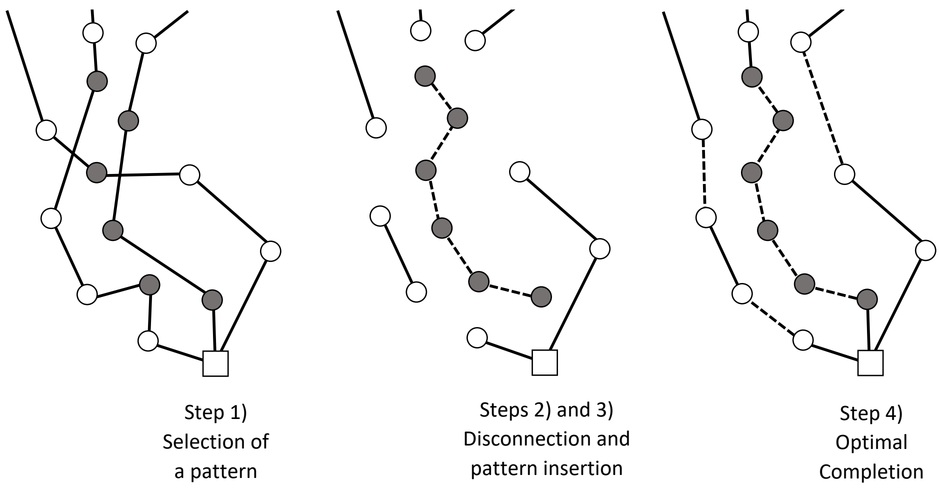


Figure 2.2: Illustration of the pattern injection process

Let $\mathcal{R}_{\text{INIT}}$ represent the set of routes containing at least one customer of p. After Step 3, the routes in $\mathcal{R}_{\text{INIT}}$ have been partitioned into fragments, which can be classified into three sets: $\mathcal{R}_{\text{BEG}}$ contains all fragments that start with a depot, $\mathcal{R}_{\text{MID}}$ contains all fragments without a depot (including pattern p), and $\mathcal{R}_{\text{END}}$ includes all fragments that end with a depot. Note that some route fragments in $\mathcal{R}_{\text{BEG}} \cup \mathcal{R}_{\text{END}}$ may contain only the depot.

During Step 4, these fragments will be optimally reconnected into a set of feasible routes via Algorithm 2. Since we work with symmetric CVRP instances, fragments can potentially be reversed during this process. An efficient algorithm for this step is critical since the number of possible recombinations grows exponentially with the number of fragments. Therefore, our algorithm relies on *pruning* techniques to detect and abort non-improving route combinations as early as possible in the recursions. $\mathcal{R}_{\text{BEST}}$ is a global variable that represents the best set of reconnected routes found so far. It is initially set to $\mathcal{R}_{\text{BEST}} = \mathcal{R}_{\text{INIT}}$. Variable $\mathcal{R}$ represents the complete routes that are currently being built. As depicted in Line 1 of Algorithm 2, the route fragments are recursively concatenated (operation $\oplus$) as long as the collective cost of the current fragments in $\mathcal{R}_{\text{BEG}}$, $\mathcal{R}_{\text{MID}}$, $\mathcal{R}_{\text{END}}$, and $\mathcal{R}$ remains smaller than that of $\mathcal{R}_{\text{BEST}}$. In this procedure, the cost of a set of routes (or route fragments) is given by the sum of the cost of its elements: $C(\mathcal{R}) = \sum_{\sigma\in\mathcal{R}} C(\sigma)$.

Algorithm 2: $\textsc{Best-Reconnect}(\mathcal{R}_{\text{BEG}}, \mathcal{R}_{\text{MID}}, \mathcal{R}_{\text{END}}, \mathcal{R})$.

```
1  if $C(\mathcal{R}_{BEG} \cup \mathcal{R}_{MID} \cup \mathcal{R}_{END} \cup \mathcal{R}) < C(\mathcal{R}_{BEST})$ then
2      if $|\mathcal{R}_{BEG}| = 0$ then
3          $\mathcal{R}_{\text{BEST}} = \mathcal{R}$
4      else
5          Select $\sigma_{\text{BEG}} \in \mathcal{R}_{\text{BEG}}$
6          for $\sigma_{MID} \in \mathcal{R}_{MID}$ do
7              $\textsc{Best-Reconnect}(\mathcal{R}_{\text{BEG}} - \{\sigma_{\text{BEG}}\} \cup \{\sigma_{\text{BEG}} \oplus \sigma_{\text{MID}}\}, \mathcal{R}_{\text{MID}} - \{\sigma_{\text{MID}}\}, \mathcal{R}_{\text{END}}, \mathcal{R})$
8              $\textsc{Best-Reconnect}(\mathcal{R}_{\text{BEG}} - \{\sigma_{\text{BEG}}\} \cup \{\sigma_{\text{BEG}} \oplus \textsc{Rev}(\sigma_{\text{MID}})\}, \mathcal{R}_{\text{MID}} - \{\sigma_{\text{MID}}\}, \mathcal{R}_{\text{END}}, \mathcal{R})$
9          if $|\mathcal{R}_{BEG}| \neq 1$ or $|\mathcal{R}_{MID}| = 0$ then
10             for $\sigma_{END} \in \mathcal{R}_{END}$ do
11                 $\textsc{Best-Reconnect}(\mathcal{R}_{\text{BEG}} - \{\sigma_{\text{BEG}}\}, \mathcal{R}_{\text{MID}}, \mathcal{R}_{\text{END}} - \{\sigma_{\text{END}}\}, \mathcal{R} \cup \{\sigma_{\text{BEG}} \oplus \sigma_{\text{END}}\})$
```

In each recursion, one single fragment σ_{BEG} of $\mathcal{R}_{\text{BEG}}$ is tentatively concatenated with each fragment $\sigma \in \mathcal{R}_{\text{MID}} \cup \mathcal{R}_{\text{END}}$ and each reversed fragment $\textsc{Rev}(\sigma)$ for $\sigma \in \mathcal{R}_{\text{MID}}$ (Line 8). Each such concatenation leads to a recursive call. During all recursive calls, we invariably have that $|\mathcal{R}_{\text{BEG}}| = |\mathcal{R}_{\text{END}}| = |\mathcal{R}_{\text{INIT}}| - |\mathcal{R}|$. This value represents the number of routes that still need to be built. Whenever only one route remains, we do not permit a connection to the last fragment of $\mathcal{R}_{\text{END}}$ unless all fragments in $\mathcal{R}_{\text{MED}}$ have been exhausted (Line 9). When this last condition occurs, the base case (Line 2) is finally attained and a possible reconnection has been obtained. At this point, due to the filtering condition, its cost is known to be smaller than the best known, and therefore $\mathcal{R}$ can be updated (Line 3).

This algorithm can be used to penalize or prohibit capacity-constraint violations in the routes. In the former case, a linear penalty term is added in the cost evaluation functions. In the latter case, the recursion is stopped in case of infeasibility (same as setting an infinite penalty). To efficiently perform the concatenations and cost evaluations, each fragment σ within the algorithm is characterized by a total demand $Q(\sigma)$ and distance $D(\sigma)$. Whenever a concatenation operation $\oplus$ between fragments is performed, the associated capacities and distances are derived for the new fragment. Equations (2-1–2-2) compute these values by induction on the concatenation operation in $O(1)$

time:

$$Q(\sigma_1 \oplus \sigma_2) = Q(\sigma_1) + Q(\sigma_2) \tag{2-1}$$

$$D(\sigma_1 \oplus \sigma_2) = D(\sigma_1) + d_{\sigma_1(|\sigma_1|)\sigma_2(1)} + D(\sigma_2). \tag{2-2}$$

Based on this information, the cost of a route or fragment of route can be evaluated as:

$$C(\sigma) = \omega^Q \max\{Q(\sigma) - Q, 0\} + D(\sigma), \tag{2-3}$$

where ω^Q represents the penalty factor for each unit load excess over the vehicle capacity Q. The best solution reconnection found is applied in case of improvement over $\mathcal{R}_{\text{INIT}}$, otherwise, the solution remains unchanged.

2.3.3 Design choices and parameters

Three main decisions need to be taken when applying PILS within a metaheuristic: (1) which solutions are used for pattern extraction, (2) which patterns are injected, and (3) which solutions are submitted to pattern injection.

The success of PILS primarily depends on its ability to extract diverse patterns from high-quality solutions. Indeed, a pool of diverse but low-quality patterns (similar to those found in random solutions) would mainly lead to random moves. In contrast, an overly-restricted pattern set would lead to few possible injections and to excessive guidance towards the same solution characteristics and a resulting loss of diversity. To achieve a meaningful trade-off between these two extremes, our method uses pattern extraction with a fixed probability of P_{EX} on each local minimum produced by the metaheuristic. This design choice, i.e., only extracting patterns from local minima, guarantees a good correlation between pattern frequency and pattern quality while at the same time maintaining diversity (see Section 2.5.2). Moreover, probability P_{EX} drives the computational effort allocated to pattern extraction without changing the characteristics of the extracted patterns.

Regarding the selection of patterns for injection, we observe again that a good performance comes from a trade-off between quality and diversity. In particular, injecting all Φ_{FREQ} frequent patterns would either result in a low diversity whenever Φ_{FREQ} is small, or into a large computational effort whenever Φ_{FREQ} is larger. To achieve a better compromise between diversity and computational effort, a subset $\Phi_{\text{SIZE}} < \Phi_{\text{FREQ}}$ of frequent patterns is selected for injection. These patterns are randomly selected from the heap to form the set of candidate patterns $\mathcal{P}$. PILS then performs a single search loop over the entire neighborhood $\mathcal{N}_{\text{PILS}}(\mathcal{P}, \mathcal{S})$ (iterating over all patterns in $\mathcal{P}$) and directly applies every improving move. Finally, we opted to apply PILS immediately before the local search phases in the respective metaheuristics to maximize its impact on the search trajectory.

2.4 Application of PILS in two CVRP metaheuristics

To demonstrate the robustness and generality of PILS, we study its application within two state-of-the-art metaheuristics for the CVRP: the hybrid genetic search (HGS) of [51], and the knowledge-guided local search (KGLS) of [5]. While both metaheuristics produce high-quality solutions on classical test instances, they are also structurally very different. HGS evolves a diversified pool of solutions using recombination and local search operations, whereas KGLS improves a single incumbent solution in successive steps via a sophisticated local search based on ejection chains. The following paragraphs discuss the main components of these methods and their extension with PILS.

Proposed in [51], HGS (also called HGSADC or UHGS) combines the exploration capabilities of evolutionary algorithms, the improvement capabilities of local searches, and advanced population-diversity management schemes into a very effective solution method for vehicle routing problems. This metaheuristic uses the classical order crossover (OX) and giant-tour solution representation of [40] to generate new solutions that are improved by local search with the classical RELOCATE, SWAP, 2-OPT and 2-OPT* neighborhoods. Population diversity is preserved during the search via an active population management and biased fitness function, which favors diverse and high-quality individuals, as well as active diversification phases, which consists of reintroducing new initial solutions in the population. Due to its simplicity and generality, HGS emerged as the first algorithm able to produce state-of-the-art results for over sixty vehicle routing problem variants and other permutation-based problems, finding the best-known results for thousands of classical benchmark instances [49]. To adapt this method, we simply include the pattern extraction step of PILS with probability $P_{\text{EX}} = 10\%$ after the classical local search, and the pattern injection step before it. The structure of the resulting algorithm is displayed in Algorithm 3.

KGLS [5] is a local-search based metaheuristic with a single solution trajectory that embeds sophisticated and complementary local search operators into a guided-local search framework [4]. The local search relies on CROSS-EXCHANGE and RELATION-CHAINS operators for inter-route improvement, as well as Lin-Kernighan heuristic [7] for intra-route solution improvement. From an initial solution obtained from a construction procedure, KGLS iteratively

Algorithm 3: HGS with PILS

```
1  Generate initial population
2  while CPU time < $T_{\text{MAX}}$ do
3      Select $P_1$ and $P_2$ in the population
4      Generate offspring $C$ by crossover of $P_1$ and $P_2$
5      Apply Pattern Injection on $C$
6      Apply Local Search on $C$ (using Relocate, Swap, 2-opt
         and 2-opt*)
7      if $C$ is infeasible then Repair $C$;
8      With probability $P_{\text{EX}}$, apply Pattern Extraction on $C$
9      Select survivors whenever maximum population size is attained
10     if $It_{\text{DIV}}$ iterations without improvement then Diversify
         population;
```

identifies and penalizes a subset of undesirable edges with large width and length and applies the local search algorithm, which will be therefore guided towards new solutions. Moreover, very sophisticated neighborhood restrictions and data structures contribute to enhance the effectiveness of the local search, resulting in a scalable algorithm that effectively solves very large CVRP instances [6]. To apply PILS within KGLS, we simply include the pattern injection function immediately before each local search phase with $P_{\text{EX}} = 100\%$ (since KGLS generates fewer local minima than HGS), and the pattern extraction function afterward, as described in Algorithm 4.

Algorithm 4: KGLS with PILS.

```
1  Construct an initial solution $S$
2  Apply Local Search on $S$
3  while CPU time < $T_{\text{MAX}}$ do
4      Penalize undesirable edges in $S$
5      Apply Pattern injection on $S$
6      Apply Local Search on $S$ (using Cross-Exchange,
         Relocation-Chains and Lin-Kernighan algorithm)
7      Apply Pattern extraction on $S$
```

2.5 Computational Experiments

The goal of our computational experiments is threefold. A first experiment aims as setting the parameters of PILS and estimate the impact of PILS' main design decisions and parameters on its performance. In a second experiment we examine the main corollary underpinning the PILS heuristic: that pattern frequency and quality are correlated, i.e., that high-quality patterns more frequently appear in high-quality solutions. A final experiment attempts to evaluate the impact of different instance characteristics on the performance of PILS and the estimate the benefits of PILS when integrated into state-of-the-art metaheuristics in general. Each of these analyses will be covered in a dedicated subsection.

All experiments are executed on the classical CVRP datasets of [50], containing 100 benchmark instances with 100 to 1000 customer requests, different depot configurations (R=random, C=centered, E=eccentric), customer distributions (R=uniform, C=clustered, RC=mixed), and different average route length (short routes with large customer demands relative to the vehicle capacity, or longer routes with small customer demands relative to the vehicle capacity). We integrate PILS into the HGS and KGLS metaheuristics as specified in Section 2.4, leading to method variants called HGS-PILS and KGLS-PILS. HGS is coded in C++ and compiled with GCC 7.2.0, whereas KGLS uses Java 9.0.4. In all experiments, these methods are run on a single core of a Xeon X5675 3.07 GHz with 16 GB of RAM. Source code, datasets, and detailed experimental results are available at `https://w1.cirrelt.ca/~vidalt/en/VRP-resources.html`.

2.5.1 Impact of PILS parameters

The functioning of PILS is determined by three main parameters: Φ_{FREQ}, Φ_{SIZE}, and L_{MAX}. Parameter Φ_{FREQ} is the number of most-frequent patterns that are monitored in the heap and therefore drives the diversity of the search. Larger values allow tentative insertions of a more diverse set of patterns, whereas smaller values guide the search towards fewer *elite* patterns. Parameters Φ_{SIZE} and L_{MAX} control the number of patterns of each size that are tentatively injected in each search phase and the maximum pattern size respectively. These two parameters establish a trade-off between computational effort and solution improvement potential. Large patterns, in particular, can lead to higher-order PILS moves that are difficult to find otherwise, but their injections require reconnecting a larger number of solution fragments, leading to larger recursion depths in Algorithm 2.

We first evaluate the sensitivity of HGS-PILS and KGLS-PILS to changes in these parameters. Starting from a baseline configuration in which $\Phi_{\text{FREQ}} = 5n$, $\Phi_{\text{SIZE}} = n$ and $L_{\text{MAX}} = 5$ obtained from an initial calibration, we vary each parameter in turn (a so-called "one factor at a time" or OFAT analysis) to measure its impact on HGS-PILS and KGLS-PILS. For each configuration and problem instance, we execute the two methods five times with different random seeds and initial solutions, setting a CPU time limit linearly proportional to the number of customers, using 240 seconds for each 100 customers. The average results of these configurations over all instances and runs are reported in Table 2.1. The left part of the table describes the investigated parameter configurations, whereas the right part of the table reports, for each of the two methods, the average solution quality and the fraction of the total computing time (in percent) used by PILS (extraction and injection). The quality of the solution is reported as a gap to the optimal or best-known solution value for this instance, computed for each instance as $\textsc{Gap}(\%) = 100(z - z_{\text{BKS}})/z_{\text{BKS}}$, where z is the solution value obtained by the method and z_{BKS} represents the optimal or best known solution value (BKS) for this instance in the literature. The total computational time used by PILS is reported as $\text{T}_{\text{PILS}}(\%)$. Good solutions therefore correspond to gap values that are close to zero.

Table 2.1: Parameter sensitivity analysis

				HGS		KGLS	
PILS	Φ_{Freq}	Φ_{Size}	L_{MAX}	Gap(%)	$\text{T}_{\text{PILS}}(\%)$	Gap(%)	$\text{T}_{\text{PILS}}(\%)$
ON	5n	n	5	0.242	45.86	0.520	7.28
OFF	–	–	–	0.273	–	0.555	–
ON	5n	n	3	0.267	23.66	0.546	3.01
ON	5n	n	4	0.248	35.18	0.536	4.93
ON	5n	n	6	0.261	55.89	0.525	9.78
ON	5n	n	7	0.284	64.53	0.525	12.45
ON	5n	0.2n	5	0.257	15.47	0.534	2.29
ON	5n	0.5n	5	0.246	30.45	0.522	4.25
ON	5n	1.5n	5	0.258	55.56	0.537	10.03
ON	5n	2n	5	0.258	62.23	0.534	12.61
ON	2n	n	5	0.274	44.44	0.532	6.09
ON	3n	n	5	0.255	44.98	0.529	6.63
ON	10n	n	5	0.262	47.22	0.535	8.13
ON	20n	n	5	0.270	48.56	0.527	8.96

Remarkably, the wide majority of the considered HGS-PILS configurations as well as all the KGLS-PILS configurations lead to performance improvements over the baseline configuration in which PILS is deactivated (second line in Table 2.1). Some search parameters such as L_{MAX} and Φ_{Freq} have a larger influence of the method performance, whereas the value of Φ_{Size} has less impact.

Inserting too large patterns with $L_{\text{MAX}} = 7$ or beyond leads to a diminished final solution quality, since the large number of solution fragments needing reconnection significantly increases the share of time spent in Algorithm 2. Similarly, inserting only small patterns with $L_{\text{MAX}} = 3$ does not allow to fully exploit PILS search capabilities.

The influence of Φ_{FREQ} on search performance is also visible. Small values of this parameter should be avoided, as they lead to reduced search diversity and worse performance, whereas large values distribute the computational effort of PILS over too many (possibly less frequent) patterns.

Parameter Φ_{SIZE} directly drives the number of tentative insertions and the share of time spent in PILS before reaching the termination criterion. Interestingly, HGS-PILS configurations with $\Phi_{\text{SIZE}} = 1.5$ or 2.0 spend more than 60% of their total CPU time in PILS, but still perform better than a simple deactivation of PILS. This means that the time spent within PILS is at least as meaningful for the search success as the time spent in a conventional local search, which represents most of the remaining computational effort.

Finally, we note that PILS represents a smaller proportion of KGLS-PILS computational effort (13% at most) than that of HGS-PILS. This is due to the fact that KGLS relies on a trajectory-based search with more sophisticated and time-consuming local-search operators than HGS (Cross-Exchange, Relocation Chains, and the Lin-Kernighan heuristic), such that the share of time spent in PILS is naturally smaller. Despite this behavioral difference, it is notable that our baseline configuration, in which $\Phi_{\text{FREQ}} = 5n$, $\Phi_{\text{SIZE}} = n$, and $L_{\text{MAX}} = 5$, represents a good choice for both algorithms. We therefore opted to maintain this configuration for the remainder of the study.

2.5.2 Pattern frequency versus solution quality

Our second set of experiments investigates the relation between the frequency of the patterns and the quality of the solutions in which they appear. For this experiment, we use a smaller subset of 10 instances with 200 to 300 delivery locations for which optimal solutions are known. We run our baseline configuration and interrupt the search after 20% of the CPU time to analyze the pattern pool at an early stage of the search. For each pattern size, we sort the resulting patterns from most frequent to least frequent and distribute them into equal-sized bins containing n patterns each. Finally, we calculate in each bin the fraction of patterns that appear in the optimal solution. The result of this analysis is displayed in Figure 2.3.

The results of this analysis demonstrate, for both HGS-PILS and KGLS-

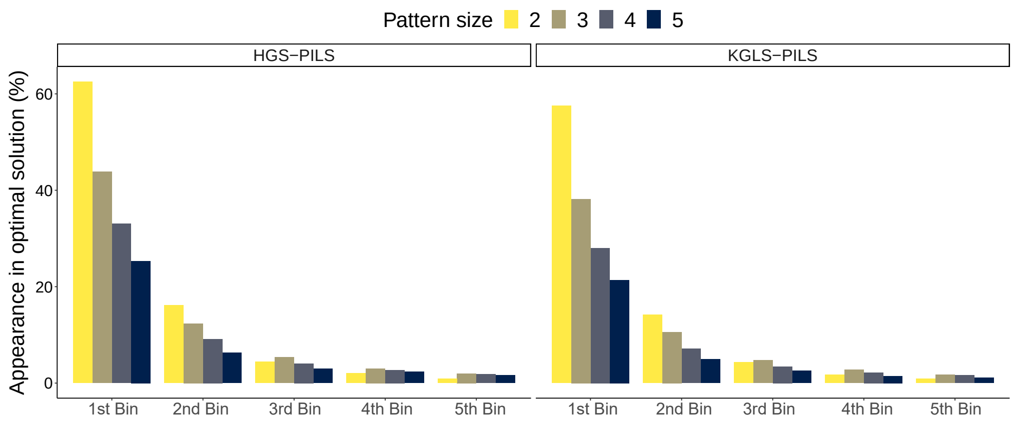


Figure 2.3: Pattern frequency and appearance in optimal solutions

PILS, that the most frequent patterns (left) have a much higher probability to be part of optimal solutions than the less frequent ones (right). Moreover, the probability to belong to the optimal solution decreases when the pattern size l grows, highlighting that larger optimal patterns are generally more difficult to identify. This behavior was expected since long patterns are much more informative on the structure of optimal CVRP solutions and therefore likely to be more difficult to identify.

Since frequent patterns appear more frequently in optimal solutions, we can also examine whether they are also found in higher-quality solutions in general. We therefore conduct an additional analysis that consist in storing, during the search, each pattern along with the objective value of the best solution in which it appeared. As in the previous experiment, the patterns are sorted by frequency and grouped into equal-sized bins containing n patterns each. Figure 2.4 reports the average quality Gap(%) of each bin over all runs and instances.

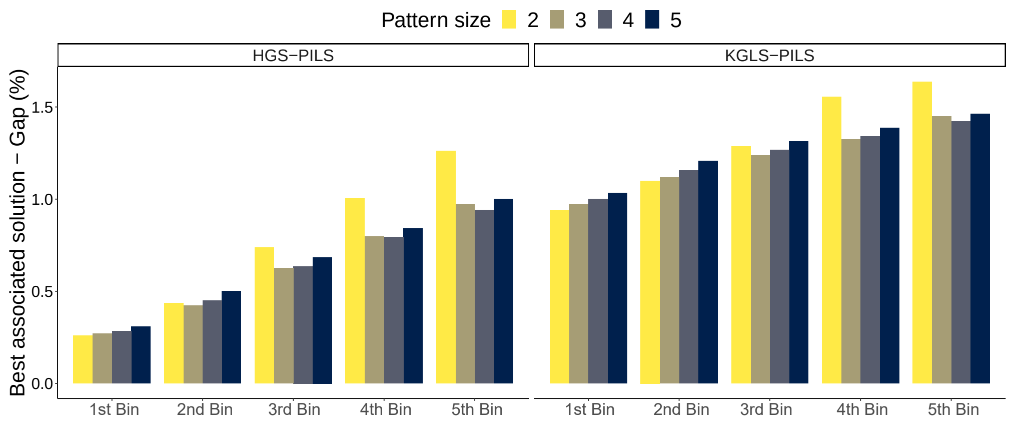


Figure 2.4: Pattern frequency and quality of the associated solutions

These results confirm the positive correlation between pattern frequency and solution quality. The most frequent patterns are, on average, found in solutions of better quality, up to 0.30% better when comparing the solution quality associated with the patterns of the first bin with that of the last bin considered in the figure (fifth bin). These two experiments confirm our initial hypothesis that pattern frequency is a good surrogate for quality, and allow to concentrate the mining procedure on frequent patterns without a need for other filters related to solution quality.

2.5.3
Performance impact of PILS

Our last experiment evaluates the performance impact of PILS when applied on instances with different characteristics. Table 2.2 displays the results of the comparison of the classical HGS with HGS-PILS and KGLS with KGLS-PILS. For each pair of methods and each instance subset, this table displays the average gap values of both approaches, the percentage time spent in PILS, as well as the result of a paired-samples Wilcoxon test (at a significance level of $p = 0.05$) evaluating the statistical significance of the performance difference. The first line corresponds to the complete set of instances, whereas each other line selects a subset of instances with different characteristics, e.g., size, depot location, route length, and customer distribution, using the same nomenclature as in [50].

Table 2.2: Impact of PILS on solution quality for HGS and KGLS different subsets of instances

		HGS	**HGS-PILS**		**Sign.**	**KGLS**	**KGLS-PILS**		**Sign.**
Category	#	Gap(%)	Gap(%)	T_{PILS}(%)		Gap(%)	Gap(%)	T_{PILS}(%)	
All	100	0.273	0.242	45.86	✓	0.555	0.520	7.28	✓
Smallest	50	0.129	0.108	45.15	✓	0.413	0.379	6.44	✓
Largest	50	0.418	0.376	46.56	✓	0.696	0.662	8.12	✓
Short routes	40	0.226	0.206	44.08		0.581	0.517	7.64	✓
Long routes	40	0.337	0.280	47.94	✓	0.564	0.548	6.60	
Depot (R)	34	0.317	0.287	46.45		0.635	0.558	7.51	✓
Depot (E)	34	0.267	0.203	45.15	✓	0.487	0.468	5.92	
Depot (C)	32	0.234	0.235	45.99		0.542	0.537	8.48	✓
Customer (RC)	34	0.258	0.228	46.81	✓	0.553	0.515	8.13	✓
Customer (C)	32	0.259	0.227	45.01	✓	0.551	0.545	6.40	✓
Customer (R)	34	0.301	0.271	45.70	✓	0.560	0.503	7.26	✓

The results in Table 2.2 show that PILS improves the overall performance of both metaheuristics despite their structural differences (population-based versus local search-based). The average gap of HGS over all instances decreases by 0.031% when combined with PILS, while the average gap of KGLS decreases by 0.035%. Solution improvements become increasingly difficult as we approach the optimal or best-known values, such then even a small quality improvement of the order of 0.03% over the state-of-the-art is an important achievement.

PILS benefits the search equally on small and large instances, with significant effects observed in both cases. It also improves the performance of HGS for instances with long routes containing many customer visits, likely due to the fact that it compensates for the simplicity of its intra-route neighborhood operators. In contrast, KGLS already uses a sophisticated implementation of Lin-Kernighan algorithm for effective intra-route optimization, but encounters more difficulties to optimize customer allocations among different routes on instances with short routes (i.e., larger customer demands relative to the vehicle capacities). In this situation, we observe that PILS significantly boosts KGLS performance with complementary moves that compensate for this weakness.

To gain more insights into the moves that are applied by PILS, we collect a variety of statistics about the injected patterns, as reported in Figures 2.5 to 2.8. These figures represent the proportion of applied PILS moves for each "move order" (i.e., number of replaced edges), pattern size, and number of involved routes, during all HGS-PILS and KGLS-PILS executions.

From these experiments, we observe that the largest and smallest patterns are equally likely to be injected. The majority of PILS moves (approximately 80%) modify two to five edges, but some larger moves involving up to ten edges are also found and applied to improve the solutions. The ability to find such high-order moves (e.g., improving 9-OPT or 10-OPT moves) in a controllable amount of time is noteworthy. Finally, the proportion of time dedicated to PILS remains stable for all subgroups of instances, never exceeding more than 50% of the total search effort for HGS-PILS, and 10% of the total search effort for KGLS-PILS.

In a final analysis, Figure 2.9 shows to which extent PILS influences the performance of the two metaheuristics over time. It reports the average Gap(%) of HGS, HGS-PILS, KGLS and KGLS-PILS at different time steps: after 1%, 2%, 5%, 10%, 15%, 20%, 30%, 50%, 75% and 100% of the allotted time. One would expect that PILS requires some time to learn good solution patterns and, therefore, that it contributes to the search performance only at later stages. Yet, in the case of HGS-PILS, our experiments do not necessarily corroborate this initial intuition since PILS already boosts the convergence at early stages of the search: solution-quality differences are already visible after around 10% of the total search time. In contrast, PILS appears to impact the search trajectory of KGLS only at later search stages. This behavior is confirmed

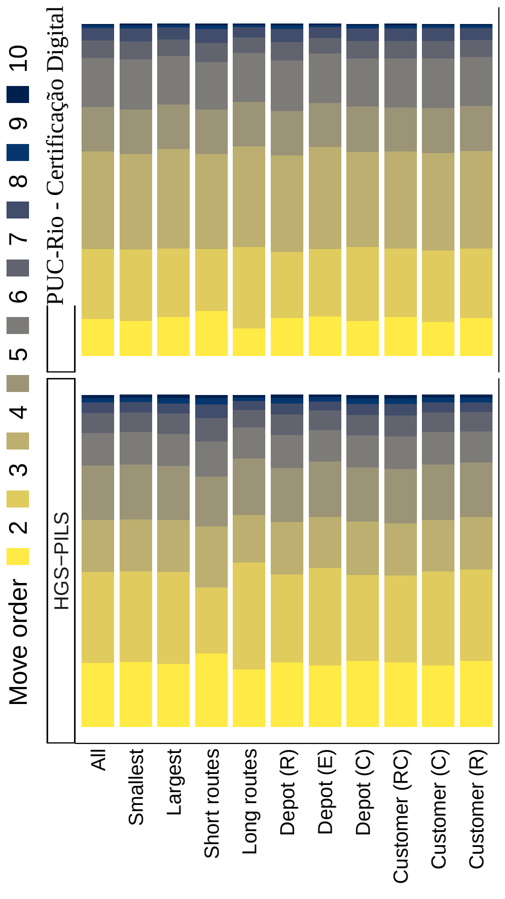


Figure 2.5: Proportion of applied PILS moves per "move order"

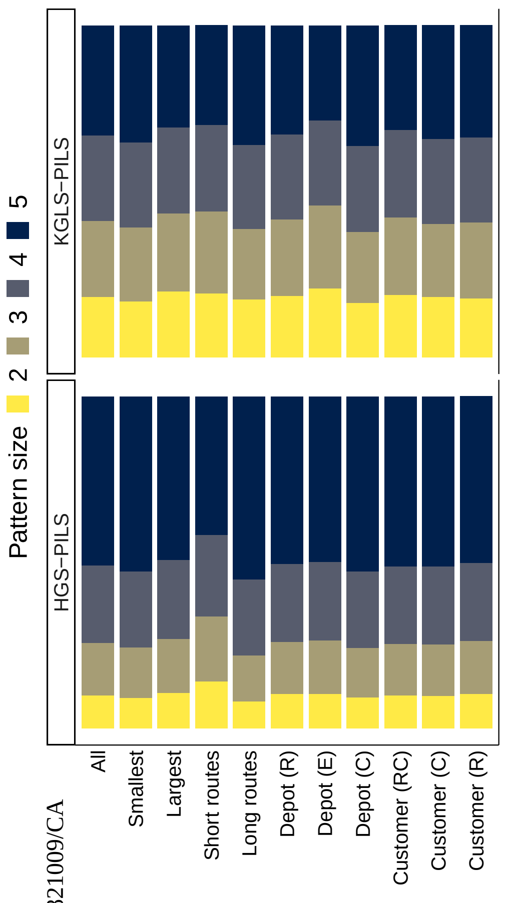


Figure 2.6: Proportion of applied PILS moves per pattern size

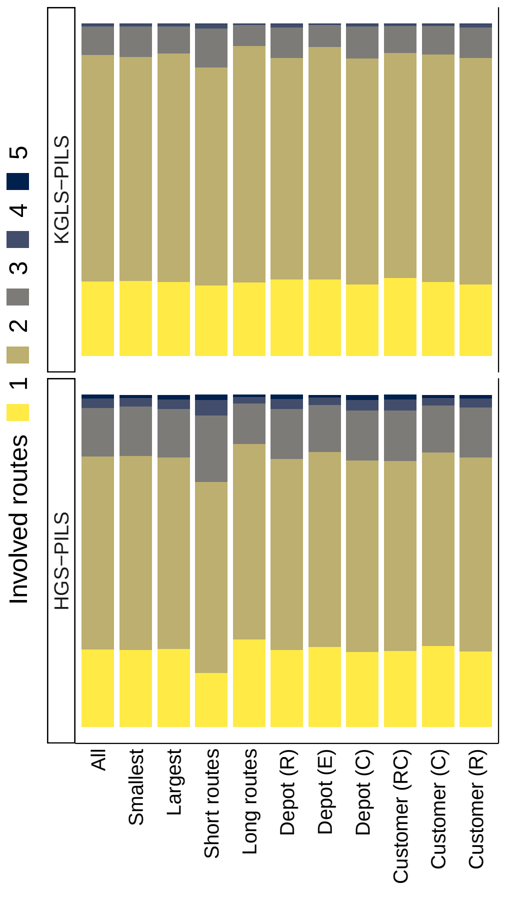


Figure 2.7: Proportion of applied PILS moves per number of involved routes

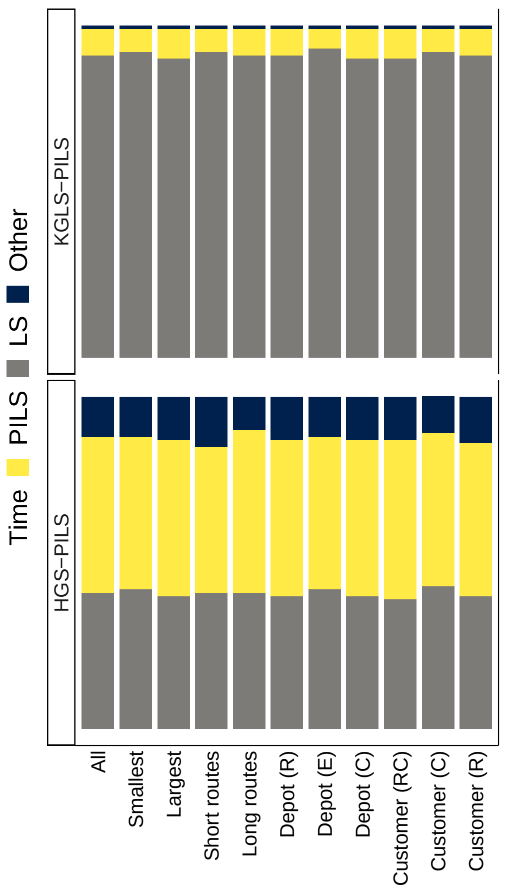


Figure 2.8: Proportion of CPU time spent in different components of the algorithms

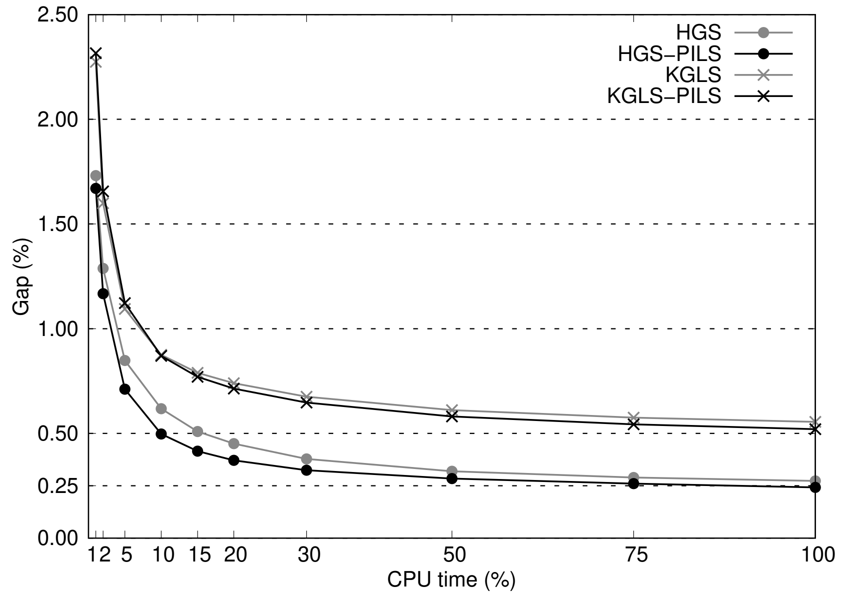


Figure 2.9: Convergence of HGS, HGS-PILS, KGLS and KGLS-PILS solutions over time

on Table 2.3 by the results of paired-samples Wilcoxon tests for the method pairs HGS/HGS-PILS and KGLS/KGLS-PILS at the different stages of the search. A likely cause for this observation is that KGLS performs extraction steps only from a single incumbent solution instead of from a population, and thus it takes more time to learn a diversified set of patterns.

Table 2.3: Impact of PILS at different stages of the search – Statistical significance

CPU Time		1%	2%	5%	10%	15%	20%	30%	50%	75%	100%
Wilcoxon HGS	p	<0.001	<0.001	<0.001	<0.001	<0.001	<0.001	<0.001	<0.001	<0.001	<0.001
with HGS-PILS	sign	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Wilcoxon KGLS	p	0.418	0.305	0.386	0.075	0.001	0.001	<0.001	<0.001	<0.001	<0.001
with KGLS-PILS	sign					✓	✓	✓	✓	✓	✓

2.6 Conclusions

In this study, we have introduced PILS: a simple and versatile strategy to identify high-order local-search moves using frequent pattern mining. Our PILS application to the CVRP, a notoriously difficult combinatorial optimization problem, is built upon an effective recursive algorithm that optimally rebuilds solutions from a set of route fragments and a pattern. We integrated PILS into two structurally different state-of-the-art metaheuristics, one population-based algorithm and one trajectory-based algorithm, to evaluate its ability to find new moves and contribute to the search performance. Our experiments confirm that pattern frequency is positively correlated with solution quality and pattern appearance probability within optimal solutions. Moreover, for a fixed computational effort, PILS significantly enhances metaheuristic performance and compensates the weaknesses of each metaheuristic for specific instance subgroups. It complements classical local search operators and identifies synergistic high-order moves, which would never be found otherwise.

Numerous possible future research avenues arise from this study. Firstly, PILS can easily be extended to different combinatorial optimization problems and to solution structures that may require more sophisticated pattern extraction strategies. Secondly, one could also attempt to learn *undesirable* solution patterns to complement the information of the promising ones. Such patterns should of course be *removed* from solutions rather than inserted into them.

Finally, from a more general viewpoint, PILS research occurs in a context in which metaheuristic and pattern recognition research can mutually benefit from each other. Indeed, pattern recognition models often lead to intractable problems, which call for efficient heuristic solution approaches, while metaheuristic research directly benefits from enhanced learning strategies. Indeed, the largest part of metaheuristic research, over the past two decades, has been dedicated to finding simple and efficient strategies to guide surrogate (e.g., constructive or local search) heuristics towards promising search-space regions, construction decisions and moves. Learning desirable solution structures is therefore a defining task for metaheuristic search. Given the tremendous experimental and theoretical progress recently made on a variety of learning algorithms (e.g., belief propagation, deep neural networks, reinforcement learning) and their successful application to some combinatorial optimization problems (e.g., [73, 74]), it is increasingly important to join the strengths and analysis techniques of both fields, to progress towards a new generation of algorithms which remain conceptually simple, amenable to analytic reasoning, and effective. We hope that this first study connecting pattern mining and local search will encourage future work in this direction.

3
Neural Networks for Local Search and Crossover in Vehicle Routing: A Possible Overkill?

3.1
Introduction

Vehicle routing problems (VRP) represent one of the most studied classes of NP-hard problems due to their practical difficulty and ubiquity in real-life applications such as food distribution, parcel delivery, or waste collection, among others [49, 132]. Problems in this class generally seek to plan efficient itineraries for a fleet of vehicles to service a geographically-dispersed set of customers. The capacitated VRP (CVRP) is the most canonical variant among all existing routing problems. Its objective is to minimize the total distance traveled by the vehicles to service the customers, subject to a single constraint representing the vehicle capacities: i.e., the sum of customers' demands over a route should not exceed the vehicle capacity.

Over the years, there have been dramatic improvements in the heuristic and exact (i.e., provably optimal) solution of VRPs. To date, the best performing exact algorithms rely on branch-cut-and-price strategies, with tailored cutting-plane algorithms and sophisticated column-generation routines [56, 134]. With these methods, it is now possible to solve most existing instances with 200 or 300 customers. However, the time for an exact solution remains highly volatile at this scale, and most larger instances remain unsolved. Consequently, extensive research has been conducted on metaheuristics for this problem to find high-quality solutions in a shorter and more controlled time [134].

As it stands now, metaheuristics can consistently locate high-quality solutions for CVRPs with up to 1,000 customers in a matter of minutes [135, 136]. Of all existing methods, the Hybrid Genetic Search (HGS) algorithm developed in [51, 63] and [136] is known to achieve the best-known solution quality consistently on most problems and instances of interest. Notably, during the 12th DIMACS implementation challenge on the CVRP organized in 2022 [137], it was used as the base algorithm for four out of the five best methods. Very-large problem instances counting dozens of thousands of customers can also be solved using tailored data structures and decomposition strategies [138, 138]. Considering the latest generation of metaheuristics such as HGS, it is clear that two main operators —local search and crossover— are instrumental in finding improving solutions.

– **Local Search (LS)** consists in systematically exploring a neighborhood obtained by small changes over a current solution to identify improvements. This process is iterated until attaining a local minimum. Classical neighborhoods for the CVRP involve exchanges or relocations of client visits and edge reconnections. They typically include $\mathcal{O}(n^2)$ possible neighbors, where n represents the number of customers. Due to its iterative nature, LS typically takes the largest share of the computational time. Several techniques have been developed to reduce computational complexity. In particular, [139] observed that the search could be limited to relocations and exchanges of customers that are *geographically related.* The resulting strategy, called granular search, limits classical neighborhoods to $\mathcal{O}(\Gamma n)$ moves, where Γ is a user-defined parameter. However, although very simple in design, a straightforward distance-based relatedness criterion may hinder the search process, especially if optimal solutions require a few long edges.
– In contrast, **Crossover** operators focus on diversifying the search. They consist of recombining two existing (parent) solutions into a new (offspring) solution that inherits promising characteristics from both. For the CVRP, crossover operators are not primarily designed for solution improvement, but instead used to create promising starting points for subsequent LS. Various crossovers have been used in previous works [140, 136]. As shown in Section 3.2, the Ordered Crossover (OX) is widely used, and consists in juxtaposing a fragment of the first solution with the remaining client visits ordered as in the second solution. By doing so, it implicitly creates a re-connection point, which is typically random.

Note that in both LS and Crossover, there is interest in using relatedness information between client vertices to (i) speed up the LS or (ii) identify a subset of more promising crossover operations. It is also noteworthy that the relatedness information used until now for the LS (and possibly used for the crossover) is a broader concept that goes beyond simple distance criteria, and which could be possibly learned.

In recent years, graph neural networks (GNNs) have emerged as a tool to apply machine learning techniques to combinatorial optimization problems posed over graphs. To the best of our knowledge, the first attempt in this context was proposed by [141] for the TSP. Underpinned by enhancements in hardware and artificial intelligence research over the last years, the development of deep NNs made them relevant to a wide range of difficult combinatorial optimization problems, such as SAT, Minimum Vertex Cover, and Maximum Cut [142, 143]. When applied to solve CVRPs, these networks are usually combined with reinforcement learning (RL [144, 145]) or typically used for node classification or edge prediction [146, 147]. Despite extensive research, GNNs for directly solving CVRPs remain limited to small problem instances with up to 100 customers and generally do not compare favorably with classic optimization methods (exact or heuristic) in terms of solution quality. This is possibly due to the fact that good solutions for combinatorial optimization problems result from tacit structural knowledge about the problem (learnable solution structure) along with a significant amount of trial-and-error to build the best possible solution fitting almost perfectly the constraints at hand. After all, an optimal solution is a very specific outlier. Whereas better knowledge of solution structures can be learned to guide the search, avoiding some (explicit or implicit) enumeration of solutions without compromising solution quality is generally challenging. Consequently, using pure learning algorithms without any other form of solution enumeration is likely to be unsuccessful.

Given these observations, a promising path toward better solution methods for VRPs concern the hybridization of learning-based and traditional solution methods. In this study, in particular, we aim to learn and use relatedness information in the LS and crossover operators of HGS to improve this state-of-the-art method substantially. We capitalize upon the work of [146], which trained a GNN to predict occurrence probabilities of edges in high-quality solutions (i.e., heatmap), and used this information to sparsify the underlying graph and accelerate related solution procedures. Instead, we leverage the heatmaps as a source of relatedness information to define neighborhood restrictions in the LS and possible re-connection points in the crossover. Throughout an extensive experimental campaign, we evaluate how HGS' performance varies with these surgical changes, for better or worse, on more than 10000 different instances containing from 100 to 1000 customers. Therefore, we make the following specific contributions.

1. We introduce a framework for defining and exploiting *relatedness* information between pairs of customers in the context of the CVRP.

2. We show that relatedness measures can be exploited to steer the LS towards the most promising moves. Additionally, we use relatedness information to extend the classical OX crossover, trading some of its

inherent randomness for better choices of re-connection points between the parents. Our approaches are generic and applicable to any type of relatedness measure. We will specifically consider relatedness from two sources: geographical relatedness as given by the distance between two customers, and learnable relatedness (i.e., heatmap) obtained from a GNN.

3. We suggest a practical technique to exploit the output of a single GNN (heatmaps for fixed-size graphs) for problem instances of varying size. To achieve this, we decompose the original instance into a sequence of fixed-size subproblems and aggregate the resulting heatmap information. This approach has the benefit of only requiring a single trained model of moderate size.

4. Finally, we conduct an extensive computational campaign to measure the enhancements achieved with the proposed techniques and analyze the impact of each change. We observe that incorporating relatedness information within the crossover and LS operators largely benefit the search, such that learning-based approaches seem to be successful at first sight. However, after a closer analysis, we also observe that these improvements are mostly insensitive to the source of the relatedness information (geographical or learned). Therefore, the problem-specific knowledge and strategies that we integrated contributed more to the algorithm's performance than GNN-based algorithms for defining relatedness.

3.2 Methodology

The CVRP is defined over a complete graph $G = (V, E)$, where the set of vertices $V = \{0, 1 \ldots, n\}$ contains a vertex 0 representing the depot, and the remaining vertices represent customers. Each customer $i \in \{1, \ldots, n\}$ is characterized by a demand d_i. Edges (i, j) model direct travel between vertices i and j for a distance d_{ij}. A solution to this problem is a set of routes originating and ending at the depot and visiting customers, such that (i) the total demand over each route does not exceed a vehicle-capacity limit Q, (ii) each customer is visited exactly once, and (iii) the total travel distance is minimized.

We additionally assume that we can calculate a *relatedness* metric $\phi(i, j)$ for each edge (i, j). This definition is general: in the simplest setting, relatedness could be the inverse of distance, i.e., $\phi(i, j) = 1/d_{ij}$. In a more informed setting, we can instead consider defining $\phi(i, j)$ as the output of a graph neural network (GNN) as seen in [146], predicting the probability of occurrence of an edge in a high-quality solution. Probabilities of this kind are typically called heatmaps. In the remainder of the study, we will refer to $\phi_{\text{D}}(i,j)$ for distance-relatedness, and $\phi_{\text{N}}(i,j)$ for GNN-based relatedness. This information will now be used to refine the two most important HGS operators.

3.2.1 Hybrid Genetic Search

The Hybrid Genetic Search [136] relies on simple solution generation and improvement steps. A complete pseudo-code is provided in Algorithm 5. The method starts by initializing a population of size μ with random solutions that are improved by local search. After this initialization phase, HGS iteratively generates new solutions by selecting two random solutions in the population, recombining them using an ordered crossover (OX), and applying local search for improvement. To promote exploration, solutions that exceed capacity limits are not directly rejected but instead penalized according to their amount of infeasibility. The penalty weights are adapted during the search to achieve a target percentage of feasible solutions, and infeasible solutions are maintained in a separate subpopulation. Whenever a solution is infeasible after local search, an extra Repair step is applied, which simply consists of a classic local search with a temporarily ($10\times$) higher penalty coefficient.

During the overall search process, the number of solutions in the feasible and infeasible populations is monitored. Whenever any population exceeds $\mu + \lambda$ solutions, a survivors' selection phase is triggered to retain only the best μ individuals, according to a ranking metric based on solution value and contribution to the population diversity. Finally, the algorithm restarts each time n_{IT} consecutive solution generations have been done without improvement of the best solution, and it terminates upon a time limit T_{MAX} by returning the best solution found over all the restarts.

3.2.2 Local Search using Relatedness Measures

Local Search (LS) is a conceptually simple and efficient method to solve combinatorial optimization problems of the form $\min_{x \in X} c(x)$, where X is the space of all solutions and c is the objective function. A neighborhood is defined as a mapping $\mathcal{N} : X \to 2^X$ associating with any solution x a set of neighbors $\mathcal{N}(x) \subset X$. For the CVRP, $\mathcal{N}(x)$ is usually defined relative to a set of operations (i.e., moves) that can modify the current solution x. A move τ is a small modification that can be applied on x to obtain a neighbor

Algorithm 5: Hybrid Genetic Search for the CVRP (HGS)

```
1  Initialize population with random solutions improved by local search
2  while $time < T_{max}$ do
3      Select parent solutions $P_1$ and $P_2$
4      Apply the crossover operator on $P_1$ and $P_2$ to generate an
         offspring $C$
5      Educate offspring $C$ by local search
6      Insert $C$ into respective subpopulation
7      if $C$ is infeasible then
8          With 50% probability, repair $C$ (local search) and insert it
             into respective subpopulation
9      if maximum subpopulation size reached then
10         Select survivors
11     Adjust penalty coefficients for infeasibility
12 Return best feasible solution
```

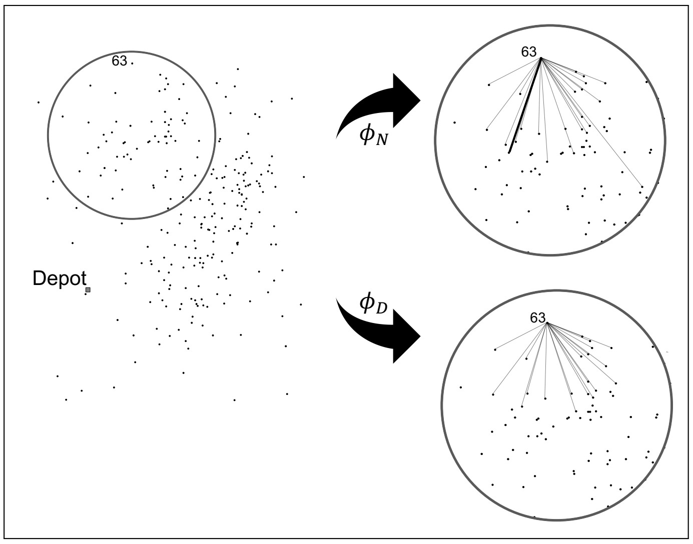
$\tau(x) \in \mathcal{N}(x)$. HGS uses four main types of moves and some of their immediate extensions [136]:

- Relocate: Moves a visit to customer i immediately after a visit to a different customer j or the depot;
- Swap: Exchanges the visits of customers i and j;
- 2-Opt: Reverts a customer-visit sequence $(i, \dots, j)$;
- 2-Opt*: Exchanges customers i and j and their succeeding visits.

The moves are evaluated in a random order of the indices i and j, and any improvement is directly applied. This process is repeated until a local minimum is reached, i.e., a situation where no improving move exists for all the considered neighborhoods. Without further pruning, all these neighborhoods contain $O(n^2)$ solutions. Using incremental calculations (keeping track of partial load and distance over the routes), it is possible to conduct a complete evaluation of all neighborhoods in $O(n^2)$ time. Moreover, the number of complete neighborhood searches (i.e., loops) needed to converge is rarely greater than 10 in practice.

A quadratic complexity for the LS operator is adequate for small problems, but this can become a significant bottleneck otherwise due to its frequent use. Considering this, [139] introduced a "granular search" mechanism that consists in limiting the moves to customer pairs (i, j) that are geographically close, i.e., such that j belongs to a set $\Phi(i)$ formed of the Γ closest customers of i. Consequently, the total number of moves and the complexity of each LS

Twenty customers most related to customer 63 according to ϕ_{D} and ϕ_{N}:



Customer 63 in an optimal solution:

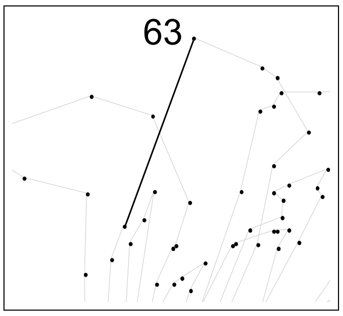


Figure 3.1: Sets of related customers according to ϕ_{D} and ϕ_{N}, on instance X-n247-k50

loop reduce down to $O(n\Gamma)$ time. Indeed, it rarely makes sense to relocate or exchange customer visits that are far away from each other. Moreover, this strategy ensures that each move creates at least one short edge [148, 149, 138].

Since its inception, granular search has been adapted to many VRP variants. Especially, to handle customer constraints on service-time windows, [149] extended the concept to filter node pairs (i, j) based on a compound metric that includes distance, unavoidable waiting times, and unavoidable time-window violations arising from this customer succession.

In this study, we instead extend the filtering criterion by relying on relatedness information from the GNN. As illustrated in Figure 3.1 for instance X-n247-k50 from [50], the Γ most-related customers according to the relatedness metrics ϕ_{D} and ϕ_{N} can differ very significantly. In this particular example, the GNN-based relatedness even includes an edge (in boldface) contained in the optimal solution that is otherwise missing when considering distance only.

For each i, we therefore form the set $\Phi(i)$ in two steps: we first include

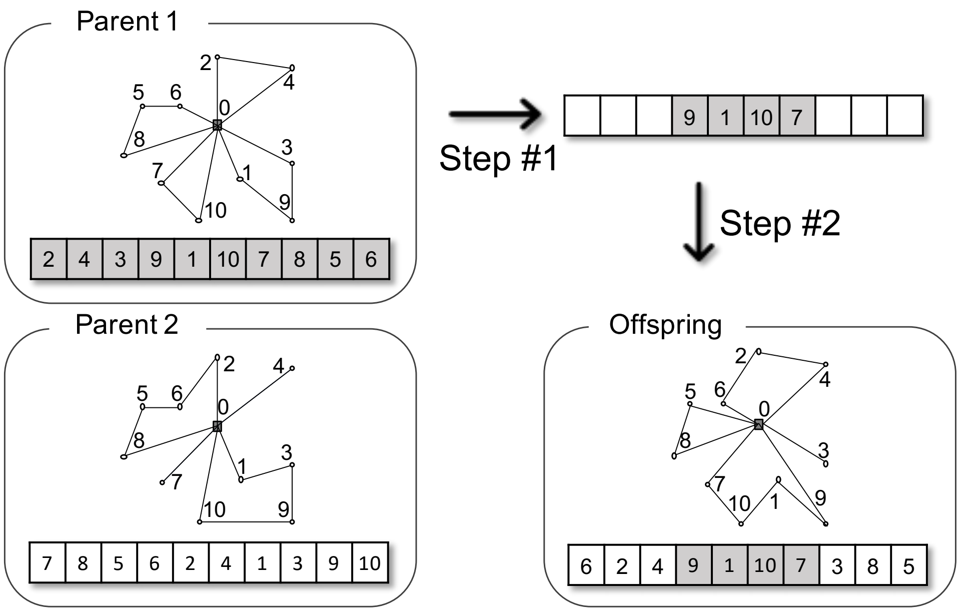


Figure 3.2: Illustration of the ordered crossover (OX)

in $\Phi(i)$ the $\lfloor \Gamma/2 \rfloor$ vertices that are most related to i according to the GNN-based relatedness metric $\phi_{\text{N}}(i,j)$, and then we complete the $\lceil \Gamma/2 \rceil$ remaining customers by increasing distance, therefore according to $\phi_{\text{D}}(i,j)$. This strategy uses learned information and ensures that the $\Gamma/2$ closest customers are still considered in the moves.

3.2.3 Crossover using Relatedness Measures

In HGS, each solution is represented as a single permutation of the customer's visits (i.e., a giant tour) during the crossover operation. This use of this simple representation is motivated by the fact that (i) one can simply represent any complete solution by concatenating the routes and omitting the visits to the depot, and (ii) reversely, given a sequence of customers visits, there exists a linear-time algorithm, called Split, that optimally segments this giant tour into routes [150].

Based on this representation, HGS employs the ordered crossover (OX – [151]) illustrated in Figure 3.2. OX works in two steps. First, a fragment F of the first parent defined by two randomly-selected cutting points is copied in place into an empty offspring. Next, the second parent is scanned from the position of the second cutting point to complete all missing customer visits circularly. This gives a new giant tour, which is then transformed into a complete CVRP solution using Split.

As it stands, OX is completely dependent upon random choices. In particular, the second step tends to concatenate unrelated customers immediately after fragment F. This creates low-quality fragments of solution requiring many LS moves for improvement. To correct this issue, we suggest relying on the relatedness metric to modify the completion step. Let i be the last customer from fragment F. Instead of arbitrarily reconnecting F with the next customer from Parent 2 obtained by a circular sweep, we select a random related customer j among the Γ customers most related to i that are not part of F or, in case of Γ customers most related to i are present in F, one could arbitrarily chose any available customer. Then, proceed to complete the offspring from this position following the order in Parent 2. This small but notable difference permits reconnecting visits that are more closely related among both parents, allowing for better solutions without sacrificing diversity. As previously, the choice of relatedness metric leads to different variants of the OX crossover. In the remainder of this study, we will refer to the modified crossover using distance-relatedness as DOX, and to the modified crossover using GNN-relatedness as NOX. Figure 3.3 illustrates the use of relatedness measures within OX, as discussed in Section 2.3. In this example, which is valid for both NOX and DOX, the algorithm selects fragment $F = [9, 1, 10, 7]$ from Parent 1. With this, $i = 7$ is the last element of F (Step #1). Then, it proceeds to select a random related customer among the Γ most-related customers of i that are not part of F. We assume, in this example, that this customer is $j = 8$. Finally, it proceeds to complete the offspring from this position, following the order in Parent 2 (Step #2).

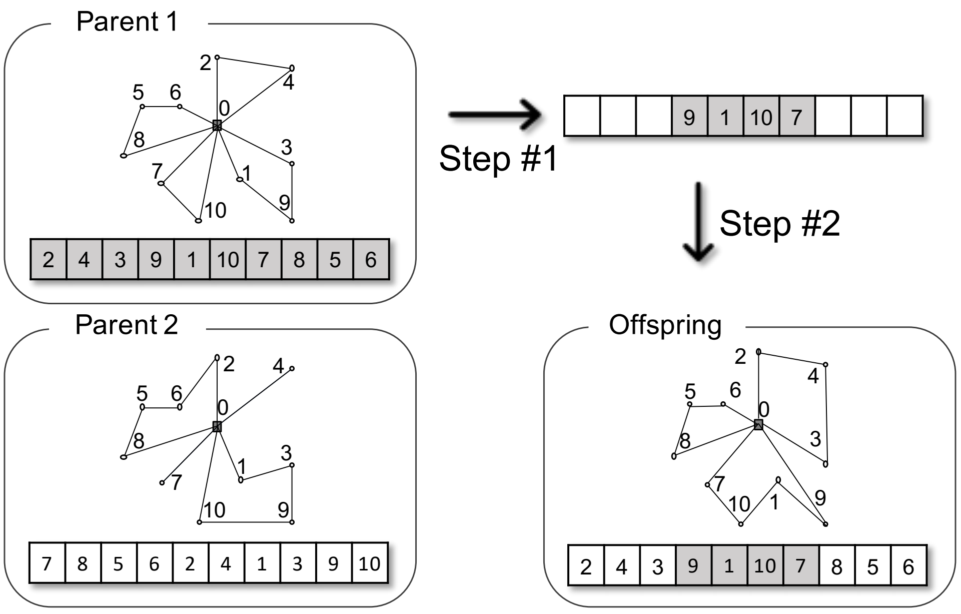


Figure 3.3: Illustration of the crossover using relatedness measures (NOX and DOX)

3.3 Experimental Analyses

This section presents extensive computational experiments designed to: (i) calibrate and evaluate the impact of the granular search parameter Γ, which governs the size of the LS neighborhoods; (ii) measure the impact of our enhancements on the LS and OX operators as well as the usefulness of different relatedness criteria; (iii) confront the characteristics, the computational effort, and the performance of heatmaps produced by different GNN configurations, and (iv) analyze the extension of GNNs originally trained on fixed-size graphs to instances of varying sizes. We address objective (i) in Section 3.3.4, whereas objectives (ii, iii, iv) are covered in Sections 3.3.5 and 3.3.6.

We will analyze, in the following sections, the performance of HGS in its original form (baseline) along with five combinations of ϕ_{D} and ϕ_{N} for local search and crossover operators, which are listed below:

- **HGS-D-O (baseline)**: HGS with granular search and OX;
- **HGS-D-D**: HGS with granular search and DOX;
- **HGS-D-N**: HGS with granular search and NOX;
- **HGS-N-O**: HGS with neural granular search and OX;
- **HGS-N-D**: HGS with neural granular search and DOX;
- **HGS-N-N**: HGS with neural granular search and NOX.

3.3.1 Computational Environment

All experiments are run on a single thread of an Intel Gold 6148 Skylake 2.4 GHz processor with 40 GB of RAM and NVIDIA Tesla P100 Pascal (12 G memory), running CentOS 7.8.2003. Unless otherwise stated, we use the original parameters defined for HGS in [136] and the GNN in [146]. To achieve fast convergence, we set smaller values for the population-size parameters in HGS: $\mu = 12$ and $\lambda = 20$. We compile HGS with g++ 9.1.0 and execute the GNN using Python 3.8.8 on Torch 1.9.1.

3.3.2 Benchmark Instances

Our experiments use two main sets of CVRP instances: Set X from [50], and Set XML from [152]. Set X is a well-known benchmark of 100 instances containing between 100 and 1000 customers. This set includes very diverse instances that mimic important characteristics of real-world situations concerning depot positions, route length, customer demands, and locations. The XML set [152] includes 10,000 instances of 100 customers each, drawn from a similar distribution as set X. One advantage of the XML set is that number of customers is constant in all instances, and all optimal solutions are provided. This permits comparisons with proven optima instead of best-known solutions (BKS) collected from all previous works. In contrast, many instances of set X are still unsolved to proven optimality.

3.3.3 Parametrization and Training of the GNN

The GNN proposed by [146] is a seminal work of the proposals of [161, 157], built for solving graph-based learning problems and the traveling salesperson problem (TSP). Their proposals were based on the Graph Convolutional Network of [161], which takes a graph as an input and extracts compositional features from its nodes and edges by stacking several graph convolutional layers. The output of the neural network is an edge adjacency matrix denoting the probabilities of edges occurring on the TSP tour.

These networks [146, 161, 157] are designed to be trained and applied for prediction over graphs (i.e., instances) of fixed size. The authors provided the final model trained on instances containing 100 customers, which can therefore be directly applied for inference on the XML instances. In contrast, Set X has instances with different numbers of customers, such that a different approach is needed to use the heatmaps. Due to these key differences, we will subdivide the presentation of our experiments into two parts, with results on XML instances in Section 3.3.5, and adaptations and results for Set X in Section 3.3.6.

In these experiments, we use the original trained GNN from [146] for heatmap generation, called Original in the rest of this study. However, even though this model is already trained, it takes around 0.85 seconds of inference time to produce the heatmap for a given instance. This is a similar order of magnitude as the time needed by HGS to solve the CVRP to near optimality (i.e., below 0.1% error) on instances containing 100 customers. Since we aim to compare CVRP solution algorithms under the same total CPU time budget (counting inference time and solution time), GNN-based methods would be at a disadvantage if a large share of the CPU time is invested in the inference step. Therefore, to estimate the performance of GNN-based algorithms in the most optimistic conditions (e.g., considering a hypothetical scenario where GPU inference is extremely fast), we will also report the results of the same method ignoring inference time. Additionally, we produce results (counting inference time) obtained with two lighter versions of the GNN, called MODEL #1 and MODEL #2, which were trained on the same examples as [146], with fewer internal nodes and internal layers. Table 3.1 summarizes the parameter setting of all the considered GNNs.

Table 3.1: GNN configurations

GNN	#NODES	#LAYERS	#EPOCHS	PRED-T(S)
ORIGINAL	300	30	1500	0.85
OPTIMISTIC	300	30	1500	IGNORED
MODEL #1	10	5	500	0.03
MODEL #2	10	5	1500	0.03

This table lists for each GNN the number of hidden layers (# LAYERS), nodes per layer (# NODES), and epochs (# EPOCHS) used for training. Finally, the last column reports the average inference time on an XML instance. The parameters of MODEL #1 and MODEL #2 were selected to achieve training and inference in a limited time. MODEL #1 (resp. #2) required 8 (resp. 24) hours of training time on our hardware.

3.3.4 Calibration of the Local Search

We focus here on the parameter Γ, which drives the exploration breadth of the LS (see Section 3.2.2) and significantly impacts the computational time of HGS. The aim of this experiment is to select a meaningful range of values for this parameter. Based on standard values used in previous works, we evaluate configurations $\Gamma \in \{5, 10, 15, 20, 30, 50, 100\}$ and analyze the sensitivity of the baseline method (i.e., HGS-D-O) to this parameter. To keep a simple experimental design, we focus on the performance of the LS by generating ten random initial solutions for each instance and applying a single LS to each of these solutions. We then report in Table 3.2 the quality of the best solution found as well as the computational time used by the ten LS runs.

In Table 3.2 and the rest of this study, solution quality is expressed as a percentage error gap calculated as $\text{GAP}(\%) = 100 \times (z - z_{\text{BKS}})/z_{\text{BKS}}$, where z represents the cost of the solution and z_{BKS} is the optimal or BKS cost value.

Table 3.2: Impact of Γ on solution quality and CPU time

	Set X		Set XML	
Γ	Gap%	Time (s)	Gap%	Time (s)
5	4.664	0.157	2.976	0.024
10	4.018	0.160	2.365	0.026
15	3.817	0.162	2.194	0.027
20	3.667	0.165	2.137	0.029
30	3.630	0.197	2.087	0.034
50	3.696	0.238	2.089	0.043
100	3.690	0.375	2.087	0.057

The results of this experiment indicate that solution quality generally improves with Γ, but with decreasing marginal returns. We cease to see notable solution quality improvements once Γ exceeds a value of 30, but CPU time dramatically increases. Given this, we set $\Gamma = 15$ in the remainder of our experiments, and additionally provide detailed results with $\Gamma \in \{20, 30, 50\}$ in Appendices A and B.

3.3.5 Experimental Results – Set XML

Having calibrated all the algorithmic components, we can now measure the impact of GNN-informed relatedness measures in the LS and crossover operator. We focus here on the instances of set XML. Given that HGS converges towards near-optimal solutions within seconds for these instances, we use $T_{\text{MAX}} = 5$ seconds per instance and report final results as well as convergence plots to measure the impact of the different versions of the LS and crossover.

Table 3.3 therefore reports the number of optimal solutions (#Opt) attained over the 10,000 instances and the average final Gap(%) for all of the methods, considering the four possible GNN configurations (Original, Optimistic, Model #1, and Model #2). Best performance is indicated in boldface. Additionally, the convergence plots of Figure 3.4 depict the progress of the average gap of the different methods over time for the Optimistic configuration of the GNN, and similar graphs are provided for the other GNN configurations in Appendix A.

As seen in these results, all HGS versions decrease the gap in a smooth way within the time limit, and all approaches except HGS-N-O outperformed HGS-D-O (the baseline HGS algorithm) in terms of their number of optimal solutions and average gap. The significance of these improvements is confirmed by two-tailed paired-samples Wilcoxon tests between each of the methods and HGS-D-O at a significance level of 0.05.

Table 3.3: Results of all methods and GNN configurations for the instances of set XML

	HGS-D-O		HGS-D-D		HGS-D-N		HGS-N-O		HGS-N-D		HGS-N-N	
GNN	#Opt	Gap%	#Opt	Gap%	#Opt	Gap%	#Opt	Gap%	#Opt	Gap%	#Opt	Gap%
Original	7715	0.030	**8105**	0.024	8086	**0.023**	7691	0.031	8046	**0.023**	8011	0.025
Optimistic	7715	0.030	8105	0.024	**8120**	**0.023**	7732	0.031	8062	**0.023**	8041	0.025
Model #1	7715	0.030	**8105**	0.024	8094	**0.023**	7697	0.031	8028	0.024	7994	0.025
Model #2	7715	0.030	**8105**	**0.024**	8102	**0.024**	7719	0.030	8067	**0.024**	8057	**0.024**

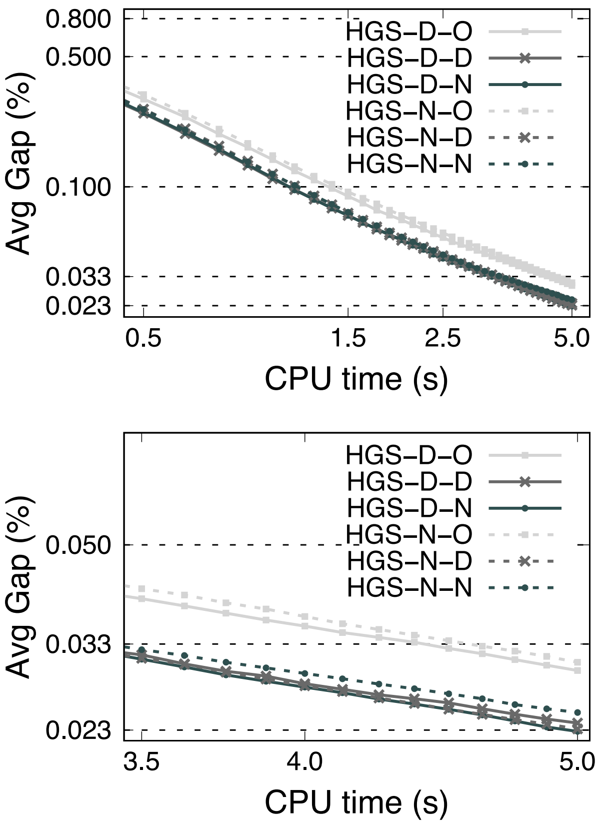


Figure 3.4: Convergence plots for all HGS variants on set XML (upper graph = complete run, lower graph = last 1.5 seconds)

However, these experiments also show that HGS-N-O does not perform significantly better than HGS-D-O, even when ignoring the inference time (i.e., Optimistic evaluation of the GNN). This indicates that the use of the GNN-based relatedness criterion in the LS does not bring significant benefits. It is an open research question to determine if different GNN architectures may perform better in the task of filtering LS neighborhoods.

Now, a comparison of configurations HGS-D-O (baseline), HGS-D-D, and HGS-D-N permits us to assess the impact of our changes on the crossover operator. We remind that HGS-D-O refers to the original OX crossover, whereas HGS-D-D and HGS-D-N modify the reconnection step to integrate relatedness information. As seen in our experiments, HGS-D-D and HGS-D-N are much better than the baseline (final gaps of 0.024% compared to

Table 3.4: Results of all methods and GNN configurations for the instances of set X

	HGS-D-O		HGS-D-D		HGS-D-N		HGS-N-O		HGS-N-D		HGS-N-N	
GNN	#Opt	Gap%	#Opt	Gap%	#Opt	Gap%	#Opt	Gap%	#Opt	Gap%	#Opt	Gap%
Original	**188**	0.368	187	**0.302**	184	0.317	177	0.395	169	0.325	172	0.317
Optimistic	**188**	0.368	187	0.302	187	**0.299**	185	0.365	177	0.307	182	**0.299**
Model #1	**188**	0.368	187	**0.302**	185	0.309	166	0.414	177	0.332	184	0.336
Model #2	**188**	0.368	187	0.302	186	**0.301**	169	0.391	175	0.314	170	0.316

0.030%), as confirmed by paired-samples Wilcoxon tests at 0.05 significance level. This is a notable breakthrough, given that it is uncommon to identify simple conceptual changes to HGS that significantly improve its state-of-the-art performance.

Finally, the choice of configuration for the GNN did not significantly affect the results, and our observations remain valid for the Original, Model #1, and Model #2 configurations.

3.3.6 Experimental Results – Set X

The instances of set X include a different number of customers, but the GNN of [146] is designed to predict heatmaps only for fixed-size graphs. Moreover, training a model on an instance of maximal size (1000 customers) and relying on dummy nodes is likely to require extensive training time (especially with its original parametrization).

To circumvent this issue, we instead propose to combine the heatmaps from different subproblems to obtain a relatedness measure for all customers. Let n_{G} be the graph size handled by the GNN. For each customer $i \in \{1, \ldots, n\}$, in turn, we collect the $n_{\text{G}} - 1$ closest customers along with the depot to form a CVRP subproblem with exactly n_{G} customers. We rely on the GNN to infer the heatmap for this graph, and use the heatmap values for all edges (i, j) such that j belongs to the subproblem and 0 otherwise. This simple approach requires n heatmaps inference steps, but the inherent parallelism of Pytorch makes it effective enough for our purposes.

As previously, we report the results of the different HGS variants in Table 3.4 for the four considered GNN parameter settings. We set a total computational time budget that is linearly proportional to n, allowing 24 seconds for the smallest instance (X-n101-k25) with 100 customers, and up to 240 seconds for the largest one (X-n1001-k43) with 1000 customers. Moreover, we perform 10 experiments with different random seeds for each of the 100 instances, leading to 1000 solution processes. The table, therefore, counts the number of optimal solutions out of 1000 as well as the average error gap when the algorithm terminates. With these time limits, the inference time of the ORIGINAL GNN represents 15.2% of the overall time budget, and the inference time of MODEL #1 and MODEL #2 is limited to 1.9% of the time budget. Convergence plots in the same format as before are additionally presented in Figure 3.5, including for the other GNN configurations in Appendix B.

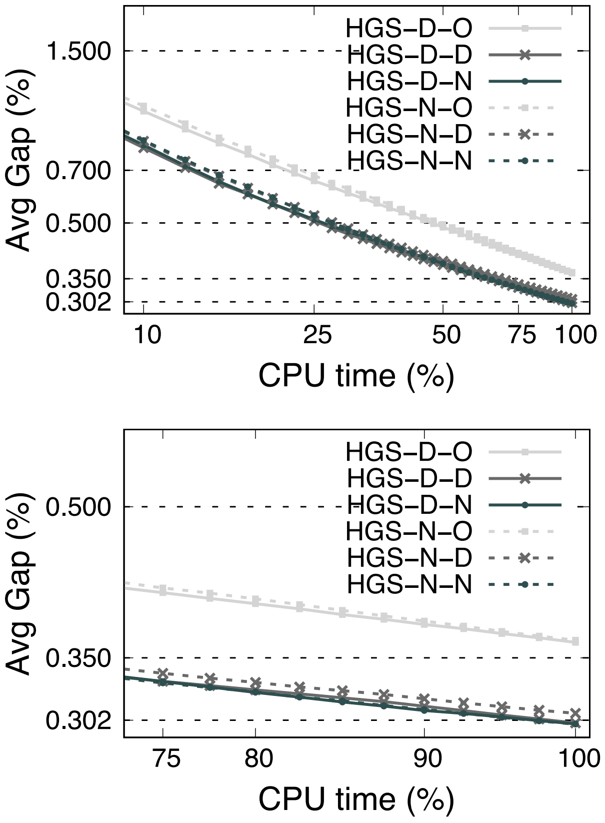


Figure 3.5: Convergence plots for all HGS variants on set X

These additional results on Set X confirm our previous observations: all HGS variants except HGS-N-O outperformed the HGS-D-O baseline. Additionally, the proposed modifications to the crossover operator (HGS-D-D and HGS-D-N) led to performance improvements that are even more expressive on that instance set, with final gaps of 0.302% and 0.299% compared to 0.368% for the original HGS. As previously, however, the use of learned information from the GNN instead of distance did not make a substantial difference in the crossover and even appeared to be detrimental in the context of the LS.

It is important to stress that, without a complete analysis involving HGS-D-D, a comparison of HGS-D-N versus HGS-D-O could have led to the conclusion that the GNN was responsible for the improvement. However, recommending the use of this method in this context would have been an "overkill" since a simpler reconnection mechanism based on distance effectively produces the same gains.

3.4 Conclusions

In this study, we have shown that relatedness metrics can be broadly used to improve the performance of the HGS [136], a state-of-the-art solution algorithm for the CVRP. Relatedness has been exploited in two ways: to focus the LS on promising moves, and to steer the crossover operator towards meaningful reconnections. As relatedness is a fairly general concept, we can freely use geographical or learnable (i.e., GNN-based) information for that purpose. As seen in our experimental analyses, these adaptations lead to significant improvements on a large benchmark counting over 10,000 instances. Additionally, we show that a simple strategy to extend GNN heatmap predictions to instances of varying size is fairly effective, circumventing the limitation due to fixed-size training. Overall, exploiting heatmaps to boost HGS operators is very effective, but also not superior to a simpler application of distance-based relatedness for similar purposes. This observation comes in contrast with the superiority claims of sophisticated learning mechanisms and ever-larger networks. Instead, it aligns with the “less-is-more” approach toward algorithmic design.

We acknowledge that some aspects studied in this study can be further investigated for future research. The first one refers to the applications of relatedness criteria to other combinatorial optimization settings and solvers (e.g., branch and bound). Another research avenue of interest concerns exploiting different relatedness sources and simpler machine learning models. Finally, from a more general viewpoint, we expect that the contributions of this study can lead to a better comprehension of the challenges involved in incorporating sophisticated machine learning techniques into state-of-the-art solvers. We believe that the general research direction towards GNN-enhanced heuristics is promising. However, careful ablation studies are critically needed to correctly identify improvement sources and measure the true contribution of learned information.

4
Support Vector Machines with the Hard-Margin Loss: Optimal Training via Combinatorial Benders' Cuts

4.1
Introduction

Support vector machines (SVMs) are among the most popular classification models due to their simplicity and solid theoretical foundations from statistical learning [79, 80, 81]. Application fields of SVMs include, among others, image classification, bioinformatics, handwritten digits recognition, face detection, and generalized predictive control [82, 83, 84]. Beyond this, SVMs are regularly used as elementary building blocks of sophisticated AutoML pipelines [85]. They achieve state-of-the-art results for a variety of applications, especially for large-scale datasets [86, 87, 88].

In its simplest form, an SVM can be defined as follows. Let $(\mathbf{X}, \mathbf{y}) = \{\mathbf{x}_i, y_i\}$ be a training set in which each $\mathbf{x}_i \in \mathbb{R}^m$ is an m-dimensional feature vector, and each $y_i \in \{-1, 1\}$ is its associated class. Then, an SVM seeks a hyperplane $\mathcal{H} = \{\mathbf{x} \in \mathbb{R}^m : \mathbf{w} \cdot \mathbf{x} + b = 0\}$ that optimizes the following objective:

$$\min_{\mathbf{w},b} \frac{1}{2}||\mathbf{w}||^2 + C\sum_{i=1}^{n} f(y_i(\mathbf{w} \cdot \mathbf{x}_i + b)). \tag{4-1}$$

The first term of Equation (4-1) acts as a regularization term and indirectly maximizes the margin of the SVM [80], whereas the second term ensures fidelity to the data and penalizes misclassified samples, with coefficient C balancing the two terms. Accordingly, the objective establishes a trade-off between maximizing the hyperplane's margin and minimizing the concomitant misclassification error. The loss function f varies with respect to the studied problem variant. In the classical SVM with hinge loss (SVM-HL), we define

$$f(u) := f_{\text{Hinge}}(u) = \max\{0, 1-u\}, \tag{4-2}$$

such that a misclassified sample, i.e., a sample for which $y_i(\mathbf{w} \cdot \mathbf{x}_i + b) < 1$, directly increases the objective value of Equation (4-1) proportionally to its error (see Figure 4.1).

However, while the classical convex SVM-HL permits fast training and scalability, it is also known to lack robustness in the presence of misclassified samples or outliers since its loss function is unbounded. As a drawback, the trained model can be severely affected by these outliers, preventing its application in several domains, especially for high-stakes decisions where robustness is critical [89].

In light of this, some works have considered the use of bounded but non-convex loss functions in an attempt to gain robustness (see e.g., [98, 106, 129]). In particular, the SVM with hard-margin loss (SVM-HML) uses

$$f(u) := f_{\text{HARD}}(u) = \begin{cases} 1 & \text{if } u < 1 \\ 0 & \text{otherwise,} \end{cases} \tag{4-3}$$

such that any misclassified sample (or sample within the margin) leads to a unit penalty (see Figure 4.2), therefore limiting the influence of misclassified samples on $\mathcal{H}$ and increasing classification robustness.

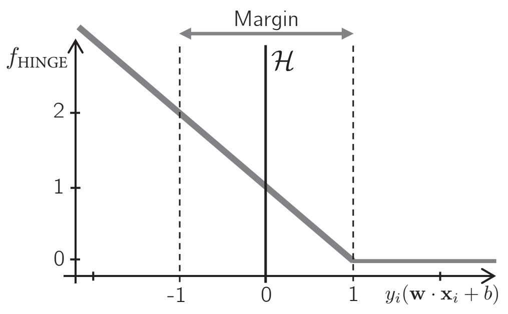


Figure 4.1: Hinge loss function

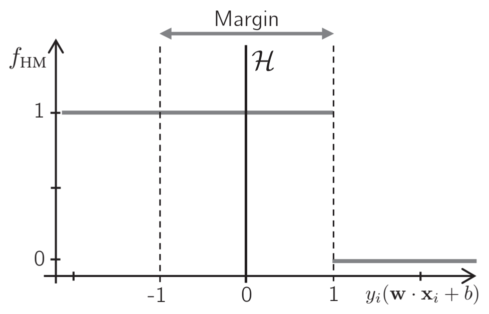


Figure 4.2: Hard-margin loss function

However, this much-needed robustness comes at the expense of computational efficiency since training SVM-HML requires solving a mixed-integer quadratic problem (MIQP) and is NP-hard [90]. This hardness, but more especially the inability to efficiently solve the SVM-HML, limits its current use. Training SVM-HML models to global optimality is currently only achievable for very small datasets, whereas current heuristics for training do not consistently find high-quality hyperplanes. It is noteworthy to highlight that SVM-HML is different from the classical *hard-margin* SVM, where the former admits misclassified samples over a fixed penalty factor while the latter becomes infeasible in the presence of non-separable data.

In this study, we contribute towards addressing those challenges and pave the way toward more efficient global optimization algorithms. We propose new mixed-integer programming approaches to train SVM-HML models to global optimality. We exploit the problem's structure to devise efficient decomposition techniques, relying on subsets of the samples to generate Combinatorial

Benders' (CB) cuts quickly. More specifically, the contribution of this study is fourfold:

– We show that CB cuts can be successfully exploited to generate useful inequalities for SVM-HML.

– We introduce sampling strategies that permit to quickly generate a diversified pool of cuts. We effectively embed these cuts within a branch-and-cut algorithm, leading to an efficient training algorithm that can achieve global optimality.

– We conduct an extensive numerical campaign to measure the performance of our training approach and the impact of important design choices. As seen in our experiments, this algorithm significantly improves the current status-quo regarding the solution of SVM-HML, solving for the first time 117 new data sets to optimality and achieving a reduction of 50% in the average optimality gap over previous approaches for the hardest datasets of the benchmark.

– Generally, our study underlines the benefits of applying cutting-edge mixed-integer programming techniques to combinatorial optimization problems that arise when training non-convex machine learning models.

The remainder of this study is organized as follows. Section 4.2 briefly reviews the related literature. Section 4.3 introduces the proposed methodology. Section 4.4 details our computational experiments, and Section 4.5 concludes this study.

4.2 Related Literature

A vast body of literature on SVMs exists, covering various topics such as applications, training algorithms, and loss functions. For the sake of brevity, we focus on recent contributions to training algorithms for SVM-HL as well as works on SVMs with non-convex loss functions, namely SVM-HML and the SVM with ramp loss (SVM-RL).

The training problem for the classical SVM-HL can be cast and efficiently solved as a continuous convex quadratic programming problem. Existing solution approaches typically detect and fix violations of first-order optimality conditions, leading to a series of small subproblems with few variables. A classical approach, called Sequential Minimal Optimization (SMO) is used in several state-of-the-art implementations [86, 87, 88, 91] and consists of solving a restricted problem of only two variables at each iteration. Still, some algorithms have also exploited larger subproblems (see e.g., [130, 88, 131]).

Training algorithms for SVM-HL are quite diverse and base on data-selection concepts [92, 93], geometric methods [79] and heuristics [94]. For a detailed presentation of algorithms for the SVM-HL and its variants, we refer the reader to the surveys of [95, 83], and [96], as well as to the book of [97].

Despite its widespread use, the sensitivity of SVM-HL to outliers has regularly raised obstacles when dealing with critical applications. Consequently, a part of recent research has explored the possibility of using non-convex loss functions to gain robustness. [98] focused on two non-convex loss functions in particular: the SVM-HML [99, 100, 101] and the SVM-RL [102, 103, 104]. Both of these functions are bounded, such that the contribution of each sample to the objective is limited. In the SVM-RL model, any sample within the margin receives a linear penalty proportional to its distance to the margin (a value between 0 and $2C$), whereas any misclassified sample outside the margin receives a fixed penalty of $2C$. In the SVM-HML, the penalty of any misclassified or within-margin sample is simply fixed to a constant C.

The solution algorithm proposed by [98] solves the training problem as a mixed-integer quadratic programming (MIQP) using state-of-the-art branch-and-cut solvers. For the SVM-HML and SVM-RL, the authors rely on indicator constraints representing logical implications between a binary variable representing the status of each sample (misclassified or not) and a linear constraint that evaluates its relative position from the separating hyperplane. However, it is well known that such constraints can be reformulated in linear form using a "big-M" constant, but doing so without carefully tuning the value of the M constant typically leads to an ineffective formulation with a weak linear relaxation, impeding an efficient solution by branch-and-cut [105].

For the SVM-RL setting, [106] explored the ramp loss function with ℓ_1-penalty, resulting in a piecewise linear programming problem. Later, [107] compared different formulations for the logical constraints and concluded that aggressive bound-tightening techniques are necessary for a successful solution approach. The strategy derived from their studies has been since implemented as a standard routine for handling such constraints in the commercial solver CPLEX for MILP/MIQPs. Finally, [108] tightened the M constants by solving sequences of continuous problems and Lagrangian relaxations. For the SVM-HML, [109] proposed hard-margin loss formulations within the context of multiple-instances classification, a setting in which class labels are defined as sets. Finally, in [110], the hard-margin loss was transformed into a re-scaled hinge loss function for imbalanced noisy classification.

Concluding, only a few studies have attempted to improve the state-of-the-art training algorithms for SVM-HML after the seminal work of [98], often by concentrating on the handling of the logical constraints and the proper calibration of the M constants. Despite this progress, optimal training remains limited to data sets counting a few hundred samples. To improve this status-quo, we investigate a different approach, which consists of the separation of CB cuts and their combination with classical logical constraints to achieve a valid problem formulation with a better linear relaxation. Moreover, to improve computational efficiency, we rely on sampling techniques for a fast generation of diverse cuts.

4.3 Methodology

We first recall the classical mathematical programming formulations for the SVM variants that are of interest for our study, namely SVM-HL and SVM-HML, and then proceed with a description of our algorithmic approach.

4.3.1 Descriptive formulations

The *SVM with hinge loss* (also known as *soft-margin* SVM) associates to misclassified and within-margin samples a penalty that is proportional to their distance in relation to the "correctly classified margin" of the found hyperplane (see Figure 4.1).

Let $\mathbf{w} \in \mathbb{R}^m$ be a vector of real variables that represents the coordinates of the hyperplane, let b be its intercept to the origin, and let ξ_i represent the misclassification penalty of sample i. With these notations, the training problem for SVM-HL can be mathematically formulated as:

$$\min_{\mathbf{w},b,\xi} \quad \frac{1}{2}||\mathbf{w}||^2 + C\sum_{i=1}^{n}\xi_i \tag{4-4}$$

$$\text{s.t.} \quad y_i(\mathbf{w}\cdot\mathbf{x}_i + b) \geq 1 - \xi_i, \quad i \in \{1,\ldots,n\} \tag{4-5}$$

$$\xi_i \geq 0, \quad i \in \{1,\ldots,n\}. \tag{4-6}$$

Objective (4-4) seeks a maximum margin separator through the term $\frac{1}{2}||\mathbf{w}||^2$ and minimizes the total misclassification penalty $C\sum_{i=1}^{n}\xi_i$, where $C > 0$ is a constant that balances both parts of the objective. Moreover, Constraints (4-5) calculate the misclassification penalty of each sample.

In the *SVM with hard-margin loss*, misclassified samples are associated a fixed penalty cost. Binary variables z are used to indicate whether a sample i is misclassified or within the margin ($z_i = 1$), or correctly classified ($z_i = 0$).

The SVM-HML can be formulated as follows:

$$\min_{\mathbf{w},b,z} \quad \frac{1}{2}||\mathbf{w}||^2 + C\sum_{i=1}^{n} z_i \tag{4-7}$$

$$\text{s.t.} \quad (z_i = 0) \Rightarrow y_i(\mathbf{w}\cdot\mathbf{x}_i + b) \geq 1 \qquad i \in \{1,\dots,n\} \tag{4-8}$$

$$z_i \in \{0,1\} \qquad i \in \{1,\dots,n\}. \tag{4-9}$$

In this formulation, the penalty term in the objective (4-7) is directly proportional to the number of misclassified samples. Constraints (4-8), represented as logical constraints, ensure that $z_i = 0$ only if sample i is correctly classified. On the use of indicator constraints, the solver is able to branch explicitly on these constraints or to derive tighter big-M values to make the formulation more stable. Namely, these constraints are strictly speaking not part of the semantic of a MILP/MIQP, but they could be directly transformed into a linear constraint by using a big-M constant and imposing $y_i(\mathbf{w}\cdot\mathbf{x}_i+b) \geq 1-Mz_i$. The drawback of such a reformulation is that it leads to a formulation that provides notably weaker linear-relaxation bounds, rendering branch-and-bound algorithms relatively inefficient.

4.3.2 Combinatorial Benders' cuts

To optimally solve the SVM-HML, we propose a solution method based on Benders' decomposition [111]. In its canonical form, this strategy exploits the structure of a mixed-integer linear program and splits its variables into two subsets. The first subset of integer or continuous variables (sometimes called "complicating variables") is selected in such a way that fixing them either decomposes or reduces the complexity of the resulting problem. The remaining variables should be continuous. The method then works by decomposing the original problem into a master problem (MP), solved over the complicating variables, and a subproblem (SP), solved as a linear program over the remaining continuous variables. The algorithm iteratively produces an incumbent solution of the MP and uses the dual of the SP to assess its feasibility. If the SP determines that the incumbent solution is feasible, then this information is integrated into the MP in the form of an additional CB cut, which eliminates the infeasible incumbent solution.

The studies of [112] and [113] paved the way towards efficient applications of Benders' decomposition to a broad class of mixed-integer programming problems (MIPs) with logical constraints. Notably, the so-called CB approach leads among others to a new solution paradigm for models of the following form:

$$\begin{aligned} \min \quad & \mathbf{c}^\intercal \mathbf{z} && && (4\text{-}10)\\ \text{s.t.} \quad & (z_i = 0) \Rightarrow \mathbf{a_i}^\intercal \mathbf{y} \ge d_i && i \in \{1,\dots,n\} && (4\text{-}11)\\ & \mathbf{z} \in \{0,1\}^n && && (4\text{-}12)\\ & \mathbf{y} \in Y, && && (4\text{-}13) \end{aligned}$$

with binary complicating variables $\mathbf{z}$, continuous variables $\mathbf{y}$ in a polytope Y (i.e., respecting a set of linear inequalities), and linear weights $\mathbf{c}$ (applied only on the coefficients of $\mathbf{z}$) to calculate the objective. In a CB approach, Model (4-10–4-13) is reformulated as follows:

$$\begin{aligned} \min \quad & \mathbf{c}^\intercal \mathbf{z} && && (4\text{-}14)\\ \text{s.t.} \quad & \sum_{i \in S} z_i \ge 1 && S \in \mathcal{S}^{\text{MIS}} && (4\text{-}15)\\ & \mathbf{z} \in \{0,1\}^n, && && (4\text{-}16) \end{aligned}$$

where $\mathcal{S}^{\text{MIS}}$ is the collection of all inclusion-minimal infeasible subsystems (MIS) of rows:

$$\mathcal{S}^{\text{MIS}} = \left\{ S \;\middle|\; \begin{array}{l} \{\mathbf{a_i}^\intercal \mathbf{y} \ge d_i \; \forall i \in S, \mathbf{y} \in Y\} \text{ is infeasible} \\ \{\mathbf{a_i}^\intercal \mathbf{y} \ge d_i \; \forall i \in \hat{S}, \mathbf{y} \in Y\} \text{ is feasible } \forall \hat{S} \subset S \end{array} \right\}. \qquad (4\text{-}17)$$

Notably, this formulation no longer contains logical implications or big-M terms. However, it includes an exponential number of constraints. For this reason, the CB approach consists of dynamically detecting and adding Constraints (4-15). As in a classical Benders' decomposition, the method alternates between solving the master problem with only a subset of the constraints $\mathcal{S} \subset \mathcal{S}^{\text{MIS}}$ found so far, and solving a subproblem to identify new violated constraints that cut the incumbent solution of the master in case of infeasibility. Namely, Each MIS corresponds to a subsystem of inequalities and equations such that the subsystem is inconsistent and every proper subsystem is consistent (i.e., feasible). Finally, we observe that $\mathcal{S}^{\text{MIS}}$ does not need to be restricted to "inclusion-minimal" subsets of rows to yield a valid formulation, but doing so significantly reduces the number of constraints in the set.

Despite its successful application in a variety of settings [114, 115, 116], the aforementioned CB framework is not directly applicable to problems with objective functions that contain both complicating variables $\mathbf{z}$ and continuous variables $\mathbf{y}$. This is unfortunately the case for the SVM-HML, due to the occurrence of the continuous variables $\mathbf{w}$ and b in the objective and the logical implications. To circumvent this issue, we introduce a weaker set of CB cuts in conjunction with the original logical implication constraints to design an effective solution method. In this case, the cuts do not carry the burden of ensuring the model's validity, but nevertheless contribute to strengthening the formulation to obtain a better linear relaxation and improve the solution process.

4.3.3 Combinatorial Benders' cuts and the SVM-HML

We first start by describing a direct application of CB to the SVM-HML and by analyzing its shortcomings. This would lead to the following formulation:

$$\min_{W,\mathbf{z}} \quad \frac{1}{2}W^2 + C\sum_{i=1}^{n} z_i \tag{4-18}$$

$$\text{s.t.} \quad \sum_{i\in S} z_i \geq 1 \qquad S \in \mathcal{S}^{\text{MIS}}_{\text{SVM-HML}}(W) \tag{4-19}$$

$$\mathbf{z} \in \{0,1\}^n \tag{4-20}$$

$$\text{with } \mathcal{S}^{\text{MIS}}_{\text{SVM-HML}}(W) = \left\{ S \;\middle|\; \begin{array}{l} \{y_i(\mathbf{w}\cdot\mathbf{x}_i + b) \geq 1 \ \forall i \in S,\ ||w||^2 \leq W^2\} \text{ is infeasible} \\ \{y_i(\mathbf{w}\cdot\mathbf{x}_i + b) \geq 1 \ \forall i \in \hat{S},\ ||w||^2 \leq W^2\} \text{ is feasible } \forall \hat{S} \subset S \end{array} \right\}. \tag{4-21}$$

As seen in this formulation, W appears in the objective and also characterizes the set of Benders' cuts. Unfortunately, the resulting formulation can no longer be practically solved as a MILP or MIQP due to this dependency. To remedy this issue, we leverage Property 1.

Property 1 Let W_{UB} be a valid upper bound on W on any optimal solution of Problem (4-18–4-21). Then, $\mathcal{S}^{\text{MIS}}_{\text{SVM-HML}}(W_{\text{UB}})$ is also set of valid inequalities for the SVM-HML.

Proof. Consider $S \in \mathcal{S}^{\text{MIS}}_{\text{SVM-HML}}(W_{\text{UB}})$. Due to the definition of $\mathcal{S}^{\text{MIS}}_{\text{SVM-HML}}(W_{\text{UB}})$, $\{y_i(\mathbf{w}\cdot\mathbf{x}_i + b) \geq 1 \ \forall i \in S$ with $||w||^2 \leq W^2_{\text{UB}}\}$ is an infeasible subsystem. Given that all optimal solutions of Problem (4-18–4-21) satisfy $W \leq W_{\text{UB}}$, then $\{y_i(\mathbf{w}\cdot\mathbf{x}_i + b) \geq 1 \ \forall i \in S$ with $||w||^2 \leq W^2\}$ is also an infeasible subsystem, and therefore at least one sample i in S must be misclassified, implying that $\sum_{i\in S} z_i \geq 1$ is a valid inequality.

With this property, the dependence upon parameter W can be avoided as soon as a valid upper bound is known. The quality of the upper bound also impacts the strength of the valid inequalities obtained. Given an initial feasible solution of the SVM-HML problem with value Φ_{UB} (obtained, for example, with a heuristic for this problem), we can use $W_{\text{UB}} = 2\sqrt{\Phi_{\text{UB}}}$ since the two terms of the objective are positive.

Finally, we opted to further relax the CB cuts by using $-W_{\text{UB}} \leq w_j \leq W_{\text{UB}}$ for $j \in \{1,\dots,m\}$ instead of $||w||^2 \leq W_{\text{UB}}^2$ in Equation (4-21). Our experimental analyses have shown that this permitted a faster cut separation with only a limited impact on the strength of the formulation. Overall, we will use the resulting valid inequalities in combination with the original formulation and the tightened bounds on the w_j coefficients, leading to the following model:

$$\min_{\mathbf{w},\mathbf{z}} \quad \frac{1}{2}||\mathbf{w}||^2 + C\sum_{i=1}^{n} z_i \tag{4-22}$$

$$\text{s.t.} \quad (z_i = 0) \Rightarrow y_i(\mathbf{w}\cdot\mathbf{x}_i + b) \geq 1 \qquad i \in \{1,\dots,n\} \tag{4-23}$$

$$-W_{\text{UB}} \leq w_j \leq W_{\text{UB}} \qquad j \in \{1,\dots,m\} \tag{4-24}$$

$$\sum_{i\in S} z_i \geq 1 \qquad S \in \mathcal{S}_{\text{SVM-HML}}^{\text{MIS-UB}} \tag{4-25}$$

$$\mathbf{z} \in \{0,1\}^n \tag{4-26}$$

with $\mathcal{S}_{\text{SVM-HML}}^{\text{MIS-UB}} =$

$$\left\{ S \;\middle|\; \begin{array}{l} \{y_i(\mathbf{w}\cdot\mathbf{x}_i + b) \geq 1 \ \forall i \in S, \ -W_{\text{UB}} \leq w_j \leq W_{\text{UB}} \ \forall j\} \text{ is infeasible} \\ \{y_i(\mathbf{w}\cdot\mathbf{x}_i + b) \geq 1 \ \forall i \in \hat{S}, \ -W_{\text{UB}} \leq w_j \leq W_{\text{UB}} \ \forall j\} \text{ is feasible } \forall \hat{S} \subset S \end{array} \right\}. \tag{4-27}$$

With this problem formulation in mind, we will focus on our general solution approach and the separation of the CB cuts in the following.

4.3.4 General Solution Approach

Our solution approach unfolds in three steps:

Step 1. Finding an initial upper bound W_{UB};

Step 2. Solving a simplified formulation by branch-and-cut to obtain a cut set $\mathcal{S}_{\text{SVM-HML}}^{\text{MIS-UB}}$;

Step 3. Solving Problem (4-22–4-27) with the set of cuts identified in the previous step.

We note that the generation of the cuts is done in a separate phase (Step 2). Our computational experiments have shown that it is more effective to generate a set of cuts beforehand, and then allow the MILP solver (CPLEX) to use its default settings when solving the complete model with these cuts, instead of dynamically providing additional cuts as the search progresses. We now proceed with a detailed description of each step of the algorithm.

Step 1 – Initial bound. We start by solving an SVM with hinge loss, cast as a continuous quadratic program through Equations (4-4–4-6), and collect the resulting value of the variables $(\mathbf{w}', b', \xi')$ defining the hyperplane. With this hyperplane, we obtain an associated SVM-HML solution by setting $z_i = \lceil \xi_i \rceil$ and calculate the resulting W_{UB} objective value.

Step 2 – Separation of the Combinatorial Benders' cuts. Next, we consider Problem (4-22–4-27) excluding the logical Constraints (4-23) while dynamically generating Constraints (4-25). After excluding the logical constraints, $||\mathbf{w}||^2$ is free to take a value of $\mathbf{0}$, and therefore the first term of the objective vanishes. The resulting problem is a variant of the linear separator problem [117, 80], which we will only use to generate CB cuts for the subsequent solution of the SVM-HML. To obtain the cuts, we apply a branch-and-cut scheme on the following master problem:

$$\textbf{Master:} \quad \min_{\mathbf{z}} \sum_{i=1}^{n} z_i \tag{4-28}$$

$$\sum_{i \in S} z_i \geq 1 \qquad S \in \mathcal{S} \tag{4-29}$$

$$\mathbf{z} \in \{0,1\}^n. \tag{4-30}$$

At each branching node, the linear relaxation of the master problem is solved and the set $\mathcal{I}^0$ of variables with values $z_i = 0$ are identified. Based on this set of variables, we define the following feasibility subproblem to check if a violated CB cut exists:

$$\textbf{Subproblem:} \quad y_i(\mathbf{w} \cdot \mathbf{x}_i + b) \geq 1 \qquad i \in \mathcal{I}^0 \tag{4-31}$$

$$-W_{\text{UB}} \leq w_j \leq W_{\text{UB}} \qquad j \in \{1, \dots, m\} \tag{4-32}$$

If this subproblem admits a feasible solution, then no CB cut needs to be added. Otherwise, the subproblem is infeasible and we can obtain an inclusion-minimal infeasible subsystem (MIS) of indices from CPLEX, giving us a new set of variables $\bar{\mathcal{I}}$ which can be added to $\mathcal{S}$. As seen in Algorithm 6, this process is repeated at each node of the branch-and-cut tree until no additional cut can be found.

To avoid spending excessive time on the solution of this auxiliary problem and to generate a diversified set of CB cuts, we rely on sampling strategies. Until a maximum time limit of T_{S}, we randomly select subsets

Algorithm 6: Generation of Combinatorial Benders' cuts on a given node

```
1  $\mathcal{S} \leftarrow \emptyset$
2  repeat
3      $\mathbf{z} \leftarrow$ Solve the linear relaxation of Problem (4-28–4-30) with $\mathcal{S}$
4      $\mathcal{I}^0 \leftarrow$ Indices such that $z_i = 0$
5      if Problem (4-31–4-32) with $\mathcal{I}^0$ admits a feasible solution then
6          break
7      else
8          $\bar{\mathcal{I}} \leftarrow$ MIS from Problem (4-31–4-32)
9          $\mathcal{S} \leftarrow \mathcal{S} \cup \bar{\mathcal{I}}$
10 until
```

of $r = \min\{n/2, 50\}$ samples, and apply the aforementioned branch-and-cut and cut separation methodology. Only afterward we repeat this process in a final run, considering all the samples and hot starting with all the CB cuts already found. We stop the cut separation as soon as a time limit T_{B} is attained.

Step 3 – Solution of the SVM-HML. Finally, we proceed with solving the complete Problem (4-22–4-27), using the union of the CB cuts that have been found in Step 2. These cuts are included as lazy constraints to avoid any overhead due to the number of cuts. Furthermore, we warm start from the solution found in Step 1, and use the default settings of CPLEX, except for the *locally valid implied bounds* parameter, which is set to *very aggressively* as suggested in [107]. The solution approach is run until it proves optimality or reaches a maximum time limit of T_{MAX}. Note that this time limit encompasses all the steps of our method, such that it is possible to limit the total computational time and compare our method with other algorithms.

4.4 Computational Experiments

We conduct extensive experimental analyses to evaluate the performance of the proposed approach, denoted as CB-SVM-HML, on a diverse set of benchmark instances. As an experimental baseline for this comparison, we rely on a reimplementation of the branch-and-cut approach of [98]. The goal of our experiments is (i) to compare the performance of these algorithms in terms of computational time and optimality gaps, and (ii) to evaluate the impact of the CB cuts and of the proposed search strategies based on sampling.

4.4.1
Data and Experimental Setup

To evaluate the performance of our algorithm, we use the same benchmark as in [98], divided into three sets: UCI, Type A, and Type B. The UCI set consists of 11 heterogeneous datasets from the UCI machine learning repository [118], which were preprocessed by [98]. Table 4.1 lists the number of samples (n) and features (m) of these datasets. Type-A and Type-B datasets have been constructed by [98] using simulated data with a controlled number of outliers. Type-A datasets contain outliers that are clustered together, and generally distant from the rest of the data points. In contrast, Type-B datasets contain outliers that are more evenly distributed. Type-A and Type-B sets both include 12 datasets with different number of samples $n = \{60, 100, 200, 500\}$ and features $m = \{2, 5, 10\}$. Finally, for all of the considered benchmark datasets, [98] obtained five different instances with different random data points. Overall, this gives us $(11 + 12 + 12) \times 5 = 175$ instances to evaluate SVM-HML solution methods. Additionally, for each of these instances, we will consider $C \in \{1, 10, 100, 1000, 10000\}$ for the penalty factor as done in [98], giving a total of 875 instances for each algorithm.

Name	n	m
adult	400	77
australian	366	45
breast	383	9
bupa	193	6
german	400	24
heart	152	20
ionosphere	196	33
pima	400	8
sonar	116	60
wdbc	319	30
wpbc	108	30

Table 4.1: Characteristics of the UCI datasets

All the algorithms considered in this study have been implemented in C++ and use the CPLEX 12.9 callable library. The experiments were run on an Intel Xeon E5-2620 2.1 GHz processor machine with 128 GB of RAM and CentOS Linux 7 (Core) operating system. All the source code and scripts needed to reproduce these experiments are provided at `https://github.com/vidalt/Hard-Margin-SVM`.

4.4.2 Performance of CB-SVM-HML

In the first set of experiments, we compare the results of the proposed CB-SVM-HML with those of the baseline algorithm of [98]. We use the same experimental conditions, and therefore run each algorithm until a time limit of $T_{\text{MAX}} = 600$ seconds for each instance and value of C. The time dedicated to the separation of CB cuts in CB-SVM-HML is set to $T_{\text{B}} = T_{\text{S}} = 30$ seconds. In other words, 5% of the total time is dedicated to the separation of CB on subsets of samples, and 5% of the time on the complete problem. As shown in our sensitivity analyses in Section 4.4.3, this amount of time already permits to separate a large number of cuts. Allocating more computational time for cut separation did not further improve the overall search process.

Tables 4.2 to 4.6 report, for each algorithm over all C values, the number of instances solved to optimality (# Opts), the average optimality gap (Gap (%)), and the average computational time in seconds (Avg Time). In the last line, Overall provides the sum of all values for columns # Opts, and the average of all values for columns Gap (%) and Avg Time. In Tables 4.3 and 4.4, the results of the instance of Type A and Type B are aggregated per number of samples $n \in \{60, 100, 200, 500\}$, whereas in Tables 4.5 and 4.6, the results are aggregated according to the dimension of the feature space $m \in \{2, 5, 10\}$. The detailed results for each instance are additionally provided in the same repository as the source code.

	Baseline [98]			CB-SVM-HML		
	# Opts	Gap (%)	Avg Time	# Opts	Gap (%)	Avg Time
Overall	149/275	33.32	295.31	152/275	20.70	280.17

Table 4.2: Performance comparison on the UCI instances

	[98]			CB-SVM-HML		
n	# Opts	Gap (%)	Avg Time	# Opts	Gap (%)	Avg Time
60	75/75	0.00	2.87	75/75	0.00	32.83
100	42/75	11.66	325.71	61/75	4.26	170.76
200	4/75	56.83	582.13	32/75	18.21	395.30
500	0/75	88.17	600.00	11/75	36.05	542.28
Overall	121/300	39.16	377.33	179/300	14.63	285.29

Table 4.3: Performance comparison on the Type-A instances – grouped by number of samples n

As seen in these experiments, CB-SVM-HML generally achieves better performance than the baseline algorithm of [98]. In general, it solves more

	[98]			CB-SVM-HML		
n	# Opts	Gap (%)	Avg Time	# Opts	Gap (%)	Avg Time
60	74/75	0.28	72.01	73/75	0.45	63.87
100	26/75	26.63	404.67	49/75	8.55	266.52
200	1/75	65.59	595.16	24/75	22.51	437.02
500	0/75	90.76	600.00	11/75	40.01	529.61
Overall	101/300	45.82	417.61	157/300	17.88	324.26

Table 4.4: Performance comparison on the Type-B instances – grouped by number of samples n

	[98]			CB-SVM-HML		
m	# Opts	Gap (%)	Avg Time	# Opts	Gap (%)	Avg Time
2	54/100	28.15	297.02	86/100	1.84	145.27
5	40/100	40.37	388.24	52/100	13.87	322.75
10	27/100	48.98	446.72	41/100	28.18	387.85
Overall	121/300	39.16	377.33	179/300	14.63	285.29

Table 4.5: Performance comparison on the Type-A instances – grouped by number of features m

	[98]			CB-SVM-HML		
m	# Opts	Gap (%)	Avg Time	# Opts	Gap (%)	Avg Time
2	51/100	32.14	305.96	83/100	1.60	148.83
5	26/100	48.06	455.49	47/100	16.13	358.51
10	24/100	57.26	491.38	27/100	35.90	465.42
Overall	101/300	45.82	417.61	157/300	17.88	324.26

Table 4.6: Performance comparison on the Type-B instances – grouped by number of features m

instances to optimality with the same time limit (488/875 compared to 371/875) and achieves smaller optimality gaps (17.65% on average compared to 39.61%). CB-SVM-HML also found optimal solutions for some instances with 500 samples. Instances of this size could not be solved to proven optimality by previous approaches. Observing the results for instances with a different number of features m, we see that CB-SVM-HML performs especially well on low-dimensional problems (i.e. when $m \in \{2, 5\}$) since MIS are generally smaller in this regime, leading to stronger CB cuts involving fewer variables. Generally, our method improved upon the baseline for all values of m.

In terms of computational time, CB-SVM-HML achieves optimality or attains smaller gaps faster than the baseline algorithm on all instances except those with $n = 60$ samples. For those small instances, the difference of performance comes from our parametrization choices: the current cut-separation algorithm uses at least 5% of the time (30 seconds) separating cuts on randomly-generated subproblems including different subsets of the samples. As a consequence, the method's computational time is bounded below by 30 seconds, whereas the baseline algorithm sometimes solves the complete problem in shorter time for very small instances. One way to reduce computational effort in those cases could involve using a variable separation time that depends on the size of the instance, or setting a limit on the number of subproblems considered in the sampling phase.

We complete this analysis in Figure 4.3 with a fine-grained study of the optimality gaps of the methods as a function of the number of samples $n \in \{60, 100, 200, 500\}$ in Type-A and Type-B instances. For each value of n and for each method, we represent the optimality gaps (over the 75 instances) as a boxplot. The boxes indicate the first and third quartile, and the whiskers extend to 1.5 times the interquartile range. Outliers that extend beyond this range are depicted as small dots.

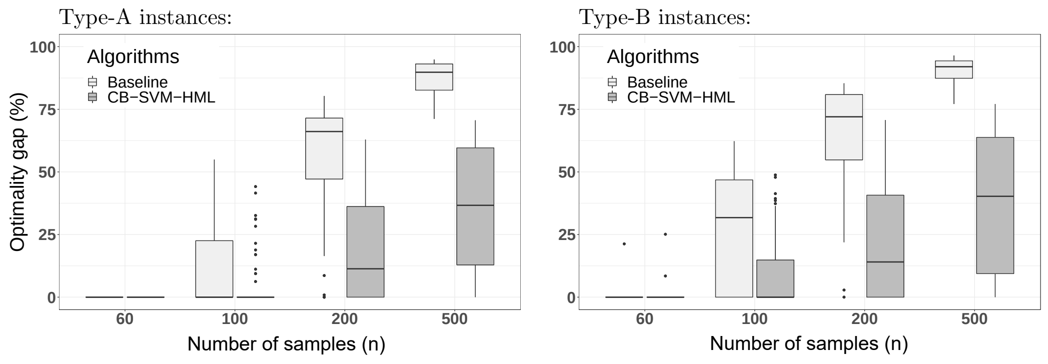


Figure 4.3: Optimality gaps on Type-A and Type-B instances as a function of the number of samples

As can be seen, CB-SVM-HML achieves better optimality gaps than the baseline algorithm for all values of n. Especially when $n = 500$, the baseline method terminates with optimality gaps as high as 85% in most cases, whereas CB-SVM-HML achieves much smaller gaps. This confirms the positive impact of the CB cuts, which significantly tighten the formulation and decrease the optimality gap.

4.4.3
Impact of the time dedicated to cut separation

Finally, we measure the impact of the amount of computational effort dedicated to separating the CB cuts on solution quality. We therefore compare eight different configurations of CB-SVM-HML with different time budgets for T_S and T_B to evaluate the impact of CB-cuts separation on randomized sample subsets (during T_{S}) as well as on the complete problem (during T_{B}). For all configurations, we fix the total time limit to T_{MAX} to 600 seconds. Figure 4.4 shows the results of this experiment as boxplots, where each plot corresponds to the optimality gaps obtained by a method configuration on an instance set. Additionally, we indicate the number of optimal solutions found by each configuration on top of each boxplot.

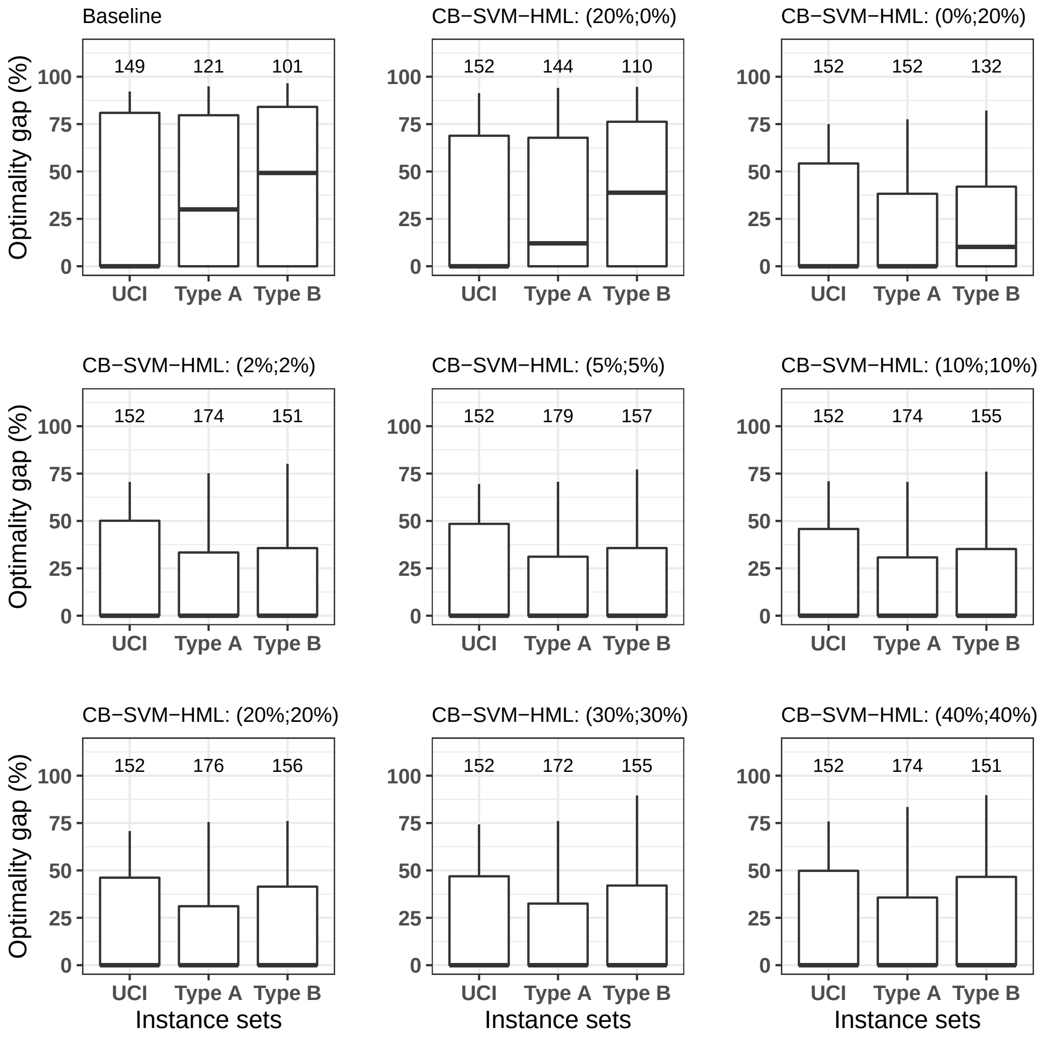


Figure 4.4: Optimality gaps achieved by CB-SVM-HML with different values of T_B and T_S

As can be seen, the configuration (5%; 5%) (which is the *reference configuration* used in Section 4.4.2) achieved the best performance among all the considered configurations. Moreover, all the configurations with combinatorial Benders' cuts achieved better gaps and number of optimal solutions than the baseline algorithm of [98]. Comparing configuration $(T_{\mathrm{S}}, T_{\mathrm{B}}) = (10\%, 10\%)$ with configurations $(20\%, 0\%)$ and $(0\%, 20\%)$, we notice that allocating the complete separation-time budget to the complete problem or to the randomized subproblems is not as effective as dividing the available time between these two approaches. Finally, allocating a larger amount of time to the separation process, as in configurations $(30\%, 30\%)$ and $(40\%, 40\%)$, also leads to a performance deterioration since there is a larger number of cuts, which leads to a heavier model, such that the remaining time to solve Step 3 is insufficient to obtain a good solution quality.

As seen in Table 4.7, the previous observations are confirmed at 0.05 significance level by paired-samples Wilcoxon tests. All versions of CB-SVM-HML obtained optimality gaps which were significantly smaller than the baseline method of [98] without cuts. Moreover, the configuration of CB-SVM-HML with $(T_{\text{S}}, T_{\text{B}}) = (5\%, 5\%)$ obtained better results than all configurations, except $(20\%, 20\%)$ for which no statistical difference was observed.

Table 4.7: Significance results of paired-samples Wilcoxon tests

Configuration		(20%,0%)	(0%,20%)	(2%,2%)	(5%,5%)	(10%,10%)	(20%,20%)	(30%,30%)	(40%,40%)
vs	p	<0.001	<0.001	<0.001	<0.001	<0.001	<0.001	<0.001	<0.001
Baseline [98]	sign	✓	✓	✓	✓	✓	✓	✓	✓
vs	p	<0.001	<0.001	<0.001	–	<0.001	0.796	<0.001	<0.001
(5%,5%)	sign	✓	✓	✓	–	✓		✓	✓

4.5 Conclusion

In this study, we have introduced CB-SVM-HML, a mixed integer programming approach based on combinatorial Benders' cuts for optimally training the SVM-HML. CB-SVM-HML operates by (i) generating an initial heuristic solution of the SVM-HML to obtain an upper bound on $||w||$, (ii) using this bound to define separation subproblems that permit to separate CB cuts, and finally (iii) solving the SVM-HML with these additional cuts. Through extensive experiments, we observed that this methodology permits substantial advances in the solution of the SVM-HML, increasing our ability to achieve optimal solutions and small optimality gaps. Our sensitivity analyses show that additionally separating CB cuts on small randomized subproblems with fewer samples also permits substantial performance improvements.

The findings of our study open many research avenues. First, we suggest pursuing methodological developments on mixed-integer programming strategies for the SVM-HML. Indeed, non-convex separation problems such as the SVM-HML with natural "big-M" MILP formulations are archetypal in many classification models (see, e.g., [119, 120, 121, 122, 123, 124]), such that new developments realized on one problem can trigger significant advances for the others. Next, while this study focuses on algorithms capable of proving optimality, research is still needed on efficient heuristics that can scale up to much larger data sets. Arguably, there is a need for both optimal algorithms and heuristics, since known optimal solutions give critical information regarding our ability to solve training problems reliably. Moreover, optimal or near-optimal solutions permit us to properly assess learning models without confounding factors due to the possible errors and instabilities of the training algorithms [66]. Finally, from a more general viewpoint, combinatorial optimization techniques can play an essential role in many other learning tasks and models. Especially given the rising concerns over equity, privacy, and transparency, sophisticated optimization strategies become necessary for challenging tasks related to model compression, training, and explanations, among others [125, 126, 127, 128].

5
Conclusions

The intersection of CO and ML has become a prospective area of research in science and engineering, mainly due to their common key elements (e.g., linear algebra, statistics, and mathematical optimization, among others). On the one hand, one can use ML techniques to improve existing CO methods while, on the other hand, CO techniques can be used to improve ML training. This thesis presented three studies intercepting CO and ML areas in both research directions.

In the first study, reported in Chapter 2, a pattern-mining strategy was used to generate high-order local-search neighborhood moves, which can be critical to exploiting the solution's search space effectively. Tested on two state-of-the-art metaheuristics for the Capacitated Vehicle Routing Problem (CVRP), we demonstrated that these generated moves, which would never be found otherwise, are valuable for improving these algorithms' overall performances. In the second study, which is also in the CVRP domain and presented in Chapter 3, a general concept of relatedness criteria was introduced. This concept was exploited in the contexts of reconnections performed by crossover operators and in the focus on promising moves by the local search. Due to its generality, two relatedness criteria were defined on geographical (distance-based criterion) and learnable (GNN-based criterion) sources to improve state-of-the-art results. In the experiments, we demonstrated that this sophisticated learning mechanism could improve state-of-the-art results but is also not superior to the performance of simply distance-based criterion. Overall, these results were against the superiority claims of improving algorithmic performance by employing sophisticated learning mechanisms or large networks, which made them *overkill* components to resolve simple decisions.

With these two studies, we observed two distinct aspects of 'ML in CO': a successful incorporation of ML into a CO method and an *overkill* approach based on a sophisticated ML technique almost equivalent to a more straightforward implementation without learning. Thus, these studies illustrated that the incorporation of ML techniques into CO methods is promising, but ablation studies are absolutely essential to properly attribute performance improvements and isolate the impact of the learning components.

In the third work of this thesis, shown in Chapter 4, we went in the direction of 'CO in ML'. In this case, we proposed new integer programming techniques and sampling strategies to train SVM-HML more efficiently. Indeed, the training of SVM-HML is a combinatorial optimization problem modeled over a combination of integer and continuous variables along with logical constraints. Remarkably, in our work, we identified that this mathematical formulation, after some adjustments, fits in the Combinatorial Benders' (CB) framework. As reported in the experiments, training this ML model became, in practice, substantially easier than the baseline once separating valid CB cuts was essential in achieving optimal solutions and smaller optimality gaps. Thus, applying CO techniques to ML models is not only essential in mitigating training time but also in making feasible the exploration of complex ML models.

From the studies conducted in this thesis, we observed that CO and ML fields could mutually benefit from each other. In particular, positive results for optimality, ML training performance, and running time were observed in the reported experiments. We believe that the performance incentives in exploiting techniques from one field to another are essential to progress towards practical algorithms. However, we also highlight the importance of considering standard approaches in the benchmark once the utilization of learning-based techniques may not compensate for the computational expenses, as discussed in Chapter 3. Overall, these lessons learned were crucial to the development of this thesis and may be promising for extension in open research opportunities. First, decomposition approaches appear to be a key component for solving open ML problems (classification, clustering, deep neural networks, etc.), particularly for making complex models feasible at a real-life scale. Second, employing ML techniques in CO algorithms for multi-objective problems and bi-level optimization seems a challenging research avenue for exploration. The computational cost of employing such techniques (e.g., pattern extraction and model training/prediction) for these problems is potentially expensive and non-trivial but encouraging to be tested for such a class of complex CO problems. Finally, we expect this thesis could present some benefits and shortcomings of the connection between CO and ML areas to the operations research and machine learning communities.

Bibliography

[1] BENGIO, Y.; LODI, A. ; PROUVOST, A.. **Machine learning for combinatorial optimization: A methodological tour d'horizon**. European Journal of Operational Research, 290(2):405–421, 2021.

[2] AGGARWAL, C.. **An introduction to frequent pattern mining**. In: Aggarwal, C.; Han, J., editors, FREQUENT PATTERN MINING, p. 1–17. Springer, Cham, 2014.

[4] VOUDOURIS, C.; TSANG, E. P.. **Guided local search**. Springer, 2003.

[5] ARNOLD, F.; SÖRENSEN, K.. **Knowledge-guided local search for the vehicle routing problem**. Computers & Operations Research, 105:32–46, 2019.

[6] ARNOLD, F.; GENDREAU, M. ; SÖRENSEN, K.. **Efficiently solving very large scale routing problems**. Computers & Operations Research, 107(1):32–42, 2019.

[7] LIN, S.; KERNIGHAN, B.. **An effective heuristic algorithm for the traveling-salesman problem**. Operations Research, 21(2):498–516, 1973.

[11] BARBALHO, H.; ROSSETI, I.; MARTINS, S. ; PLASTINO, A.. **A hybrid data mining GRASP with path-relinking**. Computers & Operations Research, 40(12):3159–3173, 2013.

[12] BOESE, K.. **Cost versus distance in the traveling salesman problem**. Technical report, UCLA Computer Science Dept, Los Angeles, 1995.

[15] CHRISTIAENS, J.; VANDEN BERGHE, G.. **Slack induction by string removals for vehicle routing problems**. Technical report, KU Leuven, Department of Computer Science, CODeS & imec, Leuven, 2018.

[18] DORIGO, M.; STÜTZLE, T.. **Ant colony optimization: Overview and recent advances**. In: Gendreau, M.; Potvin, J.-Y., editors, HANDBOOK OF METAHEURISTICS, p. 311–351. Springer, Boston, 3 edition, 2019.

[19] EL HACHEMI, N.; CRAINIC, T.; LAHRICHI, N.; REI, W. ; VIDAL, T.. **Solution integration in combinatorial optimization with applications to cooperative search and rich vehicle routing.** Journal of Heuristics, 21(5):663–685, 2015.

[23] GLOVER, F.. **Tabu search and adaptive memory programming – Advances, applications and challenges.** In: Barr, R.; Helgason, R. ; Kennington, J., editors, ADVANCES IN METAHEURISTICS, OPTIMIZATION, AND STOCHASTIC MODELING TECHNOLOGIES, p. 1–75. Springer, Boston, 1997.

[25] HOLLAND, J.. **Adaptation in natural and artificial systems: An introductory analysis with applications to biology, control, and artificial intelligence.** MIT Press, Cambridge, 1992.

[26] JOVANOVIC, R.; TUBA, M. ; VOSS, S.. **Fixed set search applied to the traveling salesman problem.** In: Aguilera, M.; Blum, C.; Santos, H.; Pinacho, P. ; Godoy, J., editors, HYBRID METAHEURISTICS, volumen 11299 de **Lecture Notes in Computer Science**, p. 63–77. Springer, Cham, 2019.

[27] KILBY, P.; SLANEY, J. ; WALSH, T.. **The backbone of the travelling salesperson.** In: PROCEEDINGS OF THE 19TH INTERNATIONAL JOINT CONFERENCE ON ARTIFICIAL INTELLIGENCE, p. 175–180, San Francisco, CA, USA, 2005.

[28] LAHRICHI, N.; CRAINIC, T.; GENDREAU, M.; REI, W.; CRIȘAN, G. ; VIDAL, T.. **An integrative cooperative search framework for multi-decision-attribute combinatorial optimization: Application to the MDPVRP.** European Journal of Operational Research, 246(2):400–412, 2015.

[31] LE BOUTHILLIER, A.; CRAINIC, T. ; KROPF, P.. **A guided cooperative search for the vehicle routing problem with time windows.** IEEE Intelligent Systems, 20(4):36–42, 2005.

[35] MUTER, I.; BIRBIL, C. ; ŞAHIN, G.. **Combination of metaheuristic and exact algorithms for solving set covering-type optimization problems.** INFORMS Journal on Computing, 22(4):603–619, 2010.

[36] NAGATA, Y.; BRÄYSY, O.. **Edge assembly-based memetic algorithm for the capacitated vehicle routing problem.** Networks, 54(4):205–215, 2009.

[38] PECIN, D.; PESSOA, A.; POGGI, M. ; UCHOA, E.. **Improved branch-cut-and-price for capacitated vehicle routing**. Mathematical Programming Computation, 9(1):61–100, 2017.

[40] PRINS, C.. **A simple and effective evolutionary algorithm for the vehicle routing problem**. Computers & Operations Research, 31(12):1985–2002, 2004.

[41] RESENDE, M.; RIBEIRO, C.; GLOVER, F. ; MARTÍ, R.. **Scatter search and path-relinking: Fundamentals, advances, and applications**. In: Gendreau, M.; Potvin, J.-Y., editors, HANDBOOK OF METAHEURISTICS, p. 87–107. Springer, Boston, 2 edition, 2010.

[42] RIBEIRO, M.; PLASTINO, A. ; MARTINS, S.. **Hybridization of GRASP metaheuristic with data mining techniques**. Journal of Mathematical Modelling and Algorithms, 5(1):23–41, 2006.

[43] SANTOS, H.; OCHI, L.; MARINHO, E. ; DRUMMOND, L.. **Combining an evolutionary algorithm with data mining to solve a single-vehicle routing problem**. Neurocomputing, 70(1-3):70–77, 2006.

[44] SCHNEIDER, J.. **Searching for backbones – a high-performance parallel algorithm for solving combinatorial optimization problems**. Future Generation Computer Systems, 19(1):121–131, 2003.

[45] SUBRAMANIAN, A.; UCHOA, E. ; OCHI, L.. **A hybrid algorithm for a class of vehicle routing problems**. Computers & Operations Research, 40(10):2519–2531, 2013.

[46] TAILLARD, É.; GAMBARDELLA, L.; GENDREAU, M. ; POTVIN, J.-Y.. **Adaptive memory programming: A unified view of metaheuristics**. European Journal of Operational Research, 135(1):1–16, 2001.

[48] TARANTILIS, C.; KIRANOUDIS, C.. **BoneRoute: An adaptive memory-based method for effective fleet management**. Annals of Operations Research, 115(1–4):227–241, 2002.

[49] TOTH, P.; VIGO, D.. **Vehicle routing: Problems, methods, and applications**. MOS-SIAM Series on Optimization, Philadelphia, 2 edition, 2014.

[50] UCHOA, E.; PECIN, D.; PESSOA, A.; POGGI, M.; VIDAL, T. ; SUBRAMANIAN, A.. **New benchmark instances for the capacitated**

vehicle routing problem. European Journal of Operational Research, 257(3):845–858, 2017.

[51] VIDAL, T.; CRAINIC, T.; GENDREAU, M.; LAHRICHI, N. ; REI, W.. **A hybrid genetic algorithm for multidepot and periodic vehicle routing problems**. Operations Research, 60(3):611–624, 2012.

[52] VIDAL, T.; CRAINIC, T.; GENDREAU, M. ; PRINS, C.. **A unified solution framework for multi-attribute vehicle routing problems**. European Journal of Operational Research, 234(3):658–673, 2014.

[55] VIDAL, T.; LAPORTE, G. ; MATL, P.. **A concise guide to existing and emerging vehicle routing problem variants**. European Journal of Operational Research, Articles in Advance, 2019.

[56] COSTA, L.; CONTARDO, C. ; DESAULNIERS, G.. **Exact branch-price-and-cut algorithms for vehicle routing**. Transportation Science, 53(4):946–985, 2019.

[60] VIDAL, T.; CRAINIC, T.; GENDREAU, M. ; PRINS, C.. **Heuristics for multi-attribute vehicle routing problems: A survey and synthesis**. European Journal of Operational Research, 231(1):1–21, 2013.

[61] BLUM, C.; ROLI, A.. **Metaheuristics in combinatorial optimization: Overview and conceptual comparison**. ACM Computing Surveys, 35(3):268–308, 2003.

[63] VIDAL, T.; CRAINIC, T.; GENDREAU, M. ; PRINS, C.. **A unified solution framework for multi-attribute vehicle routing problems**. European Journal of Operational Research, 234(3):658–673, 2014.

[65] GENDREAU, M.; HERTZ, A. ; LAPORTE, G.. **A tabu search heuristic for the vehicle routing problem**. Management Science, 40(10):1276–1290, 1994.

[66] GRIBEL, D.; VIDAL, T.. **HG-means: A scalable hybrid metaheuristic for minimum sum-of-squares clustering**. Pattern Recognition, 88:569–583, 2019.

[67] HANSEN, P.; RUIZ, M. ; ALOISE, D.. **A VNS heuristic for escaping local extrema entrapment in normalized cut clustering**. Pattern Recognition, 45(12):4337–4345, 2012.

[68] YUSTA, S.. **Different metaheuristic strategies to solve the feature selection problem**. Pattern Recognition Letters, 30(5):525–534, 2009.

[69] BAHROLOLOUM, A.; NEZAMABADI-POUR, H.. **A multi-expert based framework for automatic image annotation**. Pattern Recognition, 61:169–184, 2017.

[70] KASHEF, S.; NEZAMABADI-POUR, H.. **An advanced ACO algorithm for feature subset selection**. Neurocomputing, 147(1):271–279, 2015.

[71] IJJINA, E.; CHALAVADI, K.. **Human action recognition using genetic algorithms and convolutional neural networks**. Pattern Recognition, 59:199–212, 2016.

[72] HAN, F.; JIANG, J.; LING, Q.-H. ; SU, B.. **A survey on metaheuristic optimization for random single-hidden layer feedforward neural network**. Neurocomputing, 335:261–273, 2019.

[73] BAYATI, M.; BORGS, C.; CHAYES, J. ; ZECCHINA, R.. **Belief propagation for weighted b-matchings on arbitrary graphs and its relation to linear programs with integer solutions**. SIAM Journal on Discrete Mathematics, 25(2):989–1011, 2011.

[74] DAI, H.; KHALIL, E.; ZHANG, Y.; DILKINA, B. ; SONG, L.. **Learning combinatorial optimization algorithms over graphs**. Advances in Neural Information Processing Systems, p. 6348–6358, 2017.

[75] TALBI, E.-G.. **Machine learning into metaheuristics: A survey and taxonomy of data-driven metaheuristics**. Technical report, HAL-02745295, 2020.

[76] APPLEGATE, D. L.; BIXBY, R. E.; CHVÁTAL, V. ; COOK, W. J.. **Concorde TSP solver**, 2020.

[77] TÓTH, J.; TOMÁN, H. ; HAJDU, A.. **Efficient sampling-based energy function evaluation for ensemble optimization using simulated annealing**. Pattern Recognition, 107, 2020.

[78] LEE, J.; PERKINS, D.. **A simulated annealing algorithm with a dual perturbation method for clustering**. Pattern Recognition, 112:107713, 2021.

[79] MAVROFORAKIS, M.; THEODORIDIS, S.. **A geometric approach to support vector machine (SVM) classification**. IEEE Transactions on Neural Networks, 17(3):671–682, 2006.

[80] VAPNIK, V.. **Statistical Learning Theory**. Wiley, New York, 1998.

[81] XU, H.; CARAMANIS, C. ; MANNOR, S.. **Robustness and regularization of support vector machines**. Journal of Machine Learning Research, 10:1485–1510, 2009.

[82] BYUN, H.; LEE, S.. **Applications of support vector machines for pattern recognition: a survey**. In: INTERNATIONAL WORKSHOP ON SUPPORT VECTOR MACHINES, p. 213–236, Berlin, Heidelberg, 2002. Springer.

[83] CERVANTES, J.; GARCIA-LAMONT, F.; RODRÍGUEZ-MAZAHUA, L. ; LOPEZ, A.. **A comprehensive survey on support vector machine classification: applications, challenges and trends**. Neurocomputing, 408:189–215, 2020.

[84] CRISTIANINI, N.; SHAWE-TAYLOR, J.. **An introduction to support vector machines and other kernel-based learning methods**. Cambridge University Press, Cambridge, 2000.

[85] FEURER, M.; KLEIN, A.; EGGENSPERGER, K.; SPRINGENBERG, J.; BLUM, M. ; HUTTER, F.. **Efficient and robust automated machine learning**. Advances in Neural Information Processing Systems, 28:2962–2970, 2015.

[86] CHANG, C.; LIN, C.. **LIBSVM: a library for support vector machines**. ACM Transactions on Intelligent Systems and Technology, 2(3), 2011.

[87] COLLOBERT, R.; BENGIO, S.. **SVMTorch: Support vector machines for large-scale regression problems**. Journal of Machine Learning Research, 1:143–160, 2001.

[88] JOACHIMS, T.. **Making large-scale SVM learning practical**. In: ADVANCES IN KERNEL METHODS, p. 169–184. MIT Press, Cambridge, MA, USA, 1999.

[89] RUDIN, C.. **Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead**. Nature Machine Intelligence, 1(5):206–215, 2019.

[90] CHEN, C.; MANGASARIAN, O. L.. **Hybrid misclassification minimization**. Advances in Computational Mathematics, 5(1):127–136, 1996.

[91] PLATT, J.. **Fast training of support vector machines using sequential minimal optimization**. In: ADVANCES IN KERNEL METHODS — SUPPORT VECTOR LEARNING, p. 185–208, Cambridge, MA, 1999. MIT Press.

[92] LEE, Y.; HUANG, S.. **Reduced support vector machines: a statistical theory**. IEEE Transactions on Neural Networks, 18(1):1–13, 2007.

[93] LOOSLI, G.; CANU, S. ; BOTTOU, L.. **Training invariant support vector machines using selective sampling**. Large Scale Kernel Machines, 2, 2007.

[94] WANG, W.; XU, Z.. **A heuristic training for support vector regression**. Neurocomputing, 61:259–275, 2004.

[95] CARRIZOSA, E.; MORALES, D.. **Supervised classification and mathematical optimization**. Computers and Operations Research, 40(1):150–165, 2013.

[96] WANG, Q.; MA, Y.; ZHAO, K. ; TIAN, Y.. **A comprehensive survey of loss functions in machine learning**. Annals of Data Science, p. 1–26, 2020.

[97] SCHÖLKOPF, B.; SMOLA, A. ; BACH, F.. **Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond**. MIT Press, Cambridge, MA, 2002.

[98] BROOKS, J.. **Support vector machines with the ramp loss and the hard margin loss**. Operations Research, 59(2):467–479, 2011.

[99] ORSENIGO, C.; VERCELLIS, C.. **Multivariate classification trees based on minimum features discrete support vector machines**. IMA Journal of Management Mathematics, 14(3):221–234, 2003.

[100] ORSENIGO, C.; VERCELLIS, C.. **Discrete support vector decision trees via tabu search**. Computational Statistics and Data Analysis, 47(2):311–322, 2004.

[101] PÉREZ-CRUZ, F.; NAVIA-VÁZQUEZ, A.; FIGUEIRAS-VIDAL, A. ; ARTÉS-RODRÍGUEZ, A.. **Empirical risk minimization for support vector classifiers**. IEEE Transactions on Neural Networks, 14(2):296–303, 2003.

[102] COLLOBERT, R.; SINZ, F.; WESTON, J. ; BOTTOU, L.. **Trading convexity for scalability**. Proceedings of the 23rd International Conference on Machine Learning, p. 201–208, 2006.

[103] SHAWE-TAYLOR, J.; CRISTIANINI, N. ; OTHERS. **Kernel methods for pattern analysis**. Cambridge university press, 2004.

[104] SHEN, X.; TSENG, G.; ZHANG, X. ; WONG, W.. **On ψ-learning**. Journal of the American Statistical Association, 98(463):724–734, 2003.

[105] WOLSEY, L. A.. **Integer Programming**. John Wiley & Sons, Ltd, 2020.

[106] HUANG, X.; SHI, L. ; SUYKENS, J.. **Ramp loss linear programming support vector machine**. Journal of Machine Learning Research, 15(1):2185–2211, 2014.

[107] BELOTTI, P.; BONAMI, P.; FISCHETTI, M.; LODI, A.; MONACI, M.; NOGALES-GÓMEZ, A. ; SALVAGNIN, D.. **On handling indicator constraints in mixed integer programming**. Computational Optimization and Applications, 65(3):545–566, 2016.

[108] BALDOMERO-NARANJO, M.; MARTÍNEZ-MERINO, L. ; RODRÍGUEZ-CHÍA, A.. **Tightening big Ms in integer programming formulations for support vector machines with ramp loss**. European Journal of Operational Research, 286(1):84–100, 2020.

[109] POURSAEIDI, M.; KUNDAKCIOGLU, O.. **Robust support vector machines for multiple instance learning**. Annals of Operations Research, 216(1):205–227, 2014.

[110] HUANG, L.; SHAO, Y.; ZHANG, J.; ZHAO, Y. ; TENG, J.. **Robust Rescaled Hinge Loss Twin Support Vector Machine for Imbalanced Noisy Classification**. IEEE Access, 7:65390–65404, 2019.

[111] BENDERS, J.. **Partitioning procedures for solving mixed variables programming problems**. Numersiche Mathematik, 4:238–252, 1962.

[112] HOOKER, N.; OTTOSSON, G.. **Logic-Based Benders Decomposition**. Mathematical Programming, Series A, 96:33–60, 2003.

[113] CODATO, G.; FISCHETTI, M.. **Combinatorial Benders' cuts for mixed-integer linear programming**. Operations Research, 54(4):756–766, 2006.

[114] AKPINAR, S.; ELMI, A. ; BEKTAȘ, T.. **Combinatorial Benders cuts for assembly line balancing problems with setups**. European Journal of Operational Research, 259(2):527–537, 2017.

[115] CÔTÉ, J.; DELL'AMICO, M. ; IORI, M.. **Combinatorial Benders' cuts for the strip packing problem**. Operations Research, 62(3):643–661, 2014.

[116] ERDOĞAN, G.; BATTARRA, M. ; WOLFLER CALVO, R.. **An exact algorithm for the static rebalancing problem arising in bicycle sharing systems**. European Journal of Operational Research, 245(3):667–679, 2015.

[117] KEARNS, M.; VAZIRANI, U.. **An Introduction to Computational Learning Theory**. MIT Press, Cambridge, MA, 1994.

[118] DUA, D.; GRAFF, C.. **UCI machine learning repository**, 2017.

[119] CARRIZOSA, E.; MOLERO-RÍO, C. ; ROMERO MORALES, D.. **Mathematical optimization in classification and regression trees**. TOP, 29(1):5–33, 2021.

[120] ELIZONDO, D.. **The linear separability problem: Some testing methods**. IEEE Transactions on Neural Networks, 17(2):330–344, 2006.

[121] FLORIO, A.; MARTINS, P.; SCHIFFER, M.; SERRA, T. ; VIDAL, T.. **Optimal decision diagrams for classification**. Technical report, arXiv:2205.14500, 2022.

[122] FRÉVILLE, A.; HANAFI, S.; SEMET, F. ; YANEV, N.. **A tabu search with an oscillation strategy for the discriminant analysis problem**. Computers & Operations Research, 37(10):1688–1696, 2010.

[123] HANAFI, S.; YANEV, N.. **Tabu search approaches for solving the two-group classification problem**. Annals of Operations Research, 183(1):25–46, 2011.

[124] MURTHY, S. K.; KASIF, S. ; SALZBERG, S.. **A system for induction of oblique decision trees**. Journal of Artificial Intelligence Research (2), 2(1):1–32, 1994.

[125] FOREL, A.; PARMENTIER, A. ; VIDAL, T.. **Robust counterfactual explanations for random forests**. Technical report, arXiv:2205.14116, 2022.

[126] PARMENTIER, A.; VIDAL, T.. **Optimal counterfactual explanations in tree ensembles**. In: Meila, M.; Zhang, T., editors, PROCEEDINGS OF THE 38TH INTERNATIONAL CONFERENCE ON MACHINE LEARNING,

Proceedings of Machine Learning Research, p. 8422–8431, Virtual, 2021. PMLR.

[127] SERRA, T.; KUMAR, A. ; RAMALINGAM, S.. **Lossless compression of deep neural networks**. In: Hebrard, E.; Musliu, N., editors, CPAIOR 2020: INTEGRATION OF CONSTRAINT PROGRAMMING, ARTIFICIAL INTELLIGENCE, AND OPERATIONS RESEARCH, p. 417–430. Springer, 2020.

[128] VIDAL, T.; SCHIFFER, M.. **Born-again tree ensembles**. In: III, H. D.; Singh, A., editors, PROCEEDINGS OF THE 37TH INTERNATIONAL CONFERENCE ON MACHINE LEARNING, volumen 119 de **Proceedings of Machine Learning Research**, p. 9743–9753, Virtual, 2020. PMLR.

[129] WU, Y.; LIU, Y.. **Robust truncated hinge loss support vector machines**. Journal of the American Statistical Association, 102(479):974–983, 2007.

[130] FAN, R.; CHEN, P. ; LIN, C.. **Working set selection using second order information for training support vector machines**. Journal of Machine Learning Research, 6:1889–1918, 2005.

[131] VIDAL, T.; GRIBEL, D. ; JAILLET, P.. **Separable convex optimization with nested lower and upper constraints**. INFORMS Journal on Optimization, 1(1):71–90, 2019.

[132] VIDAL, T.; LAPORTE, G. ; MATL, P.. **A concise guide to existing and emerging vehicle routing problem variants**. European Journal of Operational Research, 286:401–416, 2020.

[134] PESSOA, A.; SADYKOV, R.; UCHOA, E. ; VANDERBECK, F.. **A generic exact solver for vehicle routing and related problems**. Mathematical Programming, 183(1):483–523, 2020.

[134] VIDAL, T.; CRAINIC, T.; GENDREAU, M. ; PRINS, C.. **Heuristics for multi-attribute vehicle routing problems: A survey and synthesis**. European Journal of Operational Research, 231(1):1–21, 2013.

[135] CHRISTIAENS, J.; VANDEN BERGHE, G.. **Slack induction by string removals for vehicle routing problems**. Transportation Science, 54(2):417–433, 2020.

[136] VIDAL, T.. **Hybrid genetic search for the cvrp: Open-source implementation and swap* neighborhood**. Computers & Operations Research, 140:105643, 2022.

[137] ARCHETTI, C.; KULLMAN, N.; MCGEOCH, C.; MENDOZA, J.; PARDALOS, P.; RESENDE, M.; UCHOA, E. ; VIDAL, T.. **12th dimacs challenge**. `http://dimacs.rutgers.edu/programs/challenge/vrp/`, 2022. Accessed: 2022-06-06.

[138] ACCORSI, L.; VIGO, D.. **A fast and scalable heuristic for the solution of large-scale capacitated vehicle routing problems**. Transportation Science, 55(4):832–856, 2021.

[138] SANTINI, A.; SCHNEIDER, M.; VIDAL, T. ; VIGO, D.. **Decomposition strategies for vehicle routing heuristics**. Technical report, Universitat Pompeu Fabra, 2021.

[139] TOTH, P.; VIGO, D.. **The granular tabu search and its application to the vehicle-routing problem**. INFORMS Journal on Computing, 15(4):333–346, 2003.

[140] NAGATA, Y.. **Edge assembly crossover for the capacitated vehicle routing problem**. Lecture Notes in Computer Science, 4446:142–153, 2007.

[141] HOPFIELD, J. J.; TANK, D. W.. **“Neural” computation of decisions in optimization problems**. Biological Cybernetics, 52(3):141–152, 1985.

[142] DAI, H.; KHALIL, E. B.; ZHANG, Y.; DILKINA, B. ; SONG, L.. **Learning combinatorial optimization algorithms over graphs**. In: PROCEEDINGS OF THE 31ST INTERNATIONAL CONFERENCE ON NEURAL INFORMATION PROCESSING SYSTEMS, p. 6351–6361, Red Hook, NY, USA, 2017. Curran Associates Inc.

[143] YOLCU, E.; PÓCZOS, B.. **Learning local search heuristics for boolean satisfiability**. In: Wallach, H.; Larochelle, H.; Beygelzimer, A.; d'Alché-Buc, F.; Fox, E. ; Garnett, R., editors, ADVANCES IN NEURAL INFORMATION PROCESSING SYSTEMS, volumen 32. Curran Associates, Inc., 2019.

[144] NAZARI, M.; OROOJLOOY, A.; TAKÁČ, M. ; SNYDER, L. V.. **Reinforcement learning for solving the vehicle routing problem**. In: PROCEEDINGS OF THE 32ND INTERNATIONAL CONFERENCE ON NEURAL INFORMATION PROCESSING SYSTEMS, p. 9861–9871, 2018.

[145] CHEN, X.; TIAN, Y.. **Learning to perform local rewriting for combinatorial optimization**. In: ADVANCES IN NEURAL INFORMATION PROCESSING SYSTEMS, volumen 32, p. 6278–6289, 2019.

[146] KOOL, W.; VAN HOOF, H.; GROMICHO, J. ; WELLING, M.. **Deep policy dynamic programming for vehicle routing problems**. In: Schaus, P., editor, INTEGRATION OF CONSTRAINT PROGRAMMING, ARTIFICIAL INTELLIGENCE, AND OPERATIONS RESEARCH, p. 190–213, Cham, 2022. Springer International Publishing.

[147] XIN, L.; SONG, W.; CAO, Z. ; ZHANG, J.. **Neurolkh: Combining deep learning model with lin-kernighan-helsgaun heuristic for solving the traveling salesman problem**. In: ADVANCES IN NEURAL INFORMATION PROCESSING SYSTEMS, volumen 34, 2021.

[148] SCHNEIDER, M.; SCHWAHN, F. ; VIGO, D.. **Designing granular solution methods for routing problems with time windows**. European Journal of Operational Research, 263(2):493–509, 2017.

[149] VIDAL, T.; CRAINIC, T. G.; GENDREAU, M. ; PRINS, C.. **A hybrid genetic algorithm with adaptive diversity management for a large class of vehicle routing problems with time-windows**. Computers and Operations Research, 40(1):475–489, 2013.

[150] VIDAL, T.. **Split algorithm in o(n) for the capacitated vehicle routing problem**. Computers & Operations Research, 69:40–47, 2016.

[151] OLIVER, I. M.; SMITH, D. J. ; HOLLAND, J. R. C.. **A study of permutation crossover operators on the traveling salesman problem**. In: PROCEEDINGS OF THE SECOND INTERNATIONAL CONFERENCE ON GENETIC ALGORITHMS ON GENETIC ALGORITHMS AND THEIR APPLICATION, p. 224–230, 1987.

[152] QUEIROGA, E.; SADYKOV, R.; UCHOA, E. ; VIDAL, T.. **10,000 optimal CVRP solutions for testing machine learning based heuristics**. In: AAAI-22 WORKSHOP ON MACHINE LEARNING FOR OPERATIONS RESEARCH (ML4OR), 2022.

[154] VESSELINOVA, N.; STEINERT, R.; PEREZ-RAMIREZ, D. F. ; BOMAN, M.. **Learning combinatorial optimization on graphs: A survey with applications to networking**. IEEE Access, 8:120388–120416, 2020.

[155] MAZYAVKINA, N.; SVIRIDOV, S.; IVANOV, S. ; BURNAEV, E.. **Reinforcement learning for combinatorial optimization: A survey**. Computers & Operations Research, 134:105400, 2021.

[157] KARIMI-MAMAGHAN, M.; MOHAMMADI, M.; MEYER, P.; KARIMI-MAMAGHAN, A. M. ; TALBI, E. G.. **Machine learning at the service of meta-heuristics for solving combinatorial optimization problems: A state-of-the-art**. European Journal of Operational Research, 296(2):393–422, 2022.

[157] JOSHI, C. K.; LAURENT, T. ; BRESSON, X.. **An efficient graph convolutional network technique for the travelling salesman problem**, 2019.

[158] GAMBELLA, C.; GHADDAR, B. ; NAOUM-SAWAYA, J.. **Optimization problems for machine learning: A survey**. European Journal of Operational Research, 290(3):807–828, 2021.

[159] PEDRO DUARTE SILVA, A.. **Optimization approaches to Supervised Classification**. European Journal of Operational Research, 261(2):772–788, 2017.

[160] BOTTOU, L.; CURTIS, F. E. ; NOCEDAL, J.. **Optimization methods for large-scale machine learning**. SIAM Review, 60(2):223–311, 2018.

[161] BRESSON, X.; LAURENT, T.. **Residual gated graph convnets**, 2017.

A
Detailed Experimental Results – Set XML

Table A.1 provides additional detailed results, covering all values of $\Gamma \in \{15, 20, 30, 50\}$ on all XML instances of [152]. It reports the number of optimal solutions ($\#\textsc{Opt}$) attained over the 10,000 XML instances and the average final $\textsc{Gap}(\%)$, calculated as $\textsc{Gap}(\%) = 100 \times (z - z_{\text{BKS}})/z_{\text{BKS}}$, where z represents the cost of the solution and z_{BKS} is the optimal or BKS cost value. The results are provided for all of the methods, considering the four possible GNN configurations ($\textsc{Original}$, $\textsc{Optimistic}$, $\textsc{Model}\ \#1$, and $\textsc{Model}\ \#2$). The best performance in each row is highlighted in boldface.

Additionally, Figure A.1 provides convergence plots for all possible GNN configurations. The graphs included in it represent the progress of the average gap over time, considering the parameter value $\Gamma = 15$. On these graphs, it is noteworthy that the $\textsc{Original}$ configuration of the GNN requires significant inference time to generate the heatmap before the optimization can start. In comparison, $\textsc{Model}\ \#1$ and $\textsc{Model}\ \#2$ start much faster and behave quite similarly to the $\textsc{Optimistic}$ curve that ignores inference time.

Next, Figure A.2 presents boxplots of the percentage error gaps of the methods for different subsets of the XML instances, using the $\textsc{Original}$ configuration of the GNN and $\Gamma = 15$. Indeed, as discussed in [50] and [152], the instances were designed to mimic important features present in real-world problems, with different conventions regarding depot position, customers position, demand distribution, and average route size. Therefore, each boxplot corresponds to a subset of the instances with a specific configuration for one of these parameters. In these boxplots, the boxes indicate the first and third quartile, and the whiskers extend to 1.5 times the interquartile range.

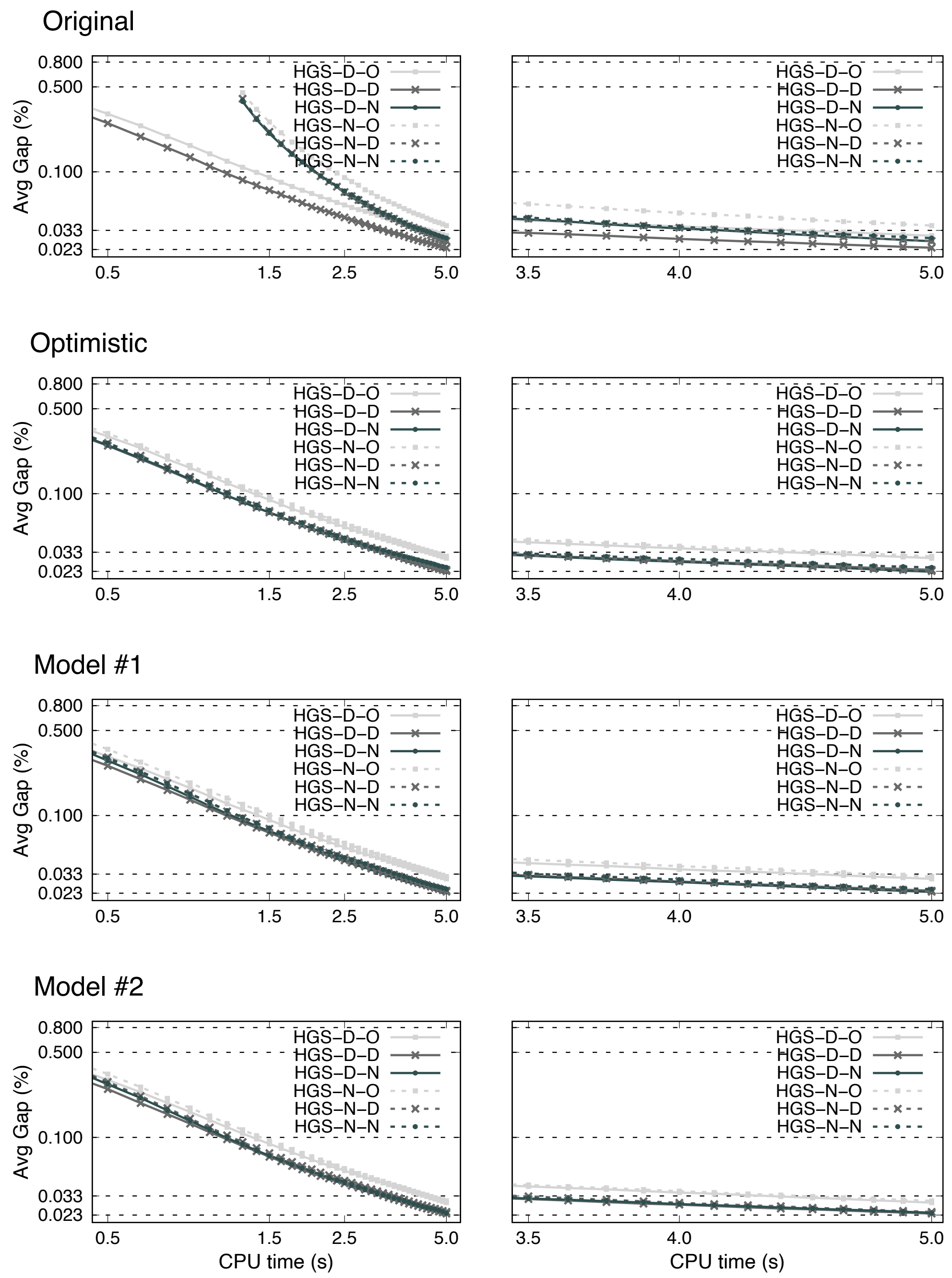


Figure A.1: Convergence plots on set XML for all GNN configurations (for each configuration, left graph = complete run, right graph = last 1.5 seconds of the run time)

Table A.1: Results for set XML for all values of $\Gamma \in \{15, 20, 30, 50\}$

	HGS-D-O		HGS-D-D		HGS-D-N		HGS-N-O		HGS-N-D		HGS-N-N	
GNN	#Opt	Gap%	#Opt	Gap%	#Opt	Gap%	#Opt	Gap%	#Opt	Gap%	#Opt	Gap%
Original (#Nodes: 300; #Hidden Layers: 30; #Epochs: 1500; Pred-T(s) = 0.85)												
15	7715	0.030	**8105**	0.024	8086	**0.023**	7691	0.031	8046	**0.023**	8011	0.025
20	7612	0.032	**7969**	**0.025**	7932	0.026	7482	0.035	7903	0.027	7811	0.028
30	7181	0.041	**7590**	**0.031**	7544	0.032	7054	0.044	7466	0.034	7419	0.036
50	6421	0.063	**6804**	**0.050**	6775	0.052	6365	0.067	6674	0.055	6684	0.055
Avg	7232.3	0.042	**7617.0**	**0.033**	7584.3	**0.033**	7148.0	0.044	7522.3	0.035	7481.3	0.036
Optimistic (#Nodes: 300; #Hidden Layers: 30; #Epochs: 1500; Pred-T(s) = Ignored)												
15	7715	0.030	8105	0.024	**8120**	**0.023**	7732	0.031	8062	**0.023**	8041	0.025
20	7612	0.032	**7969**	**0.025**	7956	0.026	7515	0.034	7942	0.026	7814	0.028
30	7181	0.041	7590	**0.031**	**7591**	**0.031**	7102	0.044	7517	0.033	7465	0.035
50	6421	0.063	6804	**0.050**	**6830**	0.052	6427	0.066	6718	0.054	6748	0.054
Avg	7232.3	0.042	7617.0	**0.033**	**7624.3**	**0.033**	7194.0	0.044	7559.8	0.034	7517.0	0.036
Model #1 (#Nodes: 10; #Hidden Layers: 5; #Epochs: 500; Pred-T(s) = 0.03)												
15	7715	0.030	**8105**	0.024	8094	**0.023**	7697	0.031	8028	0.024	7994	0.025
20	7612	0.032	**7969**	**0.025**	7941	**0.025**	7450	0.034	7822	0.027	7865	0.027
30	7181	0.041	**7590**	**0.031**	7524	0.032	7068	0.044	7464	0.034	7384	0.036
50	6421	0.063	6804	**0.050**	**6815**	0.051	6379	0.066	6709	0.056	6715	0.055
Avg	7232.3	0.042	**7617.0**	**0.033**	7593.5	**0.033**	7148.5	0.044	7505.8	0.035	7489.5	0.036
Model #2 (#Nodes: 10; #Hidden Layers: 5; #Epochs: 1500; Pred-T(s) = 0.03)												
15	7715	0.030	**8105**	**0.024**	8102	**0.024**	7719	0.030	8067	**0.024**	8057	**0.024**
20	7612	0.032	**7969**	**0.025**	7954	**0.025**	7577	0.033	7845	0.027	7900	0.026
30	7181	0.041	**7590**	**0.031**	7582	**0.031**	7105	0.042	7545	0.033	7503	0.033
50	6421	0.063	6804	0.050	**6830**	**0.049**	6394	0.065	6724	0.054	6758	0.053
Avg	7232.3	0.042	**7617.0**	0.033	**7617.0**	**0.032**	7198.8	0.043	7545.3	0.035	7554.5	0.034

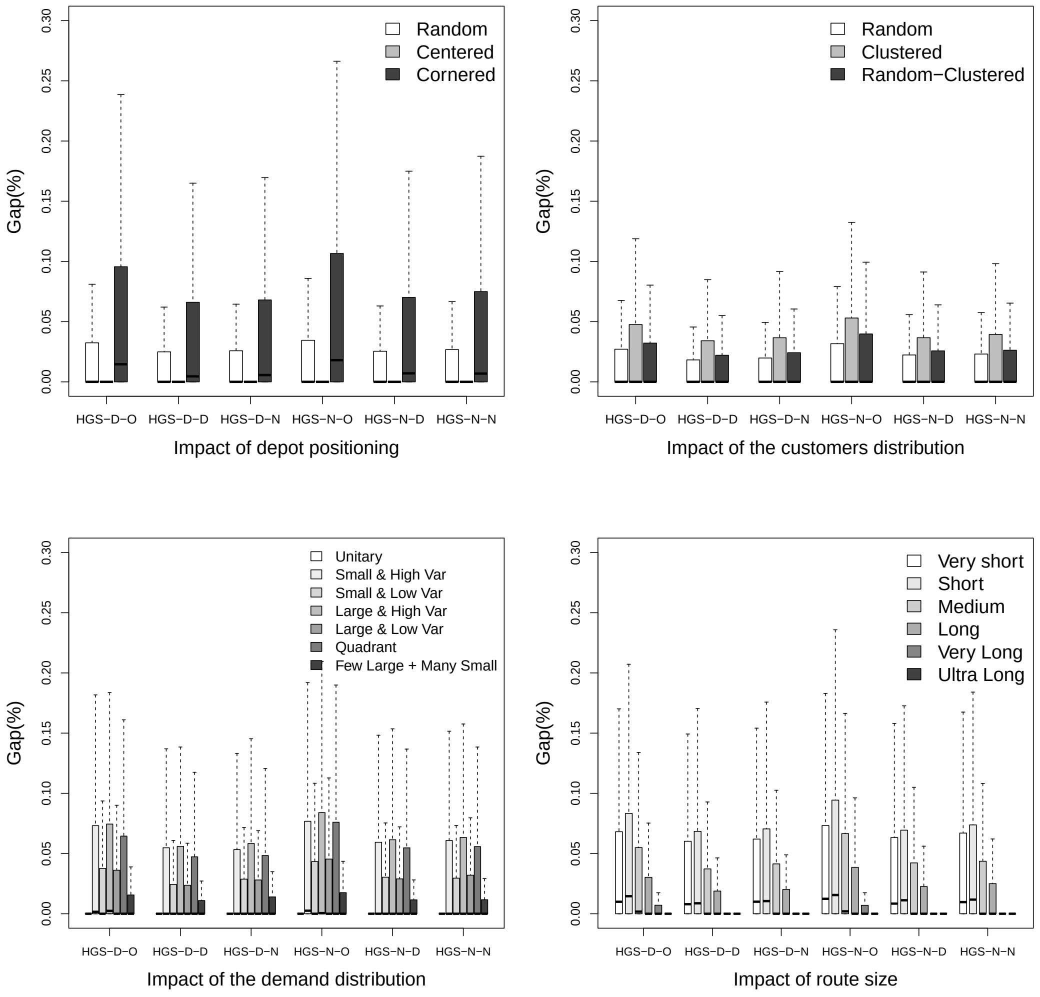


Figure A.2: Boxplots of the percentage error gaps achieved by the methods, for different subsets of the XML instances, using $\Gamma = 15$ and the Original configuration of the GNN

B
Detailed Experimental Results – Set X

Finally, Table B.1 and Figures B.1–B.2 provide the same detailed results for the X instance set of [50].

Table B.1: Results for set X for all values of $\Gamma \in \{15, 20, 30, 50\}$

	HGS-D-O		HGS-D-D		HGS-D-N		HGS-N-O		HGS-N-D		HGS-N-N	
GNN	#Opt	Gap%	#Opt	Gap%	#Opt	Gap%	#Opt	Gap%	#Opt	Gap%	#Opt	Gap%
Original (#Nodes: 300; #Hidden Layers: 100; #Epochs: 1500; Pred-T(s) = 15.2%)												
15	**188**	0.368	187	**0.302**	184	0.317	177	0.395	169	0.325	172	0.317
20	166	0.380	**186**	**0.305**	184	0.323	160	0.412	169	0.328	175	0.322
30	162	0.401	**176**	**0.318**	164	0.341	139	0.438	173	0.355	160	0.348
50	130	0.464	**158**	**0.370**	140	0.402	114	0.522	146	0.410	134	0.411
Avg	161.5	0.403	**176.8**	**0.324**	168.0	0.346	147.5	0.442	164.3	0.354	160.3	0.350
Optimistic (#Nodes: 300; #Hidden Layers: 100; #Epochs: 1500; Pred-T(s) = Ignored)												
15	**188**	0.368	187	0.302	187	**0.299**	185	0.365	177	0.307	182	**0.299**
20	166	0.380	186	**0.305**	**191**	0.306	170	0.383	176	0.308	179	**0.305**
30	162	0.401	**176**	**0.318**	171	0.320	153	0.406	**176**	0.331	167	0.326
50	130	0.464	**158**	**0.370**	153	0.372	122	0.482	154	0.380	142	0.382
Avg	161.5	0.403	**176.8**	**0.324**	175.5	**0.324**	157.5	0.409	170.8	0.332	167.5	0.328
Model #1 (#Nodes: 10; #Hidden Layers: 5; #Epochs: 500; Pred-T(s) = 1.9%)												
15	**188**	0.368	187	**0.302**	185	0.309	166	0.414	177	0.332	184	0.336
20	166	0.380	**186**	0.305	172	0.312	165	0.419	177	0.337	179	0.340
30	162	0.401	176	0.318	**181**	0.322	149	0.445	170	0.355	164	0.350
50	130	0.464	**158**	0.370	143	0.373	120	0.500	149	0.391	143	0.400
Avg	161.5	0.403	**176.8**	**0.324**	170.3	0.329	150.0	0.445	168.3	0.354	167.5	0.357
Model #2 (#Nodes: 10; #Hidden Layers: 5; #Epochs: 1500; Pred-T(s) = 1.9%)												
15	**188**	0.368	187	0.302	186	**0.301**	169	0.391	175	0.314	170	0.316
20	166	0.380	**186**	**0.305**	179	0.310	162	0.394	183	0.321	177	0.322
30	162	0.401	**176**	**0.318**	174	0.322	164	0.429	162	0.335	174	0.340
50	130	0.464	**158**	**0.370**	155	0.371	127	0.478	150	0.388	146	0.389
Avg	161.5	0.403	**176.8**	**0.324**	173.5	0.326	155.5	0.423	167.5	0.340	166.8	0.342

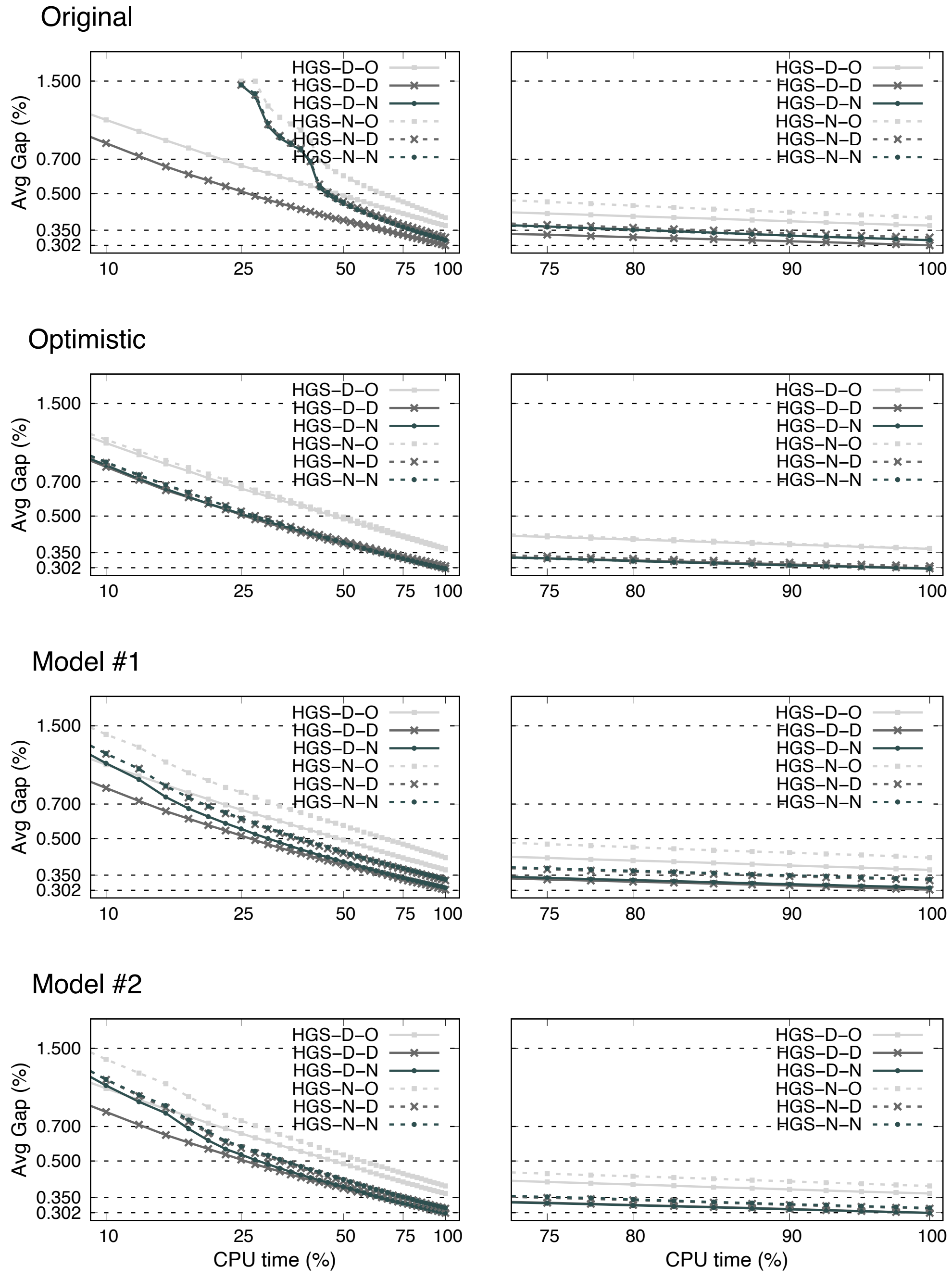


Figure B.1: Convergence plots on set X for all GNN configurations (for each configuration, left graph = complete run, right graph = last 25 % of the run time)

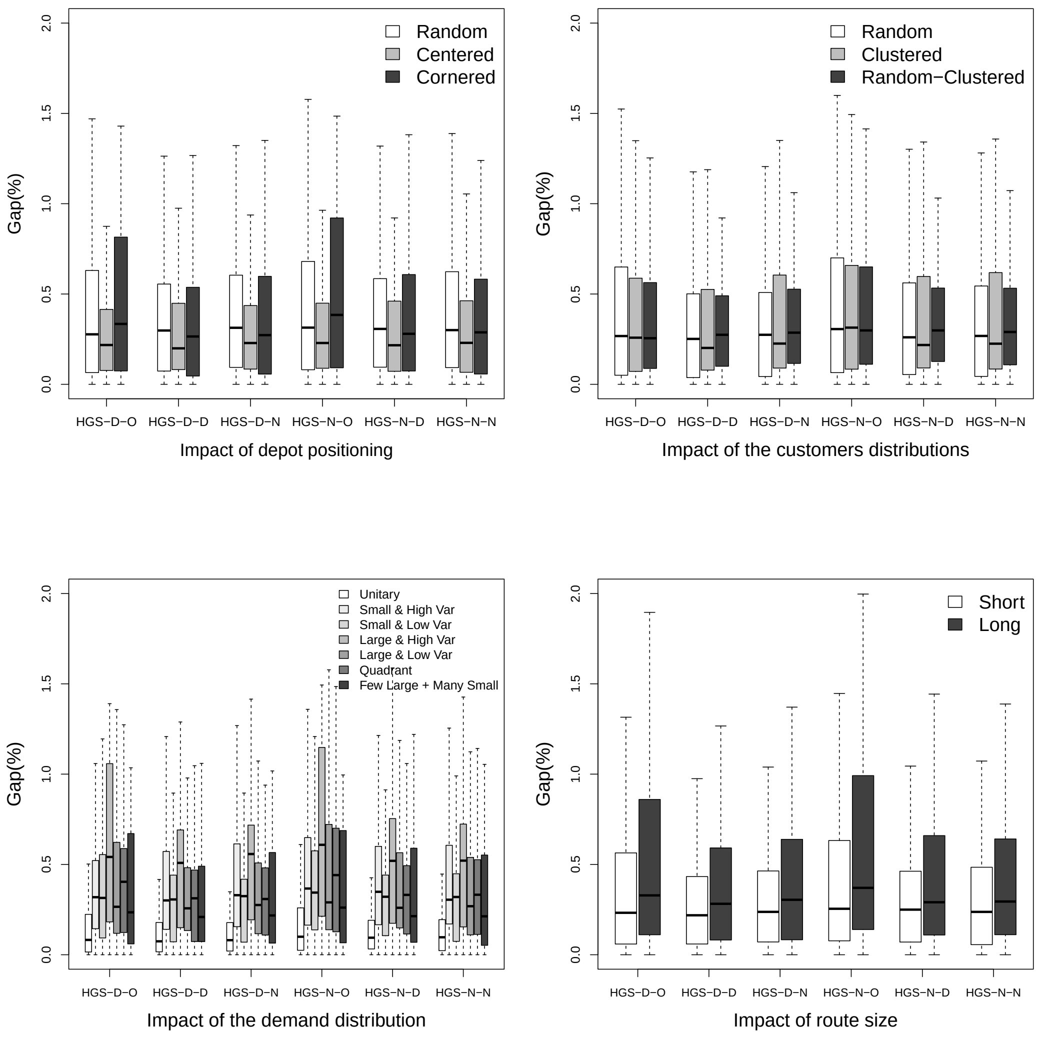


Figure B.2: Boxplots of the percentage error gaps achieved by the methods, for different subsets of the X instances, using $\Gamma = 15$ and the ORIGINAL configuration of the GNN

C
Publication Status

Chapter 2 (“PILS: Exploring high-order neighborhoods by pattern mining and injection”) is already published on *Pattern Recognition*.

– Reference: Arnold, F. , Santana, Í. , Sörensen, K. , & Vidal, T. (2021). **PILS: Exploring high-order neighborhoods by pattern mining and injection**. Pattern Recognition, 107957.

Chapter 3 (“Neural Networks for Local Search and Crossover in Vehicle Routing: A Possible Overkill?”) is *submitted*.

Chapter 4 (“Support Vector Machines with the Hard-Margin Loss: Optimal Training via Combinatorial Benders’ Cuts”) is *submitted*.