A HYBRID MATHEURISTIC APPLIED TO THE KNAPSACK PROBLEM WITH FORFEITS

Gabriel Souto de Jesus Augusto

Dissertação de Mestrado apresentada ao Programa de Pós-graduação em Engenharia de Sistemas e Computação, COPPE, da Universidade Federal do Rio de Janeiro, como parte dos requisitos necessários à obtenção do título de Mestre em Engenharia de Sistemas e Computação.

Orientadores: Pedro Henrique González Silva
Luidi Simonetti

Rio de Janeiro
Março de 2025

A HYBRID MATHEURISTIC APPLIED TO THE KNAPSACK PROBLEM WITH FORFEITS

Gabriel Souto de Jesus Augusto

DISSERTAÇÃO SUBMETIDA AO CORPO DOCENTE DO INSTITUTO ALBERTO LUIZ COIMBRA DE PÓS-GRADUAÇÃO E PESQUISA DE ENGENHARIA DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE MESTRE EM CIÊNCIAS EM ENGENHARIA DE SISTEMAS E COMPUTAÇÃO.

Orientadores: Pedro Henrique González Silva
Luidi Simonetti

Aprovada por: Prof. Pedro Henrique González Silva
Prof. Luidi Simonetti

RIO DE JANEIRO, RJ – BRASIL
MARÇO DE 2025

Souto de Jesus Augusto, Gabriel

A Hybrid Matheuristic applied to the Knapsack Problem with Forfeits/Gabriel Souto de Jesus Augusto. – Rio de Janeiro: UFRJ/COPPE, 2025.

X, 43 p.: il.; 29, 7cm.

Orientadores: Pedro Henrique González Silva
Luidi Simonetti

Dissertação (mestrado) – UFRJ/COPPE/Programa de Engenharia de Sistemas e Computação, 2025.

Referências Bibliográficas: p. 40 – 43.

1. Knapsack Problem. 2. Data Mining. 3. Iterated Local Search. I. González Silva, Pedro Henrique *et al.* II. Universidade Federal do Rio de Janeiro, COPPE, Programa de Engenharia de Sistemas e Computação. III. Título.

Agradecimentos

Gostaria de manifestar minha mais profunda gratidão, primeiramente a Deus, por me conceder a força e a sabedoria necessárias para concluir esta etapa importante da minha vida.

A todas as pessoas que tornaram este trabalho possível e que, de alguma forma, contribuíram para o meu desenvolvimento pessoal, profissional e acadêmico, expresso meu sincero reconhecimento.

À minha querida esposa, cujo amor, compreensão, apoio, encorajamento e companheirismo foram essenciais para que eu chegasse até aqui. Sua empatia, paciência e compreensão nos momentos mais desafiadores me fortaleceram para que eu pudesse continuar avançando.

Aos meus pais e à minha irmã, que sempre acreditaram em mim, apoiaram todas as minhas aspirações e continuam a me incentivar e auxiliar em tudo o que está ao seu alcance.

Aos meus orientadores, Pedro e Luidi, agradeço por fornecerem esclarecimentos diários sobre a pesquisa, por seu apoio e paciência diante das minhas limitações, por todos os ensinamentos compartilhados e por serem sempre acessíveis. Sua dedicação e compromisso com a excelência acadêmica são verdadeiramente inspiradores. Além disso, são pessoas por quem tenho profunda admiração e que me apoiaram para que eu pudesse concluir essa pesquisa.

Aos meus amigos, que tornam a vida mais leve e alegre, facilitando a superação desta árdua jornada, expresso minha profunda gratidão.

A todos, meu mais sincero agradecimento.

Resumo da Dissertação apresentada à COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Mestre em Ciências (M.Sc.)

A HYBRID MATHEURISTIC APPLIED TO THE KNAPSACK PROBLEM WITH FORFEITS

Gabriel Souto de Jesus Augusto

Março/2025

Orientadores: Pedro Henrique González Silva
Luidi Simonetti

Programa: Engenharia de Sistemas e Computação

Esta dissertação de mestrado aborda o Problema da Mochila com Perdas (KPF), uma extensão do clássico Problema da Mochila 0–1 que introduz conflitos suaves entre itens. No KPF, selecionar certos pares de itens juntos incorre em uma penalidade de lucro em vez de ser proibido de imediato. Esse modelo captura melhor cenários do mundo real envolvendo incompatibilidades parciais. A metodologia proposta, chamada Hybrid Data Mining ILS-VND (HDM-ILS-VND), integra uma Busca Local Iterada com Descida de Vizinhança Variável (ILS-VND) com dois aprimoramentos principais: mineração frequente de conjuntos de itens e ramificação local. Na fase de mineração de dados, padrões são descobertos a partir de um conjunto de soluções de elite, orientando a construção de um modelo linear misto-inteiro cuja região viável é restrita a combinações promissoras. Uma fase de ramificação local então refina ainda mais a melhor solução encontrada. Experimentos computacionais foram conduzidos em instâncias de referência usadas na literatura. Os resultados confirmam que o HDM-ILS-VND supera a abordagem ILS-VND padrão e rivaliza com métodos de última geração. Em muitos casos, ele encontra melhores soluções, mantendo tempos de computação razoáveis. Uma análise estatística completa por meio de um Teste de Postos Sinalizados de Wilcoxon valida a superioridade da abordagem híbrida proposta. Esta Dissertação destaca a vantagem de aumentar a heurística com insights de mineração de dados e ramificação local para resolver efetivamente o KPF. A estrutura computacional é mostrada muito robusta no geral.

Abstract of Dissertation presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Master of Science (M.Sc.)

A HYBRID MATHEURISTIC APPLIED TO THE KNAPSACK PROBLEM WITH FORFEITS

Gabriel Souto de Jesus Augusto

March/2025

Advisors: Pedro Henrique González Silva
Luidi Simonetti

Department: Systems Engineering and Computer Science

This Master's Thesis addresses the Knapsack Problem with Forfeits (KPF), an extension of the classic 0–1 Knapsack Problem introducing soft conflicts among items. In KPF, selecting certain item pairs together incurs a profit penalty rather than being forbidden outright. Such a model better captures real-world scenarios involving partial incompatibilities. The proposed methodology, called Hybrid Data Mining ILS-VND (HDM-ILS-VND), integrates an Iterated Local Search with Variable Neighborhood Descent (ILS-VND) with two key enhancements: frequent itemset mining and local branching. In the data mining phase, patterns are discovered from an elite set of solutions, guiding the construction of a mixed-integer linear model whose feasible region is restricted to promising combinations. A local branching phase then further refines the best solution found. Computational experiments were conducted on benchmark instances used in the literature. The results confirm that HDM-ILS-VND outperforms the standard ILS-VND approach and rivals state-of-the-art methods. On many instances, it finds better solutions while maintaining reasonable computation times. A thorough statistical analysis via a Wilcoxon Signed-Rank Test validates the superiority of the proposed hybrid approach. This thesis highlights the advantage of augmenting heuristics with data mining insights and local branching to solve KPF effectively. The computational framework is shown to be very robust overall.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Constraint options are frequently considered in optimization scenarios to manage conflicting choices where certain decisions cannot be made simultaneously. In the literature, these situations are called hard conflicts, meaning simultaneous selection is not allowed. On the other hand, soft conflicts, also known as forfeits, permit both decisions to be made together, but this comes with an associated penalty cost. Conflict choices have been studied in classical problems in the literature, such as the Minimum Spanning Tree Problem (DARMANN *et al.*, 2009; CARRABS *et al.*, 2021), Maximum Flow Problem (GALBIATI, 2011), Knapsack Problem (YAMADA *et al.*, 2002; HIFI e OTMANI, 2012; BETTINELLI *et al.*, 2017; BASNET, 2018; FÜGENSCHUH *et al.*, 2019) and Bin Packing Problem (EPSTEIN e LEVIN, 2008; MURITIBA *et al.*, 2010).

These problems with conflicts have been addressed with many solving optimization techniques, such as branch-and-bound (BETTINELLI *et al.*, 2017; CONIGLIO *et al.*, 2021), branch-and-cut (BEN SALEM *et al.*, 2018), genetic algorithms (CAPOBIANCO *et al.*, 2022), heuristics (HIFI e OTMANI, 2012; MOURA *et al.*, 2021), and hybrid methods (SOUTO *et al.*, 2024a,b,c).

Building on these efforts, the Knapsack Problem with Forfeits (KPF) was proposed in CERULLI *et al.* (2020) as a variant of the 0–1 Knapsack Problem that models "soft" conflicts between items. In the KPF, selecting both items from a conflict pair triggers a cost deduction rather than forbidding that pair outright. This framework is notably flexible in scenarios where strictly avoiding conflicts can be overly restrictive, and it has been shown to capture real-world interactions more accurately CERULLI *et al.* (2020), CAPOBIANCO *et al.* (2022).

The KPF can be applied in resource management and personnel allocation. For instance, it can schedule workers who may only operate certain machines (and thereby incur added costs when pairing them) or assign work shifts that become more expensive if combined. Similar needs appear in forming teams where particular individuals overlap in roles or require extra supervision. Such scenarios benefit

from the KPF's "soft" constraint approach, which balances penalties against overall gains CERULLI *et al.* (2020).

Healthcare applications also emerge naturally. The KPF can balance nutritional and pharmaceutical options for hospitalized patients, accounting for negative drug-drug or drug–nutrient interactions. This perspective extends to selecting medications under strict budget limits, where two interacting treatments may still be worth combining if the overall efficacy exceeds the penalty incurred.

Likewise, KPF may represent cargo-loading situations in logistics and transportation where certain items produce additional costs (e.g., contamination or safety overhead) when shipped together. By allowing but penalizing these "conflicting" item pairs, transportation planners can decide whether separating items or transporting them simultaneously while paying the associated penalty is more profitable.

In the financial domain, forfeits may help capture diversification constraints in portfolio selection. Instead of forbidding assets of a similar type, KPF charges a penalty for holding overly correlated assets, thus representing practical tradeoffs between greater diversification and the pursuit of high-yield but overlapping investments.

Considering the applications of the KPF in real-world scenarios, we proposed a new methodology for solving the KPF using an Iterated Local Search with Variable Neighborhood Descent (ILS-VND) (VIEIRA *et al.*, 2024), combined with Frequent Itemset Mining (MINING, 2006) and a Local Branching (FISCHETTI e LODI, 2003) phase to refine the solution. The results of the proposed hybrid data mining ILS-VND (HDM-ILS-VND) method show competitiveness, with most instances obtaining good solutions within a reasonable amount of time.

The remainder of the master thesis is structured as follows: Chapter 2 presents the leading works in the literature on the problem. Chapter 3 provides a detailed description of the KPF and presents a mixed integer programming formulation. In Chapter 4, we describe the proposed method and its main components. Computational experiments and their results are discussed in Chapter 5. Finally, conclusions and suggestions for future research are presented in Chapter 6.

Chapter 2

Literature Review

Following PAGE *et al.* (2021) methodology, we systematically surveyed the literature to locate, assess, and compile pertinent studies on the KPF. This structured and unbiased approach enabled us to identify, analyze, and synthesize all relevant research on the topic.

Given the objectives of the chapter, we organize it by showing in the first section (Section 2.1) a quantitative analysis of the selected papers in the systematic review. Afterward, Section 2.2 makes a more qualitative analysis of the main papers in the literature of the KPF.

2.1 Quantitative Analysis

This study was developed from the literature review in the Scopus, Web of Science (WOS), ScienceDirect, and IEEE Xplore databases from 2020 to 2025. It is important to mention that we began conducting the search in 2020 because it was the year of the first paper on the KPF. The keywords used were "knapsack", "problem", "forfeits", and, in Portuguese, "mochila", "problem", and "penalidades". The search string: (knapsack OR mochila) AND (problem OR problema) AND (forfeits OR penalidades).

The main results of the review process used the following method:

1. **Found**: Searches using the keyword and search string in main databases: Scopus, Web of Science (WOS), ScienceDirect, and IEEE Xplore. The result of this first phase is scientific papers related to search strings.

2. **Duplicates**: Indicates the number of articles that appeared in more than one database. These articles are identified and removed to avoid counting the same studies repeatedly.

3. **Filtered**: To reduce the number of articles to a relevant scope, we filtered by relevance to the research topic with the mentioned keywords, abstract, and document text.

4. **Selected**: Represents the final number of articles chosen for detailed analysis after systematic reading and assessment of the relevance, methodological quality, and results.

These steps ensure that the review is comprehensive, impartial, and evidence-based, minimizing bias and ensuring the quality of the studies included in the analysis.

Table 2.1: Quantitative Analysis by Database

Database	**Found**	**Duplicates**	**Filtered**	**Selected**
IEEE Xplore	1	1	0	0
Web of Science	8	5	2	2
Scopus	13	5	7	5
Science Direct	24	2	22	0
Total	**46**	**13**	**31**	**7**

Following the steps described before, the summary in Table 2.1 shows that the Scopus database has the highest selection rate (38.46%). All selected papers for further analysis are enumerated in Table 2.2.

Table 2.2: Selected Papers

Author	Database	Journal/Conference
CERULLI *et al.* (2020)	SCOPUS	International Symposium on Combinatorial Optimization, ISCO
CAPOBIANCO *et al.* (2022)	WOS	Soft Computing
SOUTO *et al.* (2024a)	SCOPUS	IEEE Congress on Evolutionary Computation, CEC
JOVANOVIC e VOSS (2024)	SCOPUS	Computers and Operations Research, COR
VIEIRA *et al.* (2024)	SCOPUS	Journal of Heuristics
ZHAO e HIFI (2024)	SCOPUS	Computers and Industrial Engineering, CIE
ZHOU *et al.* (2025)	WOS	International Transactions in Operational Research

After we obtain a quantitative understanding of the writers in each database, it is important to consider the timeline of publications on the KPF. For example, the number of publications peaked in 2024, with four distinct works released that year.

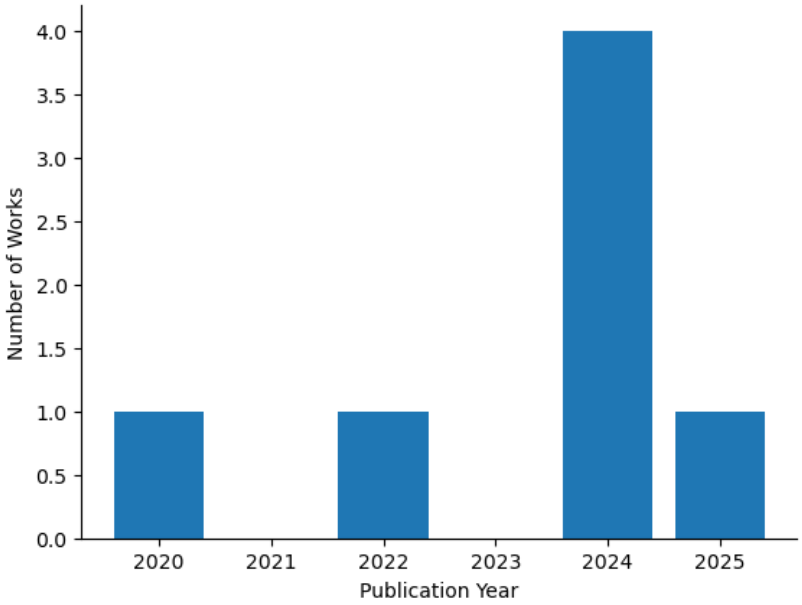

Figure 2.1: Number of works in literature per year

The absence of publications in 2021 and 2023 suggests that research on KPF was sporadic early on, followed by a concentration of studies in 2024. This pattern may reflect an emerging academic interest and the natural lag in the publication cycle. Overall, the histogram in Figure 2.1 emphasizes the increasing interest in the KPF topic in recent years, reflecting a growing community of researchers and the development of related techniques and methodologies.

2.2 Qualitative Analysis

This section presents a qualitative analysis to explore the relevant approaches to the KPF. Thus, we interpret the principal results and comprehensively describe the contributions of the existing literature.

CERULLI *et al.* (2020) introduces the Knapsack Problem with Forfeits (KPF), a variant of the classical 0/1 Knapsack Problem, where selecting specific pairs of items together incurs a forfeit cost rather than making the selection infeasible. The mathematical formulation, a Mixed-Integer Programming (MIP), is also proposed, generalizing the classical knapsack model by introducing binary variables to account for forfeit costs. Additionally, two heuristic algorithms are developed:

- *GreedyForfeits*: A constructive greedy algorithm that selects items based on profit-to-weight ratio while considering forfeits penalties.
- *CarouselForfeits*: An enhancement of the greedy approach using the Carousel

Greedy (CG) paradigm CERRONE *et al.* (2017), which iteratively refines initial greedy choices to improve solution quality.

The authors proposed benchmark instances and conducted experiments to compare their results with those obtained from running the MIP formulation on the CPLEX Solver. The *GreedyForfeits* heuristic reached solution gaps from the optimum value smaller than 15% in most cases for small instances (500 and 700 items). In contrast, *CarouselForfeits* significantly improved the quality of the solutions, reducing the solution gaps to between 1.79% and 6.36%, all while maintaining low computational times. Exact methods struggled to find optimal solutions for large instances, highlighting the need for efficient heuristics like CarouselForfeits. This heuristic demonstrates a substantial reduction in forfeits paid compared to traditional greedy approaches.

To address the difficulty of finding high-quality solutions, of the constructive heuristics proposed by CERULLI *et al.* (2020), a metaheuristic known as GA–CG Forfeits was introduced by CAPOBIANCO *et al.* (2022). This algorithm begins by initializing a population of randomly generated individuals. Parent solutions are selected over several iterations using a tournament selection mechanism. Subsequently, a child chromosome is created through a sequence of crossover and mutation operations. The offspring solutions are then refined using the *CarouselForfeits* heuristic, which reevaluates previous item selections to correct suboptimal choices. This refinement process may involve removing or swapping items to minimize penalty conflicts and maximize profit. Finally, the population is updated, and the stopping conditions are checked.

The proposed approach (GA-CG Forfeits) is the first methodology that combines a Genetic Algorithm and a *CarouselForfeits* for this problem. The GA–CG Forfeits outperforms prior heuristics, achieving a lower solution with an average gap of 2% compared to 14.53% in *GreedyForfeits* and 6.54% for *CarouselForfeits*. While more complex than pure greedy approaches, GA–CG Forfeits maintains acceptable runtimes under 2200s for large instances.

Realizing the advantages of the genetic algorithm and the room for improvement in the given literature, SOUTO *et al.* (2024a) introduces the Q-HBRKGA, a Hybrid Biased Random-Key Genetic Algorithm (BRKGA) integrated with Q-learning and Local Branching to solve KPF. The proposed Q-HBRKGA approach consists of:

- **Biased Random-Key Genetic Algorithm (BRKGA)** – A genetic algorithm with biased selection mechanisms to ensure high-quality solutions.
- **Q-Learning** – A reinforcement learning-based mechanism to dynamically guide the local search decision process.

- **Local Branching** – A mixed-integer programming (MIP)-based neighborhood search that refines high-quality solutions.

This hybrid metaheuristic approach (Q-HBRKGA) is the first integration of reinforcement learning (Q-learning) with BRKGA to solve KPF. The Q-learning agent dynamically adjusts the search process to determine whether to apply a local search procedure based on the solution's gap from the best-known solution. The performance evaluation of Q-HBRKGA against existing methods demonstrates superior results in solution quality and computational time.

Q-HBRKGA outperforms the GA-CG Forfeits, reducing the solution gap to less than 1% in most instances. Compared to *CarouselForfeits*, the improvement is even more pronounced, reducing the average of the mean solution gap from 9.30% to 0.37%. Regarding runtime, Q-HBRKGA remains competitive ($\leq$ 3 minutes in most cases) for large instances, faster than GA-CG while achieving better solution quality. A Wilcoxon Signed Rank Test also confirms that Q-HBRKGA significantly outperforms HBRKGA (version without Q-learning) in solution quality.

A substantially different approach from earlier metaheuristics is the Fixed Set Search Matheuristic (MFSS) proposed by JOVANOVIC e VOSS (2024). This method leverages a combination of:

- **Fixed Set Search (FSS)** – A population-based metaheuristic framework that learns from high-quality solutions and emphasizes subsets of frequently occurring elements JOVANOVIC *et al.* (2019).

- **Integer Programming (IP)** – Used for refining partial solutions derived from fixed sets.

- **Dynamic Fixed Set Construction** – Selects frequently occurring elements from high-quality solutions to guide the search.

MFSS iteratively enhances its solutions by fixing high-frequency items in a reduced problem space and solving subproblems efficiently with an IP solver. This process is particularly effective for large-scale KPF benchmark instances (e.g., up to 1000 items), balancing IP's exact optimization and FSS's adaptive learning.

The proposed method found better solutions than Q-HBRKGA. The researchers identified the optimal solution for the more straightforward benchmark instances, which was certified by the CPLEX Solver. For the more complex instances, the values obtained with the MFSS were superior to those found by CPLEX within a 3-hour time limit.

Another relevant work on KPF is the Iterated Local Search with Variable Neighborhood Descent (*ILS-VND*), proposed by VIEIRA *et al.* (2024). The *ILS-VND* algorithm includes:

- **Iterated Local Search (ILS)** - Uses perturbation and local search to escape local optima.
- **Variable Neighborhood Descent (VND)** - Applies multiple neighborhood structures systematically.
  - Swap-01 (Addition): Adds a single item.
  - Swap-10 (Removal): Removes a single item.
  - Swap-11 (Swap): Exchanges one selected item with another.
  - Swap-21 (Pair Removal and Addition): Replaces two items with a new one.
- **Tabu Search Mechanism** - Prevents immediate reversal of recent changes to improve search efficiency.
- **Efficient Data Structures**: Uses specialized arrays and heaps to speed up neighborhood exploration.

This paper's main contribution is a fast neighborhood exploration method that utilizes a filtering mechanism and a tabu list to enhance search efficiency. The authors' results indicate that this method outperforms the GA-CG Forfeits and Q-HBRKGA, achieving better solutions with a smaller gap. Compared to MFSS, the ILS-VND method only shows improvements in a few instances across the benchmark classes. However, a notable aspect of this approach is its runtime, which averaged around ten seconds for all benchmark instances, from the easiest to the hardest.

A new proposal, utilizing a novel approach to reinforcement learning, is presented by ZHAO e HIFI (2024). This proposal introduces a Reinforcement Learning-Driven Cooperative Scatter Search (RL-DCSS), which integrates Q-learning with Scatter Search (SS) to balance exploration and exploitation.

- **Initial Population Diversification** - A greedy constructive procedure generates an initial reference set, ensuring diversity and feasibility.
- **Path-Relinking with a Look-Ahead Strategy (PR-LAS)** - A trajectory-based method that iteratively refines solutions by linking high-quality solutions through a guided search.
- **Q-Learning-Based Dynamic Search Adjustment** - Determines the most effective perturbation strategy based on prior experience and adapts search neighborhood operators dynamically.
- **Tabu Search** - A tabu search mechanism prevents revisiting recent solutions while intensifying the search around promising regions.

- **Reference Set Update and Archive Management**: Maintains an adaptive reference set, filtering out redundant solutions and ensuring quality improvement over iterations.

The *RL-DCSS* approach is a cooperative scatter search framework enhanced with reinforcement learning, integrating machine learning techniques into combinatorial optimization, using reinforcement learning to select perturbation operators, and improving search efficiency.

The authors noted that the method establishes 22 new lower bounds identified in prior literature and matches the best-known solutions for benchmark instances. RL-DCSS outperforms previous heuristics and metaheuristics, setting new best-known results for multiple instances.

The work of ZHOU *et al.* (2025) introduces a Hybrid Evolutionary Search Method (HESM), offering several innovative features for the KPF. First, they combine a memetic algorithm—which relies on both population-based evolutionary search and local search enhancements—with an adaptive feasible and infeasible search. This adaptive mechanism dynamically toggles between searching in feasible and infeasible regions of the problem, modulating penalty values to guide the solver toward high-quality solutions.

To further refine candidate solutions, HESM uses a uniform crossover (UX). This operator preserves common items from top-performing solutions while introducing a bit of randomness, thus balancing intensification and diversification. Additionally, a specialized local optimization phase—AFITS—is included. This tabu-based local search continuously adjusts how it penalizes infeasibility, ensuring the algorithm can effectively escape poor local optima. An incremental streamlining evaluation technique rounds out the process by updating objective values in constant time whenever items are swapped in or out, thereby reducing computational overhead.

According to the authors, HESM outperforms the previous KPF solution approaches by finding 94 new lower bounds in the benchmark instances, making it the new state-of-the-art. They note that HESM finds superior solutions in 109 out of 120 instances and, in some cases, even surpasses the known optimal solutions as determined by CPLEX. However, it was not to include HESM in their subsequent comparative analyses, given that it reported a better "heuristic value" than the established optimum.

To consolidate these main approaches proposed for the KPF, Table 2.3 highlights the evolution from greedy heuristics to more sophisticated metaheuristic, hybrid, and matheuristic methods. Early works, such as (CERULLI *et al.*, 2020), (CAPOBIANCO *et al.*, 2022), focused on constructive heuristics and basic metaheuristics, while more recent methods, such as SOUTO *et al.* (2024a); VIEIRA *et al.* (2024),

Authors	Greedy Heuristic	Metaheuristic	Hybrid Method	Matheuristic
CERULLI *et al.* (2020)	X			
CAPOBIANCO *et al.* (2022)	X	X		
SOUTO *et al.* (2024a)	X	X	X	
JOVANOVIC e VOSS (2024)		X	X	X
VIEIRA *et al.* (2024)	X	X	X	
ZHAO e HIFI (2024)	X	X	X	
ZHOU *et al.* (2025)	X	X	X	
Our Method	X	X	X	X

Table 2.3: Classification Methods for KPF

introduced hybrid techniques by combining genetic algorithms, local search, and reinforcement learning to improve solution quality. The JOVANOVIC e VOSS (2024) represents a shift towards matheuristics, leveraging mathematical programming for optimization. Recent advancements, such as ZHAO e HIFI (2024); ZHOU *et al.* (2025) integrating machine learning and adaptive search strategies, further enhance performance on large-scale instances.

In the context of this literature review, our HDM-ILS-VND method aligns with recent work that goes beyond simple heuristic strategies to combine multiple advanced techniques, as better described in Chapter 4. Specifically, HDM-ILS-VND extends the Iterated Local Search with Variable Neighborhood Descent (ILS-VND) of VIEIRA *et al.* (2024) by introducing two additional components: frequent pattern mining and a Local Branching (LB) phase. By doing so, it inherits the advantages of ILS-VND—namely, its structured exploration of local neighborhoods—while leveraging data mining insights from elite solutions. These insights allow the algorithm to identify high-value item patterns that appear consistently in top solutions, thus focusing the search on more promising regions of the solution space. Meanwhile, LB is used to intensify the search for good solutions. In this sense, our work proposes an evolution seen in other methods such as the matheuristic approach of JOVANOVIC e VOSS (2024) or the reinforcement learning strategies of ZHAO e HIFI (2024) by emphasizing both data-driven decision-making focused on local refinement under the ILS-VND paradigm.

Chapter 3

Problem Definition

This chapter defines the Knapsack Problem with Forfeits (KPF), which extends the classical 0–1 Knapsack Problem by introducing "soft conflicts" when selecting specific item pairs together incurs a penalty. Section 3.1 shows the problem generalization from the Knapsack Problem with Conflicts (KPC), which models conflicts as hard constraints based on a conflict graph. Originating from graph theory and combinatorial optimization, these constrained knapsack variants have been extensively studied, leading to exact and heuristic approaches. The mathematical problem formulation, described in Section 3.2, is a quadratic 0–1 optimization problem due to penalty terms, but it can be linearized using auxiliary variables to enable solution via Mixed-Integer Linear Programming (MILP) solvers, that is, a specialized computational tool designed to address optimization problems characterized by both continuous and discrete variables. In these problems, some decision variables can take real values within specified bounds, while others are restricted to integer values. This combination allows for the modeling of complex scenarios involving decision-making, as in KPF.

3.1 Problem Motivation

The idea of adding conflict (incompatibility) constraints to the knapsack problem appeared in the literature by the early 2000s. YAMADA *et al.* (2002) was the first to explicitly study the disjunctively constrained knapsack problem, introducing a branch-and-bound algorithm for it. Throughout the 2000s and 2010s, several works addressed what they called the "knapsack problem with conflict graph". Researchers like HIFI e MICHRAFY (2006) HIFI e MICHRAFY (2007), HIFI e OTMANI (2012) proposed both metaheuristics and exact algorithms for Knapsack Problem with Conflicts (KPC), including reactive local search and specialized branch-and-bound techniques that leverage the conflict graph structure.

The KPC extends the 0–1 knapsack by forbidding certain item combinations.

Each conflict is a pair of items that cannot be in the knapsack together (modeled by an edge in a conflict graph) as described by PFERSCHY e SCHAUER (2009). The decision variables are the same (0–1 for each item), and a capacity constraint on weight still applies, but any pair of items connected by a conflict edge cannot both have value 1 (expressed by constraints like $x_i + x_j \leq 1$ for each conflicting pair).

Thus, the KPC feasibility requires ensuring the knapsack capacity and avoiding all forbidden pairs. The KPF's formulation is a relaxation of KPC: if one were to assign a sufficiently large forfeit cost (penalty) to each conflict pair, the KPF would effectively disallow those pairs, reducing them to the KPC.

PFERSCHY e SCHAUER (2009) made an important contribution, providing a pseudo-polynomial algorithm for KPC on special graph classes. They also surveyed earlier approaches and highlighted the connection to the independent set problem. Over time, the KPC problem became a well-studied generalization of knapsack, with further improvements in exact methods and heuristics for larger instances. The concept of conflict constraints also migrated to related problems like bin packing and scheduling, where items/jobs couldn't coexist in the same bin or time slot – reinforcing the general theme of "combinatorial optimization under conflict constraints".

The KPF is a more recent development, evolving from the conflict-constraint literature. Introduced by CERULLI *et al.* (2020), the KPF generalizes the conflict knapsack by relaxing the incompatibility: instead of forbidding a conflicting pair, it imposes a profit reduction (forfeit cost) if the forfeit pair is taken together. This concept of "soft conflicts" was motivated by scenarios where disallowing all conflicts might be too restrictive or suboptimal, and tolerating certain conflicts at a cost could be beneficial. The initial KPF work provided a formal definition, a MIP formulation, and heuristic algorithms.

As a generalization of the 0–1 knapsack, the KPF inherits the NP-hard complexity, as established in recent studies by (CAPOBIANCO *et al.*, 2022; JOVANOVIC e VOSS, 2024). Consequently, metaheuristic approaches have emerged prominently due to the computational challenge of KPF solving. The development of the KPF is essentially part of the trend of adding constraints to classical problems, and it is based on the knapsack problem and quadratic optimization literature.

3.2 Mathematical Formulation

A starting point for modeling the Knapsack Problem with Forfeits (KPF) is to establish some important definitions introduced by the original authors, as mentioned in the previous section. Let n be the number of items composing the set $\mathcal{I}$. Each item $i \in \mathcal{I}$ has an associated profit $p_i > 0$ in the set $\mathcal{P}$ and positive weight $w_i > 0$ in the set $\mathcal{W}$, where $i = 1, ..., n$. Let $\mathcal{F}$ be a set of l distinct forfeit pairs:

$$\mathcal{F} = \{F_k\}_{k=1,2,\ldots,l},\ F_k \subseteq \mathcal{I},\ |F_k| = 2\ \forall F_k \in \mathcal{F} \tag{3.1}$$

Each element in $\mathcal{F}$ is a two-member subset of $\mathcal{I}$ with a corresponding forfeit cost $d_k > 0, k = 1, 2, ...l$ in the set $\mathcal{D}$. Let $b > 0$ represent the upper limit on the weight of selected items (capacity). The KPF can then be formulated as:

$$\begin{aligned}
\max \quad & \sum_{i \in \mathcal{I}} p_i x_i - \sum_{(i,j) \in \mathcal{F}} d_{ij} x_i x_j \\
& \sum_{i \in \mathcal{I}} w_i x_i \leq b && (3.2) \\
& x_i \in \{0, 1\} \qquad \forall i \in \mathcal{I} && (3.3)
\end{aligned}$$

where the binary variables, represented as x_i, indicate whether item i from the set I is included in the knapsack. Constraint (3.8) ensures that the maximum capacity of the knapsack is not exceeded. The objective function is designed to maximize profit by minimizing forfeit costs.

Linearization into a Mixed-Integer Linear Program

To handle the quadratic penalty terms with standard MILP solvers, one typically linearizes the product of binary variables $x_i x_j$. This involves introducing an auxiliary variable—let's denote it y_{ij}—for each forfeit pair (i, j). The idea is that whenever x_i and x_j are both 1, y_{ij} must be 1; otherwise, $y_{ij} = 0$. We can enforce this via linear constraints such as:

$$y_{ij} \geq x_i + x_j - 1, \tag{3.4}$$

$$y_{ij} \leq x_i, \tag{3.5}$$

$$y_{ij} \leq x_j \tag{3.6}$$

and

$$y_{ij} \in [0, 1] \tag{3.7}$$

Considering Constraints (3.4), (3.7) and the negative contribution $-d_{ij} x_i x_j$ in the objective function. If $x_i + x_j = 2$ then y_{ij} must be ≥ 1, which practically forces $y_{ij} = 1$. If $x_i + x_j$ is 0 or 1, then the solver will tend to keep $y_{ij} = 0$, because $y_{ij} = 1$ would worsen the objective (it adds a penalty). Hence, Constraints (3.5) and (3.6) are implicitly enforced via the combination of the linear constraint (3.4), and the sense of the optimization process (maximization).

That is why the original paper's authors (CERULLI *et al.*, 2020) do not include separate inequalities $y_{ij} \leq x_i$ and $y_{ij} \leq x_j$ because the Constraints (3.4) and the sense of the optimization process (maximization) are enough to drive y_{ij} to 0 or 1 in the right situations.

The objective function then becomes purely linear in terms of x_i and y_{ij}:

$$\begin{aligned}
\max \quad & \sum_{i \in \mathcal{I}} p_i x_i - \sum_{(i,j) \in \mathcal{F}} d_{ij} y_{ij} \\
& \sum_{i \in \mathcal{I}} w_i x_i \leq b && (3.8) \\
& y_{ij} \geq x_i + x_j - 1, && (3.9) \\
& y_{ij} \in [0,1] \qquad \forall (i,j) \in \mathcal{F} && (3.10) \\
& x_i \in \{0,1\} \qquad \forall i \in I && (3.11)
\end{aligned}$$

Firstly introduced by CERULLI *et al.* (2020), in this linearized version, the problem can now be solved using a standard mixed-integer linear solver. While the extra variables y_{ij} make the formulation potentially larger, they transform the model into a purely linear one, permitting branch-and-bound, cutting planes, and other MILP techniques.

This linearization strategy is common in many combinatorial optimization problems that involve the product of binary variables. It allows using existing MILP solvers and well-established solution techniques, even though the resulting model can still be challenging for large-scale instances.

Chapter 4

Proposed Method

The Hybrid Data Mining ILS-VND (HDM-ILS-VND) approach extends the work of VIEIRA *et al.* (2024)—focused initially on an Iterated Local Search with Variable Neighborhood Descent (ILS-VND)—by adding frequent pattern mining and a Local Branching (LB) phase. This combination leverages the effectiveness of ILS-VND while exploiting data-mining insights from elite solutions and using local branching to refine particularly promising solutions.

This chapter is organized to explain the main components of the proposed methodology. In Section 4.1, we provide a brief overview of the approach proposed by VIEIRA *et al.* (2024). Following that, in Section 4.2, we detail the entire algorithm and describe how each component works together. The primary focus of this master's thesis is presented in Sections 4.3 and 4.4, which discuss the integration of the mining model with the ILS-VND approach and the enhanced Local Branching techniques designed to improve a given feasible solution.

4.1 ILS-VND

The proposed Iterated Local Search with Variable Neighborhood Descent (ILS-VND) developed by VIEIRA *et al.* (2024) integrates randomized perturbations with a structured local search. It begins by building a greedy-based initial solution and refining it through multiple operators that add, remove, or exchange items in a first-improvement manner. Specialized data structures are employed to improve neighborhood exploration, ensuring fast feasibility checks and cost adjustments. Periodic random perturbations help the algorithm escape local optima, while an acceptance criterion, allowing occasional acceptance of worse solutions, provides a balance of intensification and diversification.

4.1.1 Greedy Forfeits

The initial solution $\mathcal{S}$ of the HDM-ILS-VND is generated using a greedy heuristic proposed by CERULLI *et al.* (2020). The idea behind Algorithm 1 is to greedily maximize the marginal profit-to-weight ratio $\frac{p'_i}{w_i}$, where p'_i already accounts for possible forfeits if the item conflicts with any elements of the current solution. By ranking items according to their *adjusted* ratio, the algorithm balances raw profit against penalty costs. Thus, each iteration targets adding the item with the most favorable benefit. This procedure terminates when no feasible candidate can be inserted into $\mathcal{S}$, ensuring the solution remains admissible under the knapsack's capacity b.

Algorithm 1: Greedy Forfeits

```
Data: 𝓘, 𝓦, 𝓟, b, ℱ, 𝒟
begin
    𝒮 ← ∅
    b_res ← b
    while 𝓘 \ 𝒮 ≠ ∅ do
        X_iter ← ∅
        for i ∈ 𝓘 do
            if w_i ≤ b_res and i ∉ 𝒮 then
                X_iter ← X_iter ∪ {i};
            end
        end
        if X_iter = ∅ then
            return 𝒮
        end
        for i ∈ X_iter do
            p'_i ← p_i
            for F_k = {i, j} ∈ ℱ do
                if j ∈ 𝒮 then
                    p'_i ← p'_i − d_k
                end
            end
            ratio_i ← p'_i / w_i
        end
        i* ← argmax[ratio]
        if ratio_i* < 0 then
            return 𝒮
        end
        𝒮 ← 𝒮 ∪ {i*}
        b_res ← b_res − w_i*
    end
    return 𝒮
end
```

This heuristic is elucidated in the Algorithm 1 that begins by setting the solution

$\mathcal{S}$ to empty and initializing the residual capacity to b_{res}. It then repeatedly identifies a set X_{iter} containing those items $i \in \mathcal{I} \setminus \mathcal{S}$ that can still fit, whose weight $w_i \leq b_{res}$. If X_{iter} is empty, the algorithm terminates, returning $\mathcal{S}$. Next, each candidate $i \in X_{iter}$ receives an *adjusted profit* p'_i, computed by subtracting from its nominal profit p_i any penalty d_k associated with a conflicting item j already in $\mathcal{S}$. The ratio $\frac{p'_i}{w_i}$ is then calculated; if the maximum ratio is negative, the process ends. Otherwise, the algorithm selects the item i^* with the highest ratio, adds i^* to $\mathcal{S}$, and deducts w_{i^*} from b_{res}.

4.1.2 Local Search (VND)

Once the greedy solution is built, a multi-neighborhood local search is performed using Variable Neighborhood Descent (VND) (DUARTE *et al.*, 2018). A standard local search uses just one neighborhood. The VND instead uses multiple neighborhoods in a deterministic order or arrangement. The main idea behind the VND is that a solution which is a local optimum for one neighborhood may not be a local optimum for another. By switching neighborhoods whenever you get stuck in the current one, you can "escape" from local minima more effectively than using just a single neighborhood.

The original authors of the ILS-VND defined four neighborhoods, executed in two phases and relying on a first-improvement approach, as below:

1. **Swap-01**: Attempts to add one item not in the solution, as long as capacity allows and the addition generates a better solution cost.

2. **Swap-10**: Attempts to remove a single item from the solution if that removal leads to an improved solution (for example, removing an item with a high penalty contribution).

3. **Swap-11**: Tries to remove one item from the solution and add a different one (not currently in the solution).

4. **Swap-21**: Removes two items from the solution and introduces a single new item if capacity permits.

They divide the local search into two phases:

1. Phase One

    - Start with swap-11 and look for an immediate first-improvement move.
    - If no improvement is found with swap-11, then test swap-21. If swap-21 finds a better solution you do that move.

- Each time you do make an improvement (via either swap-11 or swap-21), you update your current solution and go back to checking swap-11 again

2. Phase Two

   - After phase one stops producing improvements, you proceed to these smaller moves. Start with swap-01 and look for an immediate first-improvement move.
   - If no improvement is found with swap-01, then test swap-10. If swap-10 finds a better solution you do that move.
   - Each time you do make an improvement (via either swap-01 or swap-10), you update your current solution and go back to checking swap-01 again. You stop phase two if neither swap-01 nor swap-10 can improve the solution.

Efficient Neighborhood Search

To efficiently explore these neighborhoods, especially for larger KPF instances, The algorithm employs several data structures to quickly update the solution's cost and to check feasibility when items are added or removed:

1. **Solution array**: Stores all items in a single array: the first part of the array contains the items in the solution, while the second contains the items out of the solution. This structure enables quick checks and fast $\mathcal{O}(1)$ item insertions/removals at the boundary.

2. μ **array**: For each item $i, \mu[i]$ tracks the net impact on the objective function if i is added or removed. Whenever an item i enters or leaves the solution, the μ values for that item's neighbors (items with which it forms a penalized pair) are updated accordingly in $\mathcal{O}(\Delta)$ time, where Δ is the maximum number of conflicts for any single item.

3. **Tabu list**: A simple array that stores the iteration number at which an item can be removed from the solution again. After an item i is added, it stays "tabu" for T_T iterations to prevent it from being removed immediately. This helps escape local optima and reduces the number of potential removal moves to explore.

4. **Heaps (H_S, H_X)**: Two max-heaps based on item weight:

   (a) H_S: A max-heap containing all items in the solution.

   (b) H_X: A max-heap containing all items outside the solution.

Primarily used in the swap-21 neighborhood to quickly find heavier (or lighter) items and control a filtering mechanism.

Neighborhood Filtering Mechanism

The swap-21 operator can be computationally expensive because it looks at combinations of two items in the solution and one item out, potentially leading to $\mathcal{O}(n^3)$ complexity. To address this problem, VIEIRA *et al.* (2024) imposes a rule that at least one of the removed items has a "sufficiently large" weight to free up enough capacity for any item out of the solution. Specifically, it compares the chosen item's weight with a threshold proportional to the heaviest item outside. Enforcing this rule skips many unpromising forfeits pairs, significantly reducing the time spent checking swap-21.

4.1.3 Perturbation

During each iteration loop of ILS, once a local optimum in the space defined by the intersection of neighborhood structures is found, a perturbation is applied to escape that local optimum.

The original authors have a parameter c_1 specifying how many items to add simultaneously. Even if the knapsack is full or almost full, the algorithm inserts c_1 different items outside the solution without initially checking capacity. After these insertions, the solution may exceed the knapsack's capacity. Thus, they remove items randomly from the already-present items until the knapsack becomes feasible again.

Adding multiple items disrupts the current configuration for exploring new solution neighborhoods. Once the capacity is restored via random removals, the solution is far from its old state, often allowing subsequent local search steps to discover better optima.

4.1.4 Acceptance Criterion

After local search and perturbation, a candidate solution s' may or may not be better than the incumbent s.

In this acceptance criterion, whenever a candidate solution s' has a higher solution cost than the incumbent s, the algorithm accepts it immediately. It then resets or reduces the counter k, reflecting that a constructive improvement occurred. Moreover, if s' surpasses the best solution $s*$ or improves upon the last local best cost, additional reductions in k happen. The rationale is that truly better solutions should "reward" the search by lowering the threshold for the next acceptance.

If s' does not improve over s, the algorithm checks whether k has stayed below a certain threshold. If so, k increments (the solution is effectively rejected, but search continues). However, if k exceeds that threshold, the algorithm applies a strong perturbation to the current solution s and resets k. This forces the search to move away from unproductive regions and helps prevent getting stuck at suboptimal points

Algorithm 2: Acceptance Criterion

```
1  Data: s', s, k, s*, local_cost
2  begin
3      if cost(s') > cost(s) then
4          k ← 1;
5          s ← s';
6          if local_cost < cost(s) then
7              k ← k − |s|/c2;
8              local_cost ← cost(s);
9          end
10         if cost(s*) < cost(s) then
11             s* ← s;
12             k ← k − (|s| × c3);
13         end
14     end
15     else if k ≤ |s|/2 then
16         k ← k + 1;
17     end
18     else
19         local_cost ← cost(s);
20         s ← perturb(s, c4);
21         k ← 1;
22     end
23 end
```

Therefore, this acceptance criterion approach allows occasional acceptance of worse solutions to avoid stagnation and balance intensification and diversification.

4.2 HDM-ILS-VND Algorithm

In this section, we present the HDM-ILS-VND approach as illustrated in Figure 4.1. The algorithm starts with a greedy heuristic that yields a feasible initial solution. This solution undergoes a VND-based local search to find local optimal solutions. The algorithm then applies a perturbation step to escape local optima by temporarily adding items (possibly violating capacity constraints) and removing enough items afterward to restore feasibility.

Within each main iteration loop, a second round of VND is applied in the current solution; after that, the obtained solution may be stored in an elite set following these significant steps. Once this elite set has reached its maximum capacity, the algorithm triggers frequent itemset mining. The identified patterns are then incorporated into a small mixed-integer model (here referred to as the "mining model") to search in the space defined by the combination of mined patterns. Once the criterion of accepting a solution is satisfied, the algorithm verifies if a stop criterion has been achieved. If not, the algorithm goes to the next loop iteration, starting again from the perturbation.

If the stop criterion has not been met, the algorithm repeats this cycle—perturbation, local search, and data mining. When the stop criterion is satisfied, a final local branching refinement is performed on the best solution discovered, and the search terminates. The description of this algorithm is illustrated in Figure 4.1.

In essence, HDM-ILS-VND combines greedy heuristics, local search, data-mining insights, and local branching to explore the solution space more thoroughly while maintaining practical run times.

4.3 Data Mining Integration

Having established the ILS-VND-inspired algorithm's fundamental components, we now proceed to a crucial methodology step: utilizing data mining for frequent pattern discovery within the ILS-VND. To the best of our knowledge, this is the first time that a data mining approach has been applied to the iterated local search to discover patterns for the knapsack problem with forfeits.

4.3.1 Data Mining

Frequent Itemset Mining (FIM) is a significant research area in Data Mining (DM) that involves discovering patterns and rules within databases WITTEN *et al.* (2005); LIU (2012); HAN *et al.* (2022). A key concept in data mining is association rules, which describe how often certain items appear together within a given dataset. Following the definitions present in BARBALHO *et al.* (2013); PLASTINO *et al.* (2014); GUERINE *et al.* (2016), given a set of items $I = \{i_1, i_2, ...i_n\}$ and the subsets $X \subseteq I$ and $Y \subseteq I$, where X and Y are nonempty sets, data mining algorithms produce an association rule $X \Rightarrow Y$, which describes a certain pattern. Following the definitions proposed by RIBEIRO *et al.* (2006), D is a set of transactions over I, where each transaction t is a subset of $I(t \subseteq I)$. The association rule is found in database D with support s and confidence c if, respectively, $s\%$ of transactions in

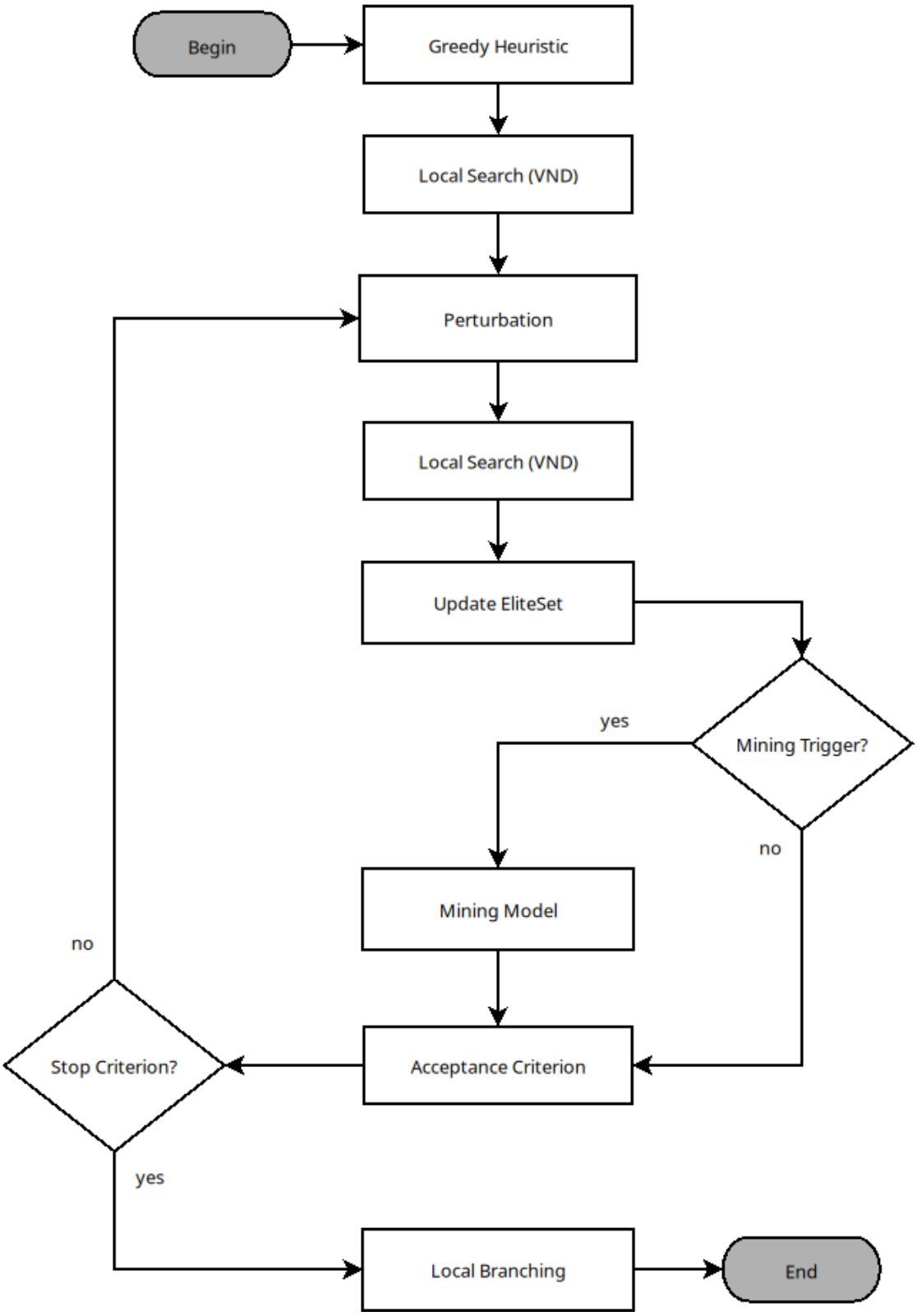

Figure 4.1: HDM-ILS-VND Algorithm

D contain $X \cup Y$, and c % of transactions in D that contain X also contain Y.

The process of extracting association rules from transactional databases is often divided into two phases. The terms sup_{min} and sup_{conf} represent minimum support and confidence, respectively. The first phase, which is called Frequent Itemset Mining (FIM), consists of identifying all frequent item sets, that is, sets of items that occur in at least sup_{min}% of the database transactions. The second phase yields, for each frequent item set Z, all association rules $A \Rightarrow B$ with confidence greater than or equal to sup_{conf}, such that $A \subset Z$, $B \subset Z$, and $A \cup B = Z$. The FMI phase demands more computational power than the second (GOETHALS e ZAKI, 2003).

This work leverages a variant of the FPmax* algorithm to mine frequent patterns from the elite set of solutions. FPmax* is designed to extract maximal frequent itemsets efficiently. These are the largest sets of items that appear frequently together in a dataset, where no superset of these items meets the minimum frequency threshold.

4.3.2 Mining Model

After the data mining phase, the proposed algorithm uses the mined patterns to construct a MIP model that takes advantage of the discovered knowledge. To do that, the mined patterns are stored in a binary pattern matrix and the patterns are used to reduce the search space. Instead of searching the original solution space, here we focus on the subspace defined by the combination of the mined patterns, as described below:

$$
\begin{aligned}
\max \quad & \sum_{i \in \mathcal{I}} p_i x_i - \sum_{(i,j) \in \mathcal{F}} d_{ij} y_{ij} \\
& \sum_{i \in \mathcal{I}} w_i x_i \leq b && (4.1) \\
& y_{ij} \geq x_i + x_j - 1, && (4.2) \\
& \sum_{p \in \mathcal{P}} z_p \geq 1 && (4.3) \\
& x_i \geq R_i^p z_p \qquad \forall i \in I, \forall p \in \mathcal{P} && (4.4) \\
& y_{ij} \in [0,1] \qquad \forall (i,j) \in \mathcal{F} && (4.5) \\
& z_p \in \{0,1\} \qquad \forall p \in \mathcal{P} && (4.6) \\
& x_i \in \{0,1\} \qquad \forall i \in I && (4.7)
\end{aligned}
$$

where the binary decision variable, defined as z_p (4.6), indicates if a pattern p from a set $\mathcal{P}$ of mined patterns was chosen. R_i^p is defined as the binary pattern matrix, which indicates if item i is in the pattern p or not.

The Constraint (4.3) ensures that at least one pattern from the set of mined

patterns $\mathcal{P}$ is selected. By requiring the sum to be at least 1, the model guarantees that the solution will incorporate at least one mined pattern. The family of Constraints (4.4) ties the selection of a pattern to the inclusion of its items in the solution. Here, x_i is a binary variable that indicates whether item i is included in the solution. The binary pattern matrix R_i^p is 1 if item i is part of pattern p (and 0 otherwise). If $z_p = 1$ (pattern p is selected) and $R_i^p = 1$ (item i is in pattern p), then the constraint forces $x_i \geq 1$, meaning x_i must be 1 (item i is included). If $z_p = 0$ (pattern not selected) or $R_i^p = 0$ (item i is not in the pattern), the constraint is trivially satisfied.

4.3.3 DM-ILS-VND

After grasping the concept of the mining model, the mining model is constructed and solved during the main Iterated Local Search with Variable Neighborhood Descent (ILS-VND) loop using the Mixed Integer Linear Programming (MILP) Solver. As illustrated in Figure 4.1, following the local search, the incumbent solution generated is added to the elite set. This elite set is a container that stores the 15 best solutions produced by the ILS-VND algorithm to mine frequent patterns.

Since calling the mining model is computationally expensive, it is called 4 times during the main loop. The newly found solution from the mining model is then subjected to acceptance and stopping criteria to determine whether it should be accepted. If so, the loop continues; otherwise, it terminates based on the criteria set.

4.4 Local Branching

In the final stage of our proposed methodology, when DM-ILS-VND is hybridized with a Local Branching (LB) technique to enhance a high-quality solution, we refer to it as Hybrid Data Mining ILS-VND (HDM-ILS-VND).

The Local Branching was proposed by FISCHETTI e LODI (2003) to navigate through the search space and generate high-quality solutions by leveraging a mixed-integer programming (MIP) solver as a black box. The LB incorporates linear inequalities into the model, defining neighborhoods for exploration. Noteworthy research in the literature, including the works of GONZALEZ *et al.* (2016); GONZALEZ e BRANDAO (2018), SOUTO *et al.* (2024a) highlight the effectiveness of this approach, demonstrating its ability to yield highly favorable results.

Considering the KPF, given a feasible solution $s \in P$, where P delineates the polyhedron defined by Constraints (3.8) - (3.10). This approach aims to improve a given feasible solution by adding the following LB constraint:

$$\sum_{i \in X | \bar{x}_i = 0} x_i + \sum_{i \in X | \bar{x}_i = 1} (1 - x_i) \leq \Delta, \tag{4.8}$$

where Δ is a positive integer representing the number of variables x_i, $i \in X$, that can change their value from one to zero, or vice versa.

Chapter 5

Computational Experiments

The proposed method was implemented in a specific programming language and tested using benchmark instances to evaluate its effectiveness. This chapter details the implementation process, the instances used, the primary results obtained, and the insights gained from the experiments.

The chapter is organized into four sections. Section 5.1 summarizes our experiment's implementation and parameters. In Section 5.2, we provide a comprehensive description of the instances we analyzed and explore the key findings of our approach. Section 5.3 presents a statistical comparison between ILS-VND and HDM-ILS-VND. Finally, in Section 5.4, we compare our findings with relevant literature and examine computational experiments, focusing on performance improvements and potential gains and losses.

5.1 Implementation Details and Parameters

The proposed method, HDM-ILS-VND, was implemented in C++ using the GCC compiler version 11.4.0 and IBM CPLEX version 22.1.1. Since HDM-ILS-VND is a stochastic algorithm, experiments were conducted ten times for each instance on an Intel 11th Gen Core i7-11800H processor, which operates at a frequency of 4.60 GHz, paired with 16 GB of RAM.

Before conducting the experiments, it is essential to provide information about the parameters used. For the ILS-VND algorithm, we used the same parameters the authors proposed in VIEIRA *et al.* (2024). We conducted a series of experiments to tune its parameters. In these experiments, selected parameters were varied while the remaining parameters were kept fixed. The empirically determined values are summarized in Table 5.1. The parameters empirically selected were the Local Branching (LB) parameter (Δ), elite set size for data mining (e), number of times data mining is triggered (dm_t), the maximum number of ILS-VND iterations (max_{iter}), and support (s) for the data-mining phase.

Table 5.1: Values tested and selected for each parameter of the HDM-ILS-VND

Parameters	Tested Values	Selected Values
dm_t	$(2; 4; 6; 8)$	4
Δ	$(\frac{\lvert\mathcal{I}\rvert}{4}; \frac{\lvert\mathcal{I}\rvert}{2}; \frac{3\lvert\mathcal{I}\rvert}{4})$	$\frac{3\lvert\mathcal{I}\rvert}{4}$
max_{iter}	$(500; 10000; 1500; 2000)$	2000
e	$(5; 10; 15; 20; 25)$	15
s	$(0.10 \times \lvert\mathcal{I}\rvert; 0.30 \times \lvert\mathcal{I}\rvert; 0.50 \times \lvert\mathcal{I}\rvert)$	$0.10 \times \lvert\mathcal{I}\rvert$

5.2 Benchmark Instances

Benchmark instances are crucial in evaluating and comparing the Knapsack Problem with Forfeits (KPF) solution methods. Since KPF extends the classical 0/1 Knapsack Problem by introducing forfeit penalties for selecting incompatible items, the complexity of instances increases significantly.

The main benchmark instances for KPF can be categorized into three major sets:

- "O" Instances (Original Set) – Introduced by CERULLI *et al.* (2020).
- "LK" Instances (Large Knapsack) – Introduced by CAPOBIANCO *et al.* (2022).
- "MF" Instances (More Forfeits) – Also introduced by CAPOBIANCO *et al.* (2022).

Each set of instances has 40 instances, leading to 120 benchmark instances widely used in the literature. The O set of instances consists of problems with item sizes of 500, 700, 800, and 1000. The knapsack capacity is $b = 3n$, where n is the number of items. Each instance includes $l = 6n$ forfeit pairs. The profits (p) are randomly generated within the range [5, 25], while the weights (w) are chosen randomly in the range [3, 20]. In addition, forfeit costs associated with pairs of incompatible items are assigned randomly in intervals [2, 15]. These characteristics make the O set a balanced benchmark, posing a moderate challenge for both heuristic and exact optimization methods.

The LK set of instances consists of the same item sizes. The knapsack capacity is $b = 5n$, with $l = 6n$ forfeit pairs. The profits (p), weights (w), and forfeit costs are selected in the same ranges. These instances were designed to test the effect of a larger knapsack capacity, meaning that more items can be selected before reaching the weight constraint. This allows for more diverse, feasible solutions, making it a valuable benchmark for testing metaheuristics and hybrid methods. The challenge in these instances lies in the increased solution space, where an efficient algorithm must balance item selection and penalty management while navigating a larger set of feasible solutions.

The MF set of instances has the same number of items as the previous sets. The knapsack capacity is $b = 3n$, with $l = 8n$ forfeit pairs. The profits (p), weights (w), and forfeit costs are selected in the same ranges. These instances are the most challenging among the three sets because of the higher number of forfeit pairs. This means that more penalties must be carefully managed, increasing the importance of intelligent search strategies. These instances are especially difficult for greedy heuristics, as naive selection leads to excessive forfeit costs.

Recent studies have used these instances to test different algorithms, revealing notable trends:

- Greedy heuristics (*GreedyForfeits*, *CarouselForfeits*) perform well on small instances but degrade on larger ones.
- Metaheuristic approaches (*GA-CG, ILS-VND, RL-DCSS*) significantly outperform greedy methods, especially on MF instances.
- Hybrid and Matheuristic approaches (*MFSS, HESM*) achieve the best results, consistently setting new best-known solutions.
- Exact solvers (CPLEX) remain competitive but are impractical for large n due to exponential runtime growth.

The latest methods (*ILS-VND, RL-DCSS, MFSS*) have demonstrated state-of-the-art performance, setting new lower bounds for instances O, LK, and MF.

5.3 ILS-VND versus HDM-ILS-VND

To evaluate the benefits of the Data Mining phase in the ILS-VND approach proposed by VIEIRA *et al.* (2024), this section provides a detailed comparison of the experiments conducted with ILS-VND on our machine and the enhanced version of HDM-ILS-VND.

5.3.1 Experiments Results

After selecting the parameters and determining the benchmark instances to be run in the algorithm, we conducted experiments to compare the results obtained from HDM-ILS-VND with those from ILS-VND, as well as with the best results available in the literature. A critical metric, referred to as the solution gap, was calculated using the following formula:

$$\text{gap} = 1 - \left(\frac{Sol_{heuristic}}{Sol_{CPLEX}} \right)$$

In this equation, $Sol_{heuristic}$ represents the cost of the heuristic solution used, while Sol_{CPLEX} indicates the cost associated with the best solution obtained through CPLEX with a 3-hour time limit.

We conducted ten trials for each of the 120 benchmark instances of the KPF. The cost of the best solution found during these experiments is presented in the following tables. According to CAPOBIANCO *et al.* (2022), the best solution identified by the CPLEX Solver, within a 3-hour time limit, was optimal for the following instances.

- All "O" instances of 500 items.
- All "O" instances except two ($n = 700$, $id = 2$ and $n = 700$, $id = 9$).
- Only a single "O" instances ($n = 800, id = 8$).

Experiments on "O" instances

As shown in Table 5.2, the HDM-ILS-VND matches or improves upon the ILS-VND solutions in most cases, improving the results for 36 instances. When HDM-ILS-VND does not improve the solution, it typically matches the quality of ILS-VND, in this case, just four instances. Although the run times for HDM-ILS-VND are often longer due to an extended data mining phase, the method found a better or equal solution value for all instances for this instance group, as illustrated in the table mentioned.

Experiments on "LK" instances

Table 5.3 presents a detailed comparison between ILS-VND and HDM-ILS-VND on the "LK" instances. Some key observations may include the fact that HDM-ILS-VND delivers a superior objective value (or lower gap) in 21 out of the 40 instances. Although it does not dominate every instance, its performance is competitive: the average gap is -0.34% for HDM-ILS-VND versus -0.30% for ILS-VND.

Despite not outperforming ILS-VND in every case, HDM-ILS-VND consistently finds better solutions in a majority of instances, suggesting a relatively low risk of obtaining a worse solution. The improved solutions obtained by HDM-ILS-VND often require higher computational times. Thus, the choice between methods will depend on whether the incremental quality gains justify the additional time expenditure in practical applications.

Overall, while HDM-ILS-VND is less dominant on the "LK" dataset compared to the "O" instances, it remains a valuable alternative, especially in contexts where even slight improvements in the solution are critical.

Table 5.2: O instances comparison

Instances		**ILS-VND**			**HDM-ILS-VND**		
# itens	Id	Best Solution	Time (s)	Gap (%)	Best Solution	Time (s)	Gap (%)
500	1	2625	10,06	0,04	**2626**	185,51	0,00
	2	**2660**	10,07	0,00	**2660**	182,27	0,00
	3	2504	10,06	0,48	**2516**	185,41	0,00
	4	2553	10,06	0,12	**2556**	188,13	0,00
	5	2624	10,07	0,04	**2625**	185,05	0,00
	6	**2615**	10,06	0,00	**2615**	189,39	0,00
	7	2626	10,06	0,04	**2627**	210,18	0,00
	8	2556	10,06	0,00	**2556**	71,69	0,00
	9	2613	10,07	0,00	**2613**	127,78	0,00
	10	2555	10,07	0,12	**2558**	185,47	0,00
Mean		2593,1	10,06	0,08	**2595,2**	171,09	0,00
700	1	3583	10,18	0,17	**3589**	183,54	0,00
	2	3415	10,15	0,26	**3424**	215,95	0,00
	3	3676	10,16	0,08	**3679**	185,47	0,00
	4	3656	10,15	0,22	**3664**	187,27	0,00
	5	3640	10,15	0,19	**3646**	189,74	0,03
	6	3594	10,15	0,06	**3596**	189,87	0,00
	7	3523	10,16	0,54	**3542**	191,49	0,00
	8	3615	10,15	0,11	**3619**	185,62	0,00
	9	3541	10,18	0,34	**3553**	183,57	0,00
	10	3645	10,14	0,19	**3650**	188,23	0,05
Mean		3588,8	10,16	0,22	**3596,2**	190,08	0,01
800	1	4171	10,22	0,31	**4179**	210,06	0,12
	2	4048	10,21	0,42	**4065**	184,38	0,00
	3	4098	10,25	0,10	**4104**	207,26	-0,05
	4	4042	10,24	0,22	**4056**	186,59	-0,12
	5	4084	10,21	0,02	**4085**	190,70	0,00
	6	4240	10,21	0,21	**4248**	184,07	0,02
	7	4116	10,22	0,12	**4125**	191,90	-0,10
	8	4054	10,22	0,22	**4063**	272,79	0,00
	9	4061	10,24	0,47	**4080**	183,37	0,00
	10	4127	10,22	-0,07	**4128**	187,37	-0,10
Mean		4104,1	10,22	0,20	**4113,3**	199,85	-0,02
1000	1	4924	10,43	0,06	**4939**	193,99	-0,24
	2	4971	10,43	-0,10	**4985**	221,42	-0,38
	3	5162	10,46	0,17	**5166**	184,61	0,10
	4	5111	10,55	0,58	**5134**	189,92	0,14
	5	5123	10,49	0,21	**5136**	187,04	-0,04
	6	5058	10,49	0,47	**5081**	186,99	0,02
	7	5092	10,51	0,16	**5112**	185,06	-0,24
	8	5176	10,46	0,04	**5182**	194,64	-0,08
	9	5079	10,44	0,57	**5106**	184,00	0,04
	10	5162	10,44	0,31	**5175**	199,07	0,06
Mean		5085,8	10,47	0,25	**5101,6**	192,67	-0,06
Total Mean		3.842,95	10,23	0,19	**3.851,58**	188,42	-0,02

Table 5.3: LK instances comparison

Instances		**ILS-VND**			**HDM-ILS-VND**		
# itens	Id	Best Solution	Time (s)	Gap (%)	Best Solution	Time (s)	Gap (%)
500	1	2718	10,08	-0,22	**2719**	256,76	-0,26
	2	**2739**	10,08	-0,37	**2739**	415,86	-0,37
	3	**2639**	10,07	0,00	**2639**	193,86	0,00
	4	**2665**	10,08	0,00	**2665**	183,92	0,00
	5	**2689**	10,07	-0,11	**2689**	315,72	-0,11
	6	2752	10,08	-0,22	**2754**	472,96	-0,29
	7	**2706**	10,08	-0,63	2701	261,51	-0,45
	8	**2681**	10,08	0,00	**2681**	185,45	0,00
	9	**2652**	10,07	0,00	**2652**	238,21	0,00
	10	**2675**	10,08	-0,38	**2675**	611,67	-0,38
Mean		**2691,6**	10,08	-0,19	2691,4	313,59	-0,19
700	1	3757	10,19	0,00	**3761**	280,93	-0,11
	2	**3611**	10,16	0,00	**3611**	678,63	0,00
	3	**3833**	10,18	-0,24	**3833**	669,09	-0,24
	4	3838	10,18	-0,08	**3840**	2416,68	-0,13
	5	**3842**	10,17	-0,50	**3842**	861,47	-0,50
	6	3718	10,17	-0,30	**3719**	633,35	-0,32
	7	**3682**	10,17	-0,16	**3682**	217,48	-0,16
	8	3790	10,19	-0,74	**3795**	203,03	-0,88
	9	**3655**	10,19	-0,11	3650	400,44	0,03
	10	**3832**	10,17	0,00	**3832**	221,69	0,00
Mean		3755,8	10,18	-0,21	**3756,5**	658,28	-0,23
800	1	**4305**	10,27	-0,16	4304	193,58	-0,14
	2	4212	10,26	-0,26	**4218**	510,32	-0,40
	3	4254	10,27	-0,07	**4257**	538,39	-0,14
	4	4210	10,28	-0,02	**4212**	230,65	-0,07
	5	4231	10,26	-1,32	**4231**	719,53	-1,32
	6	4418	10,27	-0,02	**4421**	430,70	-0,09
	7	4291	10,29	-0,16	**4293**	347,37	-0,21
	8	4155	10,29	-0,27	**4158**	447,89	-0,34
	9	4301	10,28	-0,70	**4305**	195,39	-0,80
	10	4281	10,27	-0,09	**4285**	212,23	-0,19
Mean		4265,8	10,27	-0,31	**4268,4**	382,60	-0,37
1000	1	5175	10,46	-0,54	**5176**	307,74	-0,56
	2	**5156**	10,50	0,00	5155	258,85	0,02
	3	5389	10,51	-0,92	**5390**	187,15	-0,94
	4	5424	10,50	-0,06	**5430**	196,49	-0,17
	5	**5304**	10,48	-0,95	5301	204,71	-0,89
	6	5344	10,52	0,02	**5351**	187,90	-0,11
	7	5256	10,45	-0,23	**5271**	280,44	-0,51
	8	**5383**	10,46	-1,13	5373	213,80	-0,94
	9	5334	10,48	-0,74	**5338**	217,15	-0,81
	10	5386	10,48	-0,45	**5400**	266,96	-0,71
Mean		5315,1	10,48	-0,50	**5318,5**	232,12	-0,56
Total Mean		4.007,08	10,25	-0,30	**4.008,70**	396,65	-0,34

Table 5.4: MF instances comparison

Instances		**ILS-VND**			**HDM-ILS-VND**		
# itens	Id	Best Solution	Time (s)	Gap (%)	Best Solution	Time (s)	Gap (%)
500	1	**2368**	10,08	0,00	**2368**	200,77	0,00
	2	**2318**	10,07	-0,35	2315	182,75	-0,22
	3	2283	10,08	0,04	**2284**	189,73	0,00
	4	**2269**	10,08	-0,44	2266	182,85	-0,31
	5	**2321**	10,09	0,00	**2321**	328,86	0,00
	6	**2319**	10,08	-0,13	2318	183,05	-0,09
	7	**2290**	10,08	-0,09	**2290**	211,78	-0,09
	8	2214	10,08	-0,59	**2215**	184,18	-0,64
	9	**2263**	10,08	-0,18	**2263**	190,48	-0,18
	10	**2310**	10,09	-0,22	**2310**	198,18	-0,22
Mean		**2295,5**	10,08	-0,20	2295	205,26	-0,17
700	1	3127	10,19	0,00	**3130**	190,97	-0,10
	2	**3052**	10,19	-0,46	**3052**	185,60	-0,46
	3	3222	10,18	-0,78	**3225**	185,74	-0,88
	4	3241	10,21	-0,25	**3247**	192,25	-0,43
	5	3243	10,18	-0,15	**3246**	201,50	-0,25
	6	3132	10,19	-0,10	**3133**	184,43	-0,13
	7	3049	10,20	-1,13	**3052**	189,03	-1,23
	8	3174	10,21	-0,25	**3175**	184,56	-0,28
	9	**3216**	10,21	-0,94	3210	186,72	-0,75
	10	3210	10,19	-0,22	**3212**	244,03	-0,28
Mean		3166,6	10,20	-0,43	**3168,2**	194,48	-0,48
800	1	3700	10,26	-0,24	**3701**	189,49	-0,27
	2	**3726**	10,31	-0,40	**3726**	185,70	-0,40
	3	**3668**	10,26	-1,75	3663	311,47	-1,61
	4	3519	10,28	-0,83	**3530**	184,87	-1,15
	5	3751	10,29	-0,27	**3753**	184,16	-0,32
	6	**3775**	10,31	-0,08	3774	262,70	-0,05
	7	**3681**	10,26	0,05	3678	185,92	0,14
	8	**3612**	10,27	-1,03	**3612**	185,17	-1,03
	9	3594	10,27	-0,03	**3599**	191,69	-0,17
	10	3635	10,27	-0,06	**3650**	186,90	-0,47
Mean		3666,1	10,28	-0,46	**3668,6**	206,80	-0,53
1000	1	4462	10,57	-0,27	**4464**	185,17	-0,31
	2	**4469**	10,55	-1,38	4458	185,59	-1,13
	3	4624	10,58	-1,03	**4625**	187,00	-1,05
	4	**4590**	10,53	-0,57	4583	187,97	-0,42
	5	**4530**	10,57	-1,39	4521	185,98	-1,19
	6	**4591**	10,63	-0,48	4590	185,40	-0,46
	7	4611	10,60	-0,72	**4614**	224,70	-0,79
	8	**4572**	10,57	-1,92	4566	186,69	-1,78
	9	4640	10,55	-0,19	**4641**	187,89	-0,22
	10	4661	10,61	-0,02	**4663**	186,41	-0,06
Mean		**4575**	10,58	-0,80	4572,5	190,28	-0,74
Total Mean		3.425,80	10,28	-0,47	**3.426,08**	199,21	-0,48

Experiments on "MF" instances

From Table 5.4, we again see the same overall pattern that HDM-ILS-VND can improve on some instances compared to ILS-VND, though the improvement is typically smaller than with the O-type instances, our proposed methodology was superior in half of the instances. In this case, HDM-ILS-VND achieves a slightly better best solution (or gap) than ILS-VND. These improvements are not as widespread as in the O-type dataset but show that HDM's additional mechanisms can still pay off on certain MF instances. For eight instances, both methods find identical solution costs. Where they do tie, ILS-VND typically requires a fraction of the CPU time that HDM-ILS-VND expends.

Overall, on these MF instances, HDM-ILS-VND still can outperform ILS-VND, but due to the complexity of the instance, the gains appear smaller and come at a greater computational cost.

Experiments Conclusions

The experimental results demonstrate that HDM-ILS-VND offers competitive, and often improved, solution quality compared to the ILS-VND approach. On the O instances, HDM-ILS-VND not only matched but improved the ILS-VND outcomes in a significant number of cases, with 36 instances showing clear benefits. Even when the method did not produce superior results, it maintained solution quality, highlighting its robustness. This performance is attributed to the additional data mining phase, which, despite increasing computational times, systematically contributes to narrowing the solution gap relative to the best-known CPLEX solutions.

On the more complex LK and MF instances, the advantages of HDM-ILS-VND become more nuanced. While the method showed improvements in several LK instances and achieved better gap values in the MF instances, these gains often came with longer CPU times compared to ILS-VND. Overall, the experiments indicate that the choice between HDM-ILS-VND and ILS-VND will depend on the specific application requirements, where the trade-off between solution quality and computational efficiency must be balanced.

5.3.2 Statistical Analysis

A hypothesis test was conducted to determine whether the differences in best solution values between ILS-VND and HDM-ILS-VND were statistically significant. Since the data did not follow a normal distribution and the measurements from the experiments were paired, the non-parametric Wilcoxon Signed Rank Test was employed. This test enables the rejection of the null hypothesis at a significance level denoted as θ, with a corresponding probability of $((1-\theta)\times 100\%)$. This study used

a significance level of $\theta = 0.05$, the same methodology developed by SOUTO *et al.* (2024a).

The formulated hypotheses are as follows:

- Null Hypothesis (H0): The median difference is zero.
- Alternative Hypothesis (H1): The median difference is not zero.

The null hypothesis is that there is no difference in best solution values with HDM-ILS-VND versus ILS-VND. If the null hypothesis is true, we expect to see some positive differences (improvement) and some negative differences (worsening). Since p-value $< \theta$, the null hypothesis has been rejected, and there are more positive than negative differences, the hypothesis that HDM-ILS-VND outperforms the ILS-VND is true.

The statistical test underscores a clear advantage of HDM-ILS-VND over the standard ILS-VND regarding solution quality. The Wilcoxon Signed Rank Test ($p_{value} < \theta = 0.05$) confirms this improvement is highly significant, not due to random chance.

This result has important practical implications. The HDM-ILS-VND's modifications yield better outcomes for the problem instances tested. We can expect superior solutions using HDM-ILS-VND, as evidenced by the much lower median and tighter distribution of best solutions. In summary, the analysis provides strong evidence that HDM-ILS-VND outperforms ILS-VND in finding the best solutions, with the improvement being statistically significant. This suggests that the Data Mining phase enhancements to ILS-VND improve the search effectiveness, leading to better optimal or near-optimal solutions across the benchmark instances.

5.4 HDM-ILS-VND versus literature

This section will compare our proposed method with the best-known approaches documented in the literature review from Chapter 2. The methods we will focus on include MFSS and RL-DCSS. We have not included HESM in our comparative analysis, as it reported a better heuristic value than the established optimum. Additionally, ILS-VND is excluded since its results were already discussed in the previous section.

After executing the algorithm ten times for each on 120 benchmark instances, Table 5.5 shows that the mean solution cost, time, and gap of HDM-ILS-VND are highly competitive in solving the Knapsack Problem with Forfeits (KPF), consistently achieving high-quality solutions across all instance types and sizes. Compared

Table 5.5: HDM-ILS-VND versus Literature

Instances		**HDM-ILS-VND**			**MFSS**			**RCL-DCSS**		
Type	*# items*	Solution	Time(s)	Gap	Solution	Time(s)	Gap	Solution	Time(s)	Gap
O	500	**2595,20**	171,09	0,00	**2595,20**	103,77	0,00	**2595,20**	-	0,00
	700	**3596,20**	190,08	0,01	**3596,50**	141,27	0,00	**3596,50**	-	0,00
	800	4113,30	199,85	-0,02	4114,90	164,85	-0,06	**4115,10**	-	-0,07
	1000	5101,60	192,67	-0,06	**5108,30**	204,22	-0,19	**5108,30**	-	-0,19
LK	500	2691,40	313,59	-0,18	**2691,90**	93,46	-0,20	**2691,90**	-	-0,20
	700	3756,50	658,28	-0,23	3757,60	160,41	-0,26	**3757,90**	-	-0,27
	800	4268,40	382,60	-0,37	4270,20	162,47	-0,41	**4271,60**	-	-0,44
	1000	5318,50	232,12	-0,56	5322,20	182,06	-0,63	**5325,30**	-	-0,69
MF	500	2295,00	205,26	-0,17	2295,80	217,81	-0,21	**2296,40**	-	-0,23
	700	3168,20	194,48	-0,48	3168,50	270,50	-0,49	**3169,70**	-	-0,53
	800	3668,60	206,80	-0,53	3670,30	273,74	-0,58	**3670,70**	-	-0,59
	1000	4572,50	190,28	-0,74	4580,00	309,00	-0,91	**4580,90**	-	-0,93

with MFSS and RCL-DCSS, we find gap percentages consistently close to the best-known solutions in the literature, reaffirming its strength as a high-performance metaheuristic for the KPF and an alternative solution in more practical scenarios. Another key advantage of HDM-ILS-VND is its robust performance across different instance types. While ILS-VND achieves competitive solutions in some cases, its simpler local search strategy limits its ability to find optimal or near-optimal solutions in complex scenarios.

For O instances, our methodology found all optimal solutions certified by CPLEX within 3 hours. For LK instances, HDM-ILS-VND found four values superior to MFSS and five better solutions for MF instances. Still, our method found three new best-known values across all the literature in the most challenging scenarios for 800 and 1000 items for MF instances.

Based on the values in Table 5.5, HDM-ILS-VND often leads or closely matches the best solution costs among the three compared methods. For instance, on several 500 and 700-item cases, HDM-ILS-VND matches the best-known solution cost exactly (or within 0–1 cost units in the difference of solution cost). Even for the more demanding 800- and 1000-item instances, its final cost is typically within 1–7 units of MFSS or RCL-DCSS.

When HDM-ILS-VND does happen to lose by a few cost units, the gap is usually quite small. Meanwhile, its runtime gains can be sizable and large, making HDM-ILS-VND a balanced option. In many tests, it surpasses MFSS or RCL-DCSS by a margin of a few cost units (1–3 on average) at faster speeds, reinforcing that HDM-ILS-VND consistently provides near-best or best solutions without incurring a large performance in the rare cases it doesn't lead in objective value.

To visually compare HDM-ILS-VND with three other works in the literature, we present a bar chart in Figure 5.1 highlighting the differences in solution quality. Additionally, Figure 5.2 compares computational time. Both comparisons uses the mean values for each instance type and number of items, represented on the x-axis.

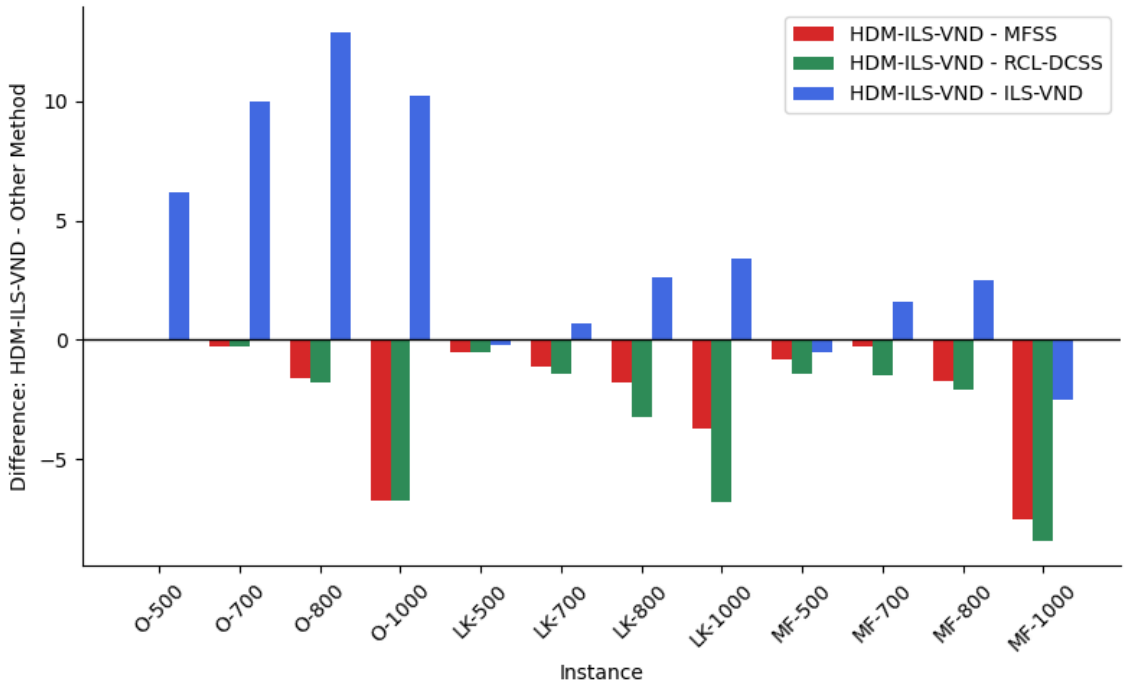

Figure 5.1: Solution quality comparison

The y-axis shows the difference in mean solution value between HDM-ILS-VND and the other methods in the literature. The time comparison follows a similar approach, using the solution mean time in the y-axis for each instance type.

Figure 5.1 shows that HDM-ILS-VND performs similarly to the other methods, with most bars close to zero. This proximity underscores that the methods are competitive in terms of solution quality. Whenever a bar is above zero, HDM-ILS-VND achieves a better solution quality; when it is below zero, it shows a worse relative solution from HDM-ILS-VND to the other method. The fact that these differences rarely stretch far from zero highlights that no single method dominates across all instances. However, in certain instances, slight positive or negative differences indicate that HDM-ILS-VND either outperforms or is outperformed by a small margin, suggesting there may be particular instance types where it could further improve. This figure reveals that HDM-ILS-VND is competitive, regularly matching or beating the other methods, with only minor differences in specific scenarios.

Figure 5.2 illustrates the runtime comparison of HDM-ILS-VND with MFSS and ILS-VND across each instance. We could not include RCL-DCSS in this comparison, as the authors did not report its runtime. A positive bar indicates that HDM-ILS-VND took longer than the other method, while a negative bar means it was faster. Notably, HDM-ILS-VND often requires significantly more time than ILS-VND, which consistently has runtimes of around 10 seconds. In contrast, when compared with MFSS, there are instances where the bar drops below zero, suggesting

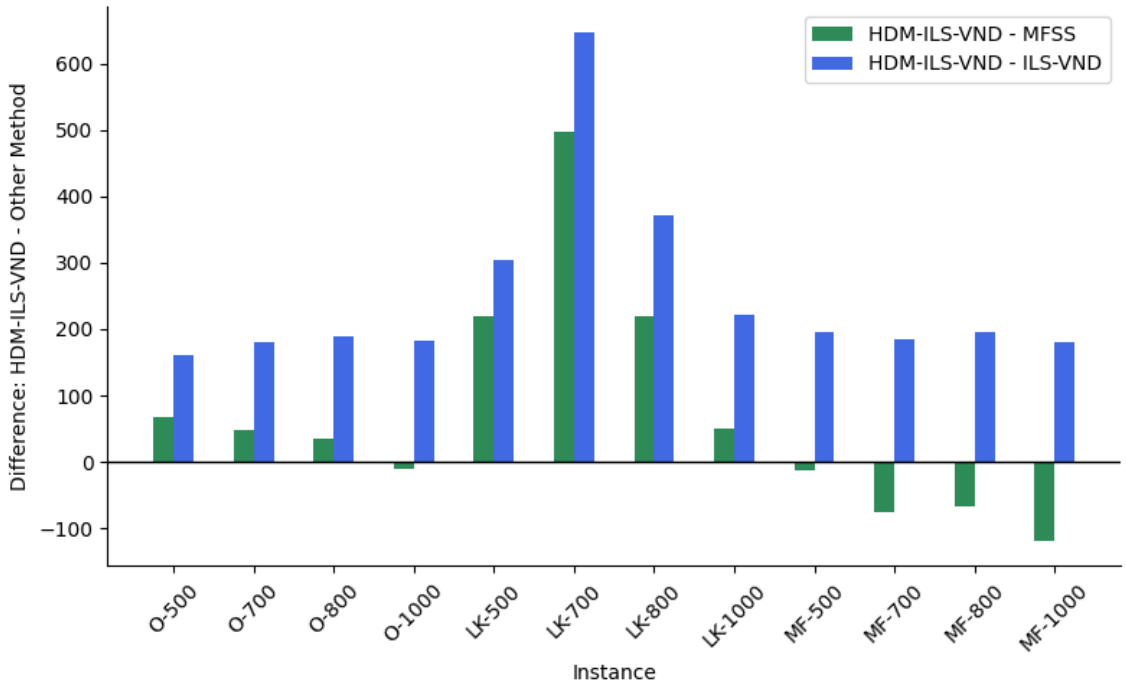

Figure 5.2: Time comparison

that HDM-ILS-VND finishes more quickly. However, the bars show strong positive values for several LK instances, indicating that HDM-ILS-VND is slower.

In summary, the comparative analysis shows that HDM-ILS-VND is a strong and competitive metaheuristic for the KPF. It produces high-quality solutions across various instance types while balancing solution quality and computational time. Although MFSS and RCL-DCSS also demonstrate competitive performance, the hybridization of data mining within HDM-ILS-VND allows it to achieve good objective values, especially for type O instances, and even establish new best-known upper bounds in many complex cases. This combination of effectiveness and efficiency positions HDM-ILS-VND as a promising alternative to traditional methods like ILS-VND, which, although faster, does not match the solution quality provided by HDM-ILS-VND.

Chapter 6

Conclusions

In this master thesis, we focus on the Knapsack Problem with Forfeits (KPF), a challenging extension of the classical 0-1 Knapsack Problem that incorporates penalty costs when conflicting items are selected. To address this complexity, we introduce a novel Hybrid Data Mining ILS-VND (HDM-ILS-VND) approach. This method enhances the standard Iterated Local Search with Variable Neighborhood Descent (ILS-VND) by integrating data mining techniques that extract frequent item set patterns from an elite pool of solutions. With these discovered patterns, a mixed-integer programming model is built to search for better solutions in the subspace defined by the combination of the mined patterns. In the final phase, a local branching strategy is applied to refine the best solution by exploring its immediate neighborhood, thereby further improving the quality of the best solution found.

Computational experiments on established benchmark instances demonstrated that HDM-ILS-VND consistently obtains better or comparable solutions than ILS-VND, often reaching new best-known results in reasonable execution times. The experiments also revealed that while data mining and local branching add complexity and computational overhead, they yield an apparent gain in solution quality. The hypothesis test confirms that these enhancements are not due to randomness. Comparisons with recent, high-performing algorithms from the literature showed that HDM-ILS-VND achieves top-tier solutions for moderate and large instances, especially in scenarios with heavier penalty structures and high capacity constraints.

The research results underscore the effectiveness of combining heuristic search with data mining for KPF. By mining frequent patterns from an elite set of solutions, we efficiently steer the local search away from unproductive regions. The selective local branching phase intensifies the search for promising solutions.

Future research could explore enhancements that could involve exploring different forms of learning within the ILS framework beyond frequent itemset mining. For instance, machine learning models could predict which items or item pairs are

most promising to include or which penalties are most detrimental, then guide the perturbation and local search processes more dynamically. Integrating advanced machine learning techniques like reinforcement learning or neural networks might reduce parameter tuning and adaptively calibrate the search based on instance features.

References

DARMANN, A., PFERSCHY, U., SCHAUER, J. "Minimal spanning trees with conflict graphs", *Optimization online*, 2009.

CARRABS, F., CERULLI, R., PENTANGELO, R., et al. "Minimum spanning tree with conflicting edge pairs: a branch-and-cut approach", *Annals of Operations Research*, v. 298, n. 1-2, pp. 65–78, 2021.

GALBIATI, G. "Approximating minimum cut with bounded size". In: *International Conference on Network Optimization*, pp. 210–215. Springer, 2011.

YAMADA, T., KATAOKA, S., WATANABE, K. "Heuristic and exact algorithms for the disjunctively constrained knapsack problem", *Information Processing Society of Japan Journal*, v. 43, n. 9, 2002.

HIFI, M., OTMANI, N. "An algorithm for the disjunctively constrained knapsack problem", *International Journal of Operational Research*, v. 13, n. 1, pp. 22–43, 2012.

BETTINELLI, A., CACCHIANI, V., MALAGUTI, E. "A branch-and-bound algorithm for the knapsack problem with conflict graph", *INFORMS Journal on Computing*, v. 29, n. 3, pp. 457–473, 2017.

BASNET, C. "Heuristics for the multiple knapsack problem with conflicts", *International Journal of Operational Research*, v. 32, n. 4, pp. 514–525, 2018.

FÜGENSCHUH, A., BAMBACH, M., BUHL, J. "Trajectory optimization for wire-arc additive manufacturing". In: *Operations Research Proceedings 2018: Selected Papers of the Annual International Conference of the German Operations Research Society (GOR), Brussels, Belgium, September 12-14, 2018*, pp. 331–337. Springer, 2019.

EPSTEIN, L., LEVIN, A. "On bin packing with conflicts", *SIAM Journal on Optimization*, v. 19, n. 3, pp. 1270–1298, 2008.

MURITIBA, A. E. F., IORI, M., MALAGUTI, E., et al. "Algorithms for the bin packing problem with conflicts", *Informs Journal on computing*, v. 22, n. 3, pp. 401–415, 2010.

CONIGLIO, S., FURINI, F., SAN SEGUNDO, P. "A new combinatorial branch-and-bound algorithm for the knapsack problem with conflicts", *European Journal of Operational Research*, v. 289, n. 2, pp. 435–455, 2021.

BEN SALEM, M., TAKTAK, R., MAHJOUB, A. R., et al. "Optimization algorithms for the disjunctively constrained knapsack problem", *Soft Computing*, v. 22, pp. 2025–2043, 2018.

CAPOBIANCO, G., D'AMBROSIO, C., PAVONE, L., et al. "A hybrid metaheuristic for the Knapsack Problem with Forfeits", *Soft Computing*, pp. 1–14, 2022.

MOURA, A. F. C., NORONHA, T., BOGUE, E. T. "Uma heurística ILS para o Problema da Mochila com Penalidades", *Simpósio Brasileiro de Pesquisa Operacional (SBPO)*, 2021.

SOUTO, G., ARITSUGI, M., MENDONÇA, I., et al. "A Q-learning hybrid BRKGA applied to the knapsack problem with forfeits". In: *2024 IEEE Congress on Evolutionary Computation (CEC)*, pp. 1–8. IEEE, 2024a.

SOUTO, G., SIMONETTI, L., GONZALEZ, P. "Estudando Variacoes do Generalized Simulated Annealing no Problema da Mochila com Penalidades". In: *Simpósio Brasileiro de Pesquisa Operacional (SBPO)*, 2024b.

SOUTO, G., MICELI, C., SIMONETTI, L., et al. "Machine Learning Assisted Hybrid Genetic Algorithm Applied to the Knapsack Problem with Forfeits". In: *International Conference on Machine Learning (ICML)*, 2024c.

CERULLI, R., D'AMBROSIO, C., RAICONI, A., et al. "The knapsack problem with forfeits". v. 12176 LNCS, p. 263 – 272, 2020. doi: 10.1007/978-3-030-53262-8_22. Disponível em: <https://www.scopus.com/inward/record.uri?eid=2-s2.0-85089209375&doi=10.1007%2f978-3-030-53262-8_22&partnerID=40&md5=eea62b2cd6af73e9ca9f1340aba0e2e0>. Cited by: 10.

VIEIRA, M. M., NOGUEIRA, B., PINHEIRO, R. G. "An integrated ILS-VND strategy for solving the knapsack problem with forfeits", *Journal of Heuristics*, v. 30, n. 5, pp. 399–420, 2024.

MINING, W. I. D. “Data mining: Concepts and techniques”, *Morgan Kaufinann*, v. 10, n. 559-569, pp. 4, 2006.

FISCHETTI, M., LODI, A. “Local branching”, *Mathematical Programming*, v. 98, pp. 23–47, 2003.

PAGE, M. J., MCKENZIE, J. E., BOSSUYT, P. M., et al. “Updating guidance for reporting systematic reviews: development of the PRISMA 2020 statement”, *Journal of clinical epidemiology*, v. 134, pp. 103–112, 2021.

JOVANOVIC, R., VOSS, S. “Fixed set search matheuristic applied to the knapsack problem with forfeits”, *Computers & Operations Research*, v. 168, pp. 106685, 2024.

ZHAO, J., HIFI, M. “A reinforcement learning-driven cooperative scatter search for the knapsack problem with forfeits”, *Computers & Industrial Engineering*, v. 198, pp. 110713, 2024.

ZHOU, Q., HAO, J.-K., JIANG, Z.-Z., et al. “Adaptive feasible and infeasible evolutionary search for the knapsack problem with forfeits”, *International Transactions in Operational Research*, v. 32, n. 3, pp. 1442–1471, 2025.

CERRONE, C., CERULLI, R., GOLDEN, B. “Carousel greedy: A generalized greedy algorithm with applications in optimization”, *Computers & Operations Research*, v. 85, pp. 97–112, 2017.

JOVANOVIC, R., TUBA, M., VOSS, S. “Fixed set search applied to the traveling salesman problem”. In: *Hybrid Metaheuristics: 11th International Workshop, HM 2019, Concepción, Chile, January 16–18, 2019, Proceedings 11*, pp. 63–77. Springer, 2019.

HIFI, M., MICHRAFY, M. “A reactive local search-based algorithm for the disjunctively constrained knapsack problem”, *Journal of the Operational Research Society*, v. 57, n. 6, pp. 718–726, 2006.

HIFI, M., MICHRAFY, M. “Reduction strategies and exact algorithms for the disjunctively constrained knapsack problem”, *Computers & operations research*, v. 34, n. 9, pp. 2657–2673, 2007.

PFERSCHY, U., SCHAUER, J. “The knapsack problem with conflict graphs”, *Journal of Graph Algorithms and Applications*, v. 13, n. 2, pp. 233–249, 2009.

DUARTE, A., MLADENOVIC, N., SÁNCHEZ-ORO, J., et al. “Variable neighborhood descent”, *Handbook of heuristics*, pp. 341–367, 2018.

WITTEN, I. H., FRANK, E., HALL, M. A., et al. "Practical machine learning tools and techniques". In: *Data mining*, v. 2, pp. 403–413. Elsevier Amsterdam, The Netherlands, 2005.

LIU, Y.-H. "Mining frequent patterns from univariate uncertain data", *Data & Knowledge Engineering*, v. 71, n. 1, pp. 47–68, 2012.

HAN, J., PEI, J., TONG, H. *Data mining: concepts and techniques*. Morgan kaufmann, 2022.

BARBALHO, H., ROSSETI, I., MARTINS, S. L., et al. "A hybrid data mining GRASP with path-relinking", *Computers & Operations Research*, v. 40, n. 12, pp. 3159–3173, 2013.

PLASTINO, A., BARBALHO, H., SANTOS, L. F. M., et al. "Adaptive and multi-mining versions of the DM-GRASP hybrid metaheuristic", *Journal of Heuristics*, v. 20, pp. 39–74, 2014.

GUERINE, M., ROSSETI, I., PLASTINO, A. "Extending the hybridization of metaheuristics with data mining: Dealing with sequences", *Intelligent Data Analysis*, v. 20, n. 5, pp. 1133–1156, 2016.

RIBEIRO, M. H., PLASTINO, A., MARTINS, S. L. "Hybridization of GRASP metaheuristic with data mining techniques", *Journal of Mathematical Modelling and Algorithms*, v. 5, n. 1, pp. 23–41, 2006.

GOETHALS, B., ZAKI, M. J. "Advances in frequent itemset mining implementations: introduction to FIMI'03". In: *Proceedings of the ICDM 2003 Workshop on Frequent Itemset Mining Implementations (FIMI'03)/Goethals, Bart [edit.]; et al.[edit.]*, 2003.

GONZALEZ, P. H., SIMONETTI, L., MICHELON, P., et al. "A variable fixing heuristic with Local Branching for the fixed charge uncapacitated network design problem with user-optimal flow", *Computers & Operations Research*, v. 76, pp. 134–146, 2016.

GONZALEZ, P. H., BRANDAO, J. "A biased random key genetic algorithm to solve the transmission expansion planning problem with re-design". In: *2018 IEEE Congress on Evolutionary Computation (CEC)*, pp. 1–7. IEEE, 2018.