Alma Mater Studiorum - Università di Bologna

DOTTORATO DI RICERCA IN

INGEGNERIA BIOMEDICA, ELETTRICA E DEI SISTEMI

Ciclo 34

Settore Concorsuale: 01/A6 - RICERCA OPERATIVA

Settore Scientifico Disciplinare: MAT/09 - RICERCA OPERATIVA

FORMULATIONS AND METAHEURISTICS FOR COMBINATORIAL OPTIMIZATION PROBLEMS

Presentata da: Carlos Rodrigo Rey Barra

Coordinatore Dottorato	**Supervisore**
Michele Monaci	Daniele Vigo

Esame finale anno 2022

Alma Mater Studiorum-Università di Bologna

DOTTORATO DI RICERCA IN

Ingegneria Biomedica, Elettrica e dei Sistemi
(curriculum Ricerca Operativa)

Ciclo XXXIV

Settore Scientifico Disciplinare: MAT/09 - R. OPERATIVA
Settore Concorsuale: 01/A6 - R. OPERATIVA

Formulations and Metaheuristics for combinatorial optimization problems

Presentata da: Carlos Rodrigo Rey Barra

Coordinatore Dottorato
Michele Monaci

Supervisori
Prof. Daniele Vigo
Prof. Paolo Toth

Esame finale anno 2022

ALMA MATER STUDIORUM-UNIVERSITÀ DI BOLOGNA

Abstract

Formulations and Metaheuristics for combinatorial optimization problems

by Carlos REY

Combinatorial optimization problems have been strongly addressed throughout history. Their study involves highly applied problems that must be solved in reasonable times. This doctoral Thesis addresses three Operations Research problems: the first deals with the Traveling Salesman Problem with Pickups and Delivery with Handling cost, which was approached with two metaheuristics based on Iterated Local Search; the results show that the proposed methods are faster and obtain good results respect to the metaheuristics from the literature. The second problem corresponds to the Quadratic Multiple Knapsack Problem, and polynomial formulations and relaxations are presented for new instances of the problem; in addition, a metaheuristic and a matheuristic are proposed that are competitive with state of the art algorithms. Finally, an Open-Pit Mining problem is approached. This problem is solved with a parallel genetic algorithm that allows excavations using truncated cones. Each of these problems was computationally tested with difficult instances from the literature, obtaining good quality results in reasonable computational times, and making significant contributions to the state of the art techniques of Operations Research.

Acknowledgements

First, I want to thank my supervisor professors Paolo Toth and Daniele Vigo, for their advice and help in this doctoral process. I will be eternally grateful for the opportunity they have given me to study at the Universita di Bologna.

Also, thank the professors who helped me in my Ph.D. process: Laura Galli, Silvano Martello, Victor Parada, and Nelson Morales. Thank you for having considered me for these projects worked on.

I also want to thank many friends and colleagues who supported me during my preparation as a Ph.D.: Luca Accorsi, Carlos Contreras, Nicolas Campana, Federico Naldini, Alan Osorio, Paolo Paronuzzi and Henri Bertrand Roger Jean- Marc Arthur Lefebvre. Thank you very much for always having a word of support and advice at all times. Also, thank all my doctoral colleagues that I met in the laboratory.

Also, thank Daniela and Gabriele for helping me throughout the process, especially during my quarantines. Without their help, I would not have been able to visit my family in Chile, and I would not have been able to return to continue my studies.

Thank friends in Chile, Wences, Pipe, Marcos, Diego y Gera. They always had a word of encouragement at different times of my trip. Thanks !!!

I also want to thank the church choir I participate in Chile. Thanks to all the members, especially in the pandemic when we have come together the most.

Thank my girlfriend, Elizabeth. Thank you for always being by my side at all times. Thank you for having the assertive phrase in times of chaos. Thank you for your patience and your love. I love you bototo.

Finally, thank my family: my father Julián, my mother Albertina, brother Julián and sister Carmen. Thank you for giving me unconditional support during these 3 years of study. It was a very difficult time due to the pandemic, so I appreciate your patience and understanding at all times.

A formal thanks to my scholarship: National Agency for Research and Development (ANID) / Scholarship Program / Doctorado Becas Chile / 2018-72190600.

Contents

List of Figures

List of Tables

-

The mystery of life is certainly the most persistent problem ever placed before the thought of man..... The inability of science to solve it is absolute. This would be truly frightening were it not for faith.

-

Guglielmo Marconi (10 Sep 1934 to the International Congress of Electro-Radio Biology, Venice).

Chapter 1

Introduction

Modern Operations Research originates from the Second World War. War scenarios such as aircraft redeployment, and camouflage were tasks deployed by essential Operations Research personnel, such as Great Britain's Army Operations Research Group (AORG), for Normandy's Battle [1]. Once the war ended, these personnel adapted the knowledge obtained during that period for civil needs and later industrial markets. Since then, Operations Research continues to grow in applications, such as allocation, network optimization, routing, and planning processes [2]. Operations Research is defined as a discipline that uses methods supported in mathematical sciences (such as mathematical modeling or statistical analysis) that help a decision-maker [3].

An important branch that is related to Operations Research is combinatorial optimization. Combinatorial optimization could be defined as a field of research that seeks a maximum (or minimum) of an objective function within a discrete space [4]. This branch has contributed to crucial research areas such as artificial intelligence, software engineering, and computer science. The complexity theory classifies many combinatorial optimization problems due to their difficulty. This theory formalizes the classification and helps to quantify the number of resources necessary to solve them, such as time and memory [5].

There are problems of combinatorial optimization that have been widely studied throughout history. The Traveling Salesman Problem (TSP), for example, consists of minimizing the distance of a route, where it is necessary to visit a set of places (for example, cities, commercial premises, among others) only once and later return to the point of origin. This problem is classified by complexity theory as NP-Hard [6]. It has various applications in industry such as drilling of printed circuit boards [7], overhauling gas turbine engines [8], X-Ray crystallography [9], and Vehicle Routing [10]. A specific case of the TSP that has been widely studied in the literature is the TSP with time windows or TSPTW; this problem consists of solving the TSP problem such that each customer has a window time in which it can be visited, therefore, the time availability of each customer must be taken into account. Another significant and widely studied problem is the Knapsack Problem (KP). Given a finite set of items, each with a weight and a profit, it is necessary to select a subset of these items so that the global weight does not exceed a specific capacity and the global profit is maximized. This problem is classified by complexity theory as NP-Complete [11]. It has various applications in the industry such as selection of investments and portfolios [12] or selection of assets for asset-backed securitization [13].

Solving these types of problems can impact a company's resource allocation. For example, a copper deposit study problem (related to knapsack constraints mentioned above) is necessary to get the site's geological and geographic properties to plan future extraction and obtain future profits. Chile is one of the leading countries in copper exports, and the demand increases every year; by 2020, the demand increased by 5.82 million tons of copper, equivalent to a year-on-year increase of 0.6%, and by 2021, it is estimated that it will increase by 5.99 million tons [14]. Maximizing copper extraction during the mining process is vital for deposits of this type, which must respond to a set of resource constraints over a time horizon. One problem that has been applied to real-world systems is the TSPTW; a recent extension is the on-demand delivery service problem [15], which considers the mobility of a customer to a delivery point, for example, Amazon Locker in the United States [16] and UNIQL-7eleven in Japan. By solving these types of problems, the company saves delivery times to customers without losing quality in the service time, allowing delivery companies to save money and generate profits at the delivery points. The examples mentioned must be solved with approaches capable of responding to the decision-maker's timing, vital for Operations Research.

Optimization problems can be solved using different approaches. These are classified into two large sets: exact and heuristic. Exact methods are characterized by finding optimal solutions, and generally solve small or medium size problems since their computing time and required computational resource are large. Examples of these approaches are Dynamic Programming, Branching Approaches (Branch and Bound, Branch and Cut, and Branch and Prize), and Constrained Programming. On the other hand, heuristics are search procedures that do not guarantee optimal solutions but ,for large problems, find solutions faster than an exact approach. In 1962 a different line of research on solution methods appeared that used the evolution of species [17] as an analogy to generate an algorithmic framework. This idea gave way to a set of techniques known as metaheuristics.

A metaheuristic is a high-level process that helps to generate a heuristic that provides a good quality solution in reasonable computational times independently of the problem [18]. This set of techniques has proven to be a good approach throughout the history of Operations Research. Figure 1.1 shows a graph with some metaheuristics mentioned in this thesis organized according to their year of definition (inspired by [19]). The arrows indicate the relationship between the metaheuristics. The fundamental techniques are Local Search and Greedy heuristics. In 1962 the first essential evolutionary computing techniques appeared: Evolutionary Programming [20] and Genetic Algorithms [21]. Various techniques based on Local Search (LS) were defined during the 1980s and 1990s, such as Simulated Annealing [22], Tabu Search [23], GRASP [24], Memetic Algorithm [25], Iterated Local Search ([26] but formally in [27]) and Variable Neighborhood Search [28]. During the 90s, techniques such as Ant Colony Optimization [29], Particle Swarm Optimization [30] and Genetic Programming [31] were also defined. Metaheuristics based on some of those mentioned above have been defined in the last 25 years, such as Granular Tabu Search [32] or Multi-Start Iterated Local Search [33].

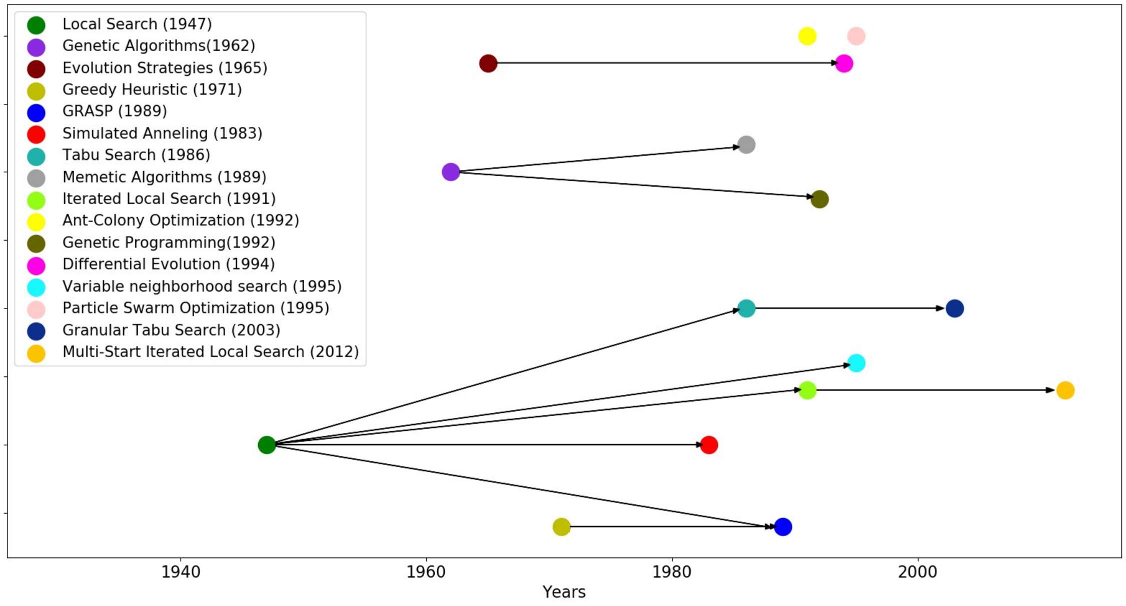


FIGURE 1.1: Timeline of some metaheuristics

Modeling a mathematical problem to obtain optimal solutions, or designing a metaheuristic to find feasible solutions in reasonable times, remain challenging activities. Operations Research makes discoveries year by year to answer real industry problems or to improve the techniques mentioned above. In this way, this thesis presents solutions for various Operations Research problems defined in recent years.

1.1 Main problems studied

1.1.1 The Traveling Salesman Problem

The Traveling Salesman Problem (TSP) has a long history in Operations Research. There are two classifications for the TSP according to the edges: asymmetric TSP or ATSP where the graph has edges with direction (i.e., the distance between two customers depends on the source customer and the destination customer); and symmetric TSP or STSP where the graph does not have edges direction (i.e., the distance between two customers does not depend on the origin customer and the destination customer since both are equivalent). The most important foundations for the ATSP were proposed by George Dantzig, Delbert Ray Fulkerson and Selmer M. Johnson between the 1950s and 1960s, with exact approaches based on integer linear programming models and the addition of valid inequalities (cutting plane methods). Subsequently, other techniques for the solution of the symmetric STSP were designed based on the Minimum Spanning Tree, approximate algorithms [34], and new branch and bound and branch and cut approaches, solving instances of almost

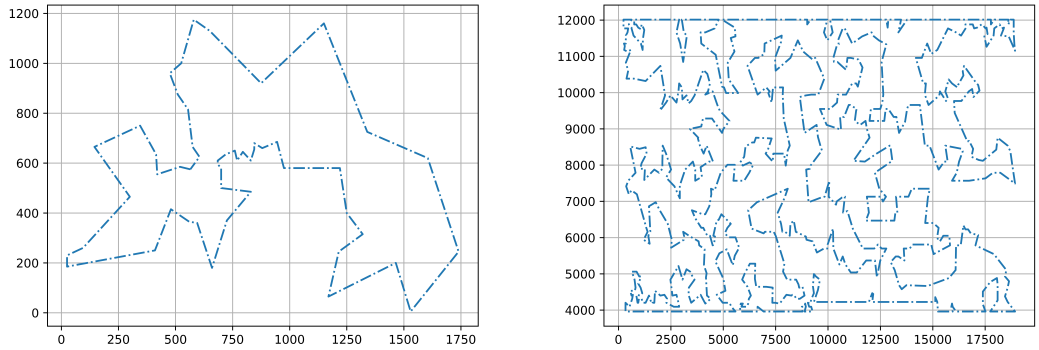


FIGURE 1.2: *For berlin52 (to the left), the heuristic took 0.01 sec with an average percentage optimality gap of 0.00%, while for vm1084 (on the right) it took 314.01 sec with an average percentage optimality gap of 0.0185%. The number of nodes and the geometry have a significant impact on the computing times. Also, consider that the Lin-Kernighan heuristic is not an exact approach, so algorithms like Branch and Bound takes larger computing times.*

2400 cities. During the 1990 the Concorde program was developed, famous for obtaining optimal solutions for different large size instances. Besides, [35] publishes the most important library of the TSP: TSPlib. This library of instances is used to test the effectiveness of TSP algorithms and is widely recognized by the operations research scientific community. Even this group of instances is used to test the effectiveness of the algorithms in other related problems such as the generalized TSP (GTSP). In the year 2000 an effective implementation of the Lin-Kernighan heuristic [36] was designed to solve large problems [37].

Formally, in the STSP, we are given $G = (V, E)$ a complete undirected graph, where V is a set of nodes (customers for the next explanation) such that $V = \{1...n\}$ and E is the set of edges such that $E = \{(i,j) : i,j \in V, i \neq j\}$: Each edge $(i,j) \in E$ has a positive and symmetric cost $c_{ij} : c_{ij} = c_{ji}$. The problem consists of generating a Hamiltonian cycle, i.e. a cycle such that all nodes are considered only once.

The execution time to solve the TSP could depend on the number of cities and their geometry. An interesting exercise to demonstrate the completeness of TSP is randomly selecting two instances from TSPlib as *berlin52* (based on the city of Berlin with 52 cities) and *vm1084* (generated by Reinelt with 1084 cities). Figure 1.2 shows the image of two solutions to classic STSP instances: $berlin52$ and $vm1084$ using the Lin-Kerninhan heurisitic with the same parameters for both executions [38].

Different variants for the problem have been defined. However, the variants considered in this thesis will focus on the definition of pickup and delivery. Pickup and Delivery problems (PDPs) are an important class of routing problem, where a finite amount of commodities (called demand) is associated with each customer. The customer's demand can be classified in the pickup, where the vehicle must load the customer's demand, and delivery, where the vehicle must unload the customer's demand. Finally, a transport vehicle with a previously defined capacity, is associated with the route. The restrictions for this type of problem are:

- All pickup and delivery requests must be satisfied.
- No transshipments of commodities are made at customers.
- The vehicle must not exceed its capacity.
- The initial node, called depot, has a pickup unit $d = 0$.

The classification of the PDPs was defined in-depth in [39], and extended for the VRP in [40]. The categories are based on three main aspects: structure, visits, and vehicles. For the structure, the origin and destination of the commodities are defined:

- Many to many (MM): Any customer can serve as an origin or destination for any commodity.
- One-to-many-to-one problems (1-M-1): the commodities are initially available at the depot and are transported to the customers. In addition, commodities available at the customers are transported to the depot.
- One-to-one (1-1) problem, each commodity has specific origin and destination.

On the other hand, visits can be classified according to the way the pickup and delivery process is carried out:

- PD indicates that each customer must be visited (mandatory) only once for pickup and delivery combinations.
- P-D indicates that each costumer can be visited more than once to satisfy the pickup and delivery combination.
- P/D, when only one type of commodity must be satisfied (i.e., either pickup or delivery for each customer, not both).

Finally, vehicles relate to the number of vehicles used in operation. For the TSP, this last classification does not apply since only one vehicle must be used. Consequently, the P-D classification cannot be applied to a Hamiltonian cycle with a single vehicle. The exceptions mentioned above can be applied to different variants of the VRP (see in-depth [41]).

Aim of this Thesis: A problem defined in recent years is the Traveling Salesman Problem with Simultaneous Pickup and Delivery and Handling Cost TSPPD-H, [42]). This problem falls precisely in the definition of PD and 1-M-1 ([1-M-1 | PD | 1] for the general classification). This thesis defines in detail the problem and two metaheuristic approaches in Chapter 2: an ILS with a dynamic local search based on the frequency of use, and a granular version of the ILS (GILS). Computational experiments on instances of the literature will be presented.

1.1.2 The Knapsack Problem

Another problem studied is the Knapsack Problem (KP). Given a set N of items, each one associated with a profit p_i and weight $w_i \forall i \in N$, and given a knapsack of capacity C, the KP problem consists of selecting a set of items such that the total profit is maximized and the total weight does not exceed C. In [43] affirm that this problem is widely studied due to 3 reasons: KP can be considered as one of the simplest problems of Integer Linear Programming; it appears as a subproblem in many more complex problems; it represents many practical situations. Essential foundations for the problem were presented by Bellman in the 1950s, designing a dynamic programming algorithm, and by Danzing in 1957 proposing a continuous relaxation. Subsequently, new approaches or improvements of Dynamic Programming were designed [44], branch and bound [45], approximate algorithms [46, 47], Lagrangian relaxations [48]. Variants of the KP and related problems are widely described in the book "Knapsack problems: algorithms and computer implementations" by [49]. This last book is considered one of the most important ones written on the problem (over 5000 citations according to Google Scholar). It continues to be an important reading in the Operations Research courses. Another important book for the study of knapsack is presented at [50].

Although different approaches have solved KPs with good results, the problem remains a challenge. Indeed, depending on the correlation between the weight and the profit of each item, the problem may take longer to resolve [51]. Dynamic programming methods guarantee to find optimal solutions in pseudo-polynomial times in the worst case and is used to solve KP instances mixing constructive and destructive heuristics [52].

KP has a broadly defined set of variations and related problems. One of them is the Multiple Knapsack Problem (MKP), where more than one knapsack with a defined capacity are considered. Another problem is the Quadratic Knapsack problem (QKP, [53]), which consists of maximizing a linear objective function (called linear profit, associated with the selection of an item) and a quadratic objective function (called quadratic profit, related to the choice of a pair of items). QKP applications found in telecommunication [54] and location problems [55].

Aim of this Thesis: The Quadratic Multiple Knapsack Problem (QMKP, [56]) is an extension of the two previous described problems (and it is detailed in the chapters 3). Several approaches are presented in order to solve this problem: Polynomial-size formulations and relaxations, a matheuristic approach, and a multi-start iterated local search metaheuristic.

1.1.3 The Open-Pit Mine Production Scheduling Problem

Within the mining systems, Operations Research has aided the extraction planning, determining where to extract and when to extract. For this reasons, the economic potential that a study of the mine should consider (shape, size) and also the associated resources (roads, vehicles, excavators, among others) must be evaluated. In this way, a preliminary study must be carried out to obtain the properties of the mine.

The mine is modeled in small blocks or three-dimensional segments with different properties (ore or trash) to obtain planning [57]. Generally, deeper blocks must be removed respecting to the precedence restriction of the blocks and have a higher cost (or a positive cost). In contrast, blocks on the surface are removed at a negative cost. An example of a block model is shown in Figure 1.3. The yellow blocks (B=$\{D, E, F, G, H, I\}$) must be removed since they have a higher profit. However, the precedence restriction must be respected, i.e., the brown blocks must first removed from the previous levels (B=$\{A, B, C\}$) with negative cost.

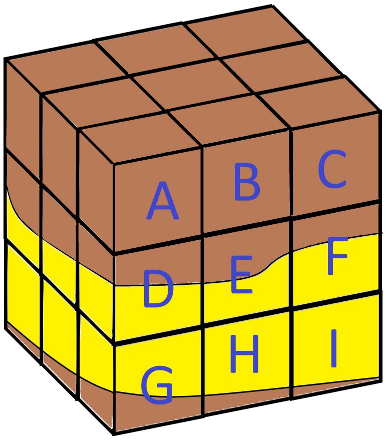


FIGURE 1.3: Example with the three-dimensional blocks

The Open-Pit Mine Production Scheduling Problem (OPMPSP) consists of scheduling the extraction of a mineral deposit divided into several smaller segments or blocks so as to maximize the Net Present Value (NPV) of the mine [58]. Let T be the set of periods, B the set of blocks, $B_{b'} \subset B$ the set of blocks preceding block b', R the set of resources, the model is described as follows:

$$max \sum_{b \in B} \sum_{t \in T} p_{bt} x_{bt} \tag{1.1}$$

s.t.

$$\sum_{s \leq t} x_{bs} \leq \sum_{s \leq t} x_{b's} \qquad \forall \quad b \in B, b' \in B_b, t \in T; \tag{1.2}$$

$$\sum_{t \in T} x_{bt} \leq 1 \qquad \forall \quad b \in B; \tag{1.3}$$

$$L_{rt} \leq \sum_{b \in B} q_{br} x_{bt} \leq U_{rt} \qquad \forall \quad t \in T, r \in R; \tag{1.4}$$

$$x_{bt} \in {}_{\text{¸}}\{0,1\} \quad \forall \quad b \in B, t \in T; \tag{1.5}$$

The problem must maximize a pre-calculated profit p_{bt} associated with a block extracted in period t (1.1) using a binary decision variable x_{bt} (1.5). The variable x_{bt} is 1 if the block $b \in B$ is extracted in the period $t \in T$, and 0 in case the block is not extracted. Constraints 1.2 consider the restriction of maintaining the precedence of the blocks, i.e. to extract block b, the preceding blocks of b (B_b) must be extracted in the same period in or a previous period. Constraints 1.3 indicate that one block must be mined only once in a single period. Finally, Constraints 1.4 are the resource constraints (for example, mine capacity or processing capacity), so the resource use is associated with a block and a resource q_{br}. In this way, the sum of the resources extracted in period t is between L_{rt} and U_{rt}.

Aim of this Thesis: In this thesis, a variant of the OPMPSP, which is called Constrained Pit Limit Problem with Phases or CPIT-P is presented. The problem was solved by a means of parallel genetic algorithm (PGA) based on the master-slave approach which is applied it to well-known instances from the literature. It is shown that the proposed algorithm can find reasonable solutions compared to previously known results for these instances.

1.2 Overview

The problems described above were approached with different techniques and presented at conferences (or accepted in journals). Table 1.1 shows a summary of the most important works of the doctoral process. Each work has the title with which it was presented, the acronym of the associated problem, the type of approach used, the associated chapter of the document, the conference (or the journal) related to the work. Describing in detail:

- "An Iterated Local Search for the Traveling Salesman Problem with Pickup, Delivery and Handling Costs" was presented at the 39th International Conference of the Chilean Computer Science Society. An article was indexed in IEEE in November 2020 in Chile

- "A Granular approach for the Traveling Salesman Problem with Pickup, Delivery and Handling Costs" was presented at the International Conference on Optimization and Decision Science (ODS) in November 2020 in Italy.

- "Polynomial-size formulations and relaxations for the quadratic multiple knapsack problem" is has been published in the European Journal of Operational Research [59].

- "Matheuristic Algorithms for the Quadratic Multiple Knapsack Problem" has been presented at the 31st European Conference on Operational Research (EURO) in Athens, July 2021.

- "Lagrangian heuristics for the Quadratic Multiple Knapsack Problem" has been presented at the 34th Conference of the European Chapter on Combinatorial Optimization (ECCO), Madrid, June 2021.

- "Open pit mining with truncated cones by a parallel genetic algorithm" was presented at IFORS 2021, and that was the internship period carried out in Santiago-Chile (3 months).

TABLE 1.1: Main works for the thesis

Title	Problem	Type Method	Chapter related	Conference(C)/Journal(J)
An Iterated Local Search for the Traveling Salesman Problem with Pickup, Delivery and Handling Costs	TSPPD-H	Heuristic	2	(C) 39th International Conference of the Chilean Computer Science Society by IEEE (2020)
A Granular approach for the Traveling Salesman Problem with Pickup, Delivery and Handling Costs	TSPPD-H	Heuristic	2	(C) International Conference on Optimization and Decision Science (ODS) by AIRO (2020)
Polynomial-size formulations and relaxations for the quadratic multiple knapsack problem	QMKP	Exact	3	(J) European Journal of Operational Research (2021)
Matheuristic Algorithms for the Quadratic Multiple Knapsack Problem	QMKP	Hybrid	3	(C) 31st European Conference on Operational Research (2021)
Lagrangian heuristics for the Quadratic Multiple Knapsack Problem	QMKP	Hybrid	3	(C) 34th Conference of the European Chapter on Combinatorial Optimization (2021)
Solving Quadratic multiple knapsack problem by multi-start iterated local search	QMKP	Heuristic	3	(C) Submitted (2022)
Lagrangian matheuristics for the Quadratic Multiple Knapsack Problem	QMKP	Hybrid	3	(J) Submitted (2022)
Open pit mining with truncated cones by a parallel genetic algorithm	CPIT-P	Heuristic	4	(C) The 22nd Conference of the International Federation of Operational Research Societies (IFORS, 2021)

Chapter 2

Iterated Local Search Algorithms for the TSPPD-H

2.1 Introduction

Routing problems with pickup and delivery have been widely studied. The single vehicle routing problem with pickups and deliveries (SVPDP-P&D), for instance, is an NP-hard problem, which implies designing a minimum cost Hamiltonian tour for a specific capacitated vehicle. Each customer may be visited once for simultaneous pickup and delivery of commodities, or twice if these operations are performed separately, applying reverse logistics [39].

A special case of SVPDP-P&D is the Traveling Salesman Problem with Pickups, Deliveries, and Handling Costs (TSPPD-H). Given $G = (V, A)$ a non-directed, complete graph, where $V = \{0, 1, ..., n\}$ is the set of vertices, and A the set of arcs; vertex 0 represents the depot and vertices $V_c = V/\{0\}$ are the customers. Each arc $(i, j) \in A$ is related to a travel time c_{ij}, and each customer $i \in V_c$ is related to α_i delivery commodity units and β_i pickup commodity units. In this problem, we assume that both commodities have the same dimension. The cost of loading/unloading a pickup commodity unit is h_a, and the cost of loading/unloading a delivery commodity unit is h_b. The capacity of the vehicle is Q, a last in, first out (LIFO) loading policy is applied. TSPPD-H requires to determine a Hamiltonian tour satisfying the capacity constraint and minimizing a global cost given by the sum of the tour cost and the handling cost related to the loading/unloading operations performed at the customers. TSPPD-H has several applications: Delivery of full bottles and pickup of empty bottles [60, 61], pickup and delivery of damaged and working bicycles in public spaces [42], pickup and delivery of new and broken machinery from hospitals for specific suppliers.

In [42] defined three essential policies for TSPPD-H. As the internal vehicle flow follows a LIFO loading policy, there is an obstruction between delivery or pickup units. If we suppose that the vehicle is a stack data structure, the beginning being the top, and the front the final part, Policy 1 always maintains pickup units at the top and delivery units in the final part. Therefore, it is always necessary to load pickup units every time a customer is visited. Policy 2, on the other hand, always maintains delivery units at the top and pickup units in the final part. Therefore, it is always necessary to load delivery units every time a customer is visited. Policy

3 requires determining the specific position of delivery or pickup units at each visit to a customer. That is, the position of pickup/delivery commodities at the top or final part of the vehicle must be decided. Thus, the handling costs are defined as the additional cost for loading delivery or pickup commodity units.

Several methods regarding TSPPD-H have been developed. In [42] presented the problem and propose three linear programming models for the three defined policies, and solve them with two exact algorithms, a Branch & Cut, and a Benders Decomposition [62] considering up to 25 customers. Clearly, it was necessary to generate new approaches to solve larger problems. The policies used in [42] are also studied in [63], where two approaches are defined in order to get the handling cost: a dynamic programming algorithm with complexity $O(n^2)$ for computing a route; the second algorithm is a heuristic approach with complexity $O(n)$, that allows to identify whether a route is promising or not. Furthermore, [63] presented three metaheuristics for TSPPD-H, an Iterated Local Search (ILS, [27]), a Tabu Search [64], and an Iterated Tabu Search based on the first two approaches. The main aim of their paper is to compare the three metaheuristics and the handling cost algorithms. Thus, similar termination criteria were defined, maintaining the simple algorithms. Recently, [65] defined a version with several routes and vehicles (VRP – Vehicle Routing Problem), solving the instances of TSPPD-H proposed by [63] with an Adaptive Large Neighborhood Search (ALNS, [66]), finding new best known solutions (BKS). However, the large-scale instances are solved in 7 hours with the DP approach and in 30 minutes with the heuristic approach.

A new approach based on ILS can be applied. As a matter of fact, considering fluctuations on the elementary procedures of an ILS would contribute to explore new solution spaces for TSPPD-H. In this chapter an ILS with the Granular version for TSPPD-H is presented. This algorithm proposes a roulette wheel method for the perturbation and a frequency-based local search procedure. In section 2.2, a variation of the handling cost algorithm for the local search is described. Section 2.3 details a new ILS for TSPPD-H, called ILS-F. In section 2.4 the granular version of the ILS-F, called GILS, is presented. Section 2.5 reports computational results and Section 2.6 presents our conclusions and future directions. A preliminary version of this ILS metaheuristic has been presented in [67].

2.2 The Handling Cost

In the TSPPD-H, for each vehicle it is necessary to analyze the handling cost at each customer and to consider that the commodities follow a LIFO policy, i.e., the last commodity that enters the vehicle must be the first to be unloaded. In this way, each time a customer is visited, an additional time is spent for the loading and unloading management on the vehicle, and this time must be considered in the global cost function.

The formulation of the handling cost was proposed in [42]. The handling cost depends on the additional operations that must be performed for unloading and reloading one unit of commodity at a customer location. The optimal solution of the TSPPD-H is then a Hamiltonian circuit on G that minimizes the sum of the cost of the total travel time and of the additional operation times.

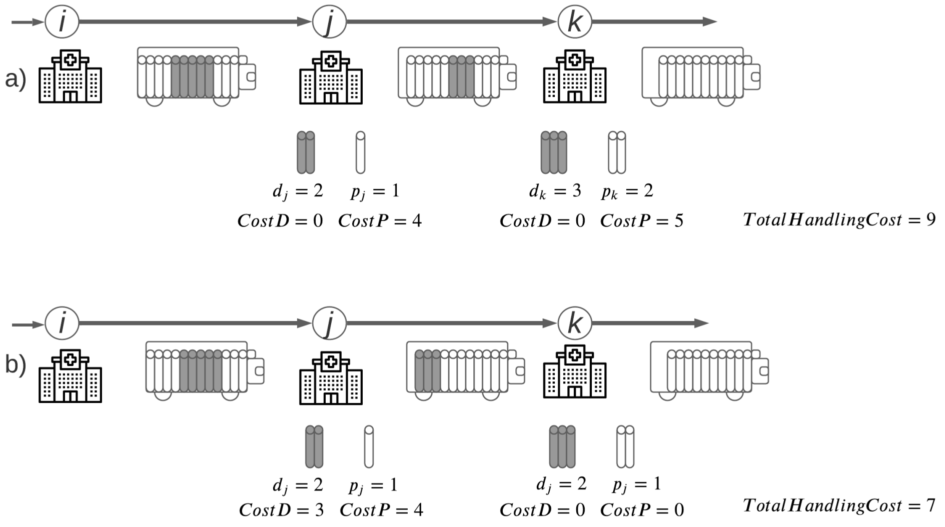


FIGURE 2.1: A Handling Cost example for delivery and pickup of oxygen tank between hospitals.

Figure 2.1 shows an example of the handling cost concept applied to oxygen tank supply vehicles for hospitals. We assume that the cost h_a and h_b are both equal to 1. For this case, there are two identical sub-routes with hospitals i, j, k, and the load management must be carried out for hospitals j and k. Empty oxygen tanks inside the vehicle are represented in white tanks and must be collected from each hospital. Full oxygen tanks are gray and must be delivered to each hospital. For routes $a)$ and $b)$, when the vehicle goes the hospital i to hospital j, it has four empty oxygen tanks at the beginning of the vehicle, five full oxygen tanks in the middle of the vehicle, and three empty oxygen tanks at the end of the vehicle. When the vehicle arrives at hospital j, pickups and delivery operations must be performed; for this example, hospital j has a delivery equal to 2 and a pickup equal to 1 unit. In this way, considering the current state of the vehicle, we could consider two policies:

- Route a):
  - Unload from the beginning of the vehicle four empty oxygen tanks.
  - Deliver two full oxygen tanks of the current load to satisfy the demand in j.
  - Reload at the beginning of the vehicle the four empty tanks previously unloaded.
  - Pickup at the beginning of the vehicle the empty tank corresponding to the demand of j.
- Route b):
  - Unload from the beginning of the vehicle four empty oxygen tanks.
  - Deliver two full oxygen tanks of the current load to satisfy the demand in j.

- Unload from the beginning of the vehicle the three remaining full oxygen tanks.
- Reload at the beginning of the vehicle the four empty tanks previously unloaded.
- Pickup at the beginning of the vehicle the empty tank corresponding to the demand of j.
- Reload at the beginning of the vehicle the three full oxygen tanks previously unloaded.

For route a) in hospital j, an additional handling cost is generated for pickup commodities ($CostP$) equal to four, since four empty oxygen tanks must be unload. In contrast, the additional cost for delivery commodities ($CostD$) is not generated, since only one full oxygen tank was unloaded corresponding to the demand of the hospital j, and there are no additional costs related to the commodities. For route b) in hospital j, an additional handling cost equal to four is generated for pickup commodities, since four empty oxygen tanks must be unload. On the other hand, the additional cost for delivery commodities is three since three additional full oxygen tanks are unloaded and must be reloaded into the vehicle. Observe, that the vehicle's state after leaving hospital j is different for each route. The handling cost of hospital j for route a) is 4 ($CostP + CostD = 4 + 0 = 4$), while for route b) the handling cost is 7 ($CostP + CostD = 4 + 3 =$7). In conclusion, more additional movements were made on route b).

The vehicle must continue its the route and arrive at the hospital k . At this point, routes a) and routes b) have different additional handling costs since the vehicle's state that arrives from hospital j impacts the load management cost. For route a), the vehicle must unload five empty oxygen tanks that are carried over from hospital j, unload three full oxygen tanks corresponding to the demand of hospital k, and finally reload the five empty oxygen tanks initially unloaded plus the empty tank corresponding to the demand of hospital k. In this case, the additional handling cost at hospital k is 5, and the global handling cost for this subroute is 9. For route b), the vehicle must only unload three full oxygen tanks and upload one empty oxygen tank, both corresponding to the demand of hospital k. For this stop, no additional costs are generated, and the final handling cost for this sub-route is 7.

The handling cost value depends on each decision made along the route. For the previous example, it is assumed that the vehicle's state when it leaves hospital i is the same for routes a) and b). However, this is not necessarily the case since the accumulated handling cost depends on the additional costs generated by previous customers. In this way, a route can have many associated handling costs. In [63] proposed the first approaches in order to get the additional handling cost. The first is a quadratic time dynamic programming (DP) algorithm, and the second is a linear time heuristic algorithm. The DP algorithm is an approach that can be improved and adapted for heuristic search moves. The following sections describe the DP algorithm and its improvements.

2.2.1 Dynamic programming algorithm

The two most important processes of the algorithm are the following:

- Recursive formulation: Given a Hamiltonian route with i and j the customer indexes along the route, $hc(i)$ is the optimal handling cost between customers $i+1$ to n with policy 2 applied to customer i. Also, p_{ij} is the handling cost associated with applying policy 1 between customers $i+1$ to $j-1$. For policy 2 to customer j, a recursive formulation of dynamic programming is as follows:

$$(DP);\ hc(i) = \min_{j \in \{i+1,n\}} \{p_{ij} + hc(j)\}, \forall i \in \{0, ..., n-1\} \tag{2.1}$$

$$hc(n) = 0 \tag{2.2}$$

  Remember that i and j are customer indexes along a previously defined route and that the termination condition of the DP algorithm is n with handling cost equal to 0. Generate matrix p: In order to generate the matrix elements $p_{ij}, \forall i, j \in \{1, ..., n\}$ three state variables are used:

  - α' as the number of delivery units onboard the vehicle.
  - β' as the number of pickup units onboard the vehicle.
  - θ as the accumulated cost of applying policy 1.

Algorithm 1 shows the procedure to calculate the matrix p and the execution of the dynamic programming algorithm. The state variables α', β', and θ are initialized within the main loop (lines 3,4, and 5). Between lines 9 to 20, the values of the matrix elements $p_{i,j}$ are calculated as follows:

- The state variable α' is updated, containing the remaining delivery units for the future customers. All delivery units corresponding to customer demand j are unloaded regardless of the policy (line 10).

- The algorithm checks if the customer j has pickup units demand and if the vehicle has pickup units on board from other customers. If the global number of pickup units units ($\beta' + \beta_j$) is greater than 0, p_{ij} is set equal to the accumulated cost of applying policy 1 up to this customer of the route plus the cost of the management associated to the units onboard the vehicle (pickup and delivery). If the sum $\beta' + \beta_j$ is equal to 0, p_{ij} is equivalent to the accumulated cost of applying policy 1.

- If the delivery request of the customer j is greater than 0, the state variable θ is updated.

- Finally, the state variable β' is updated as if policy 1 were used.

Note that each external for-loop has complexity $O(n^2)$. In [63] stated that, when using this algorithm within heuristic search moves, the complexity can reach $O(n^4)$.

Algorithm 1 DP algorithm by [63]

```
Input: $\alpha$,$\beta$,$h_a$,$h_b$
Output: $hc_0$ (Optimal Value)
// Generating $p$
for $i = 0$ to $n - 1$ do
    $\beta' = 0$
    $\theta = 0$
    $\alpha' = \alpha_0$
    for $j = 1$ to $i$ do
        $\alpha' = \alpha' - \alpha_j$
    end for
    for $j = i + 1$ to $n$ do
        $\alpha' = \alpha' - \alpha_j$
        if $\beta'$ + $\beta_j > 0$ then
            $p_{i,j}$ = $\theta$ + ($h_a$*$\alpha'$) + ($h_b$*$\beta'$)
        else
            $p_{i,j}$ = $\theta$
        end if
        if $\alpha_j > 0$ then
            $\theta = \theta + (h_b * \beta')$
        end if
        $\beta' = \beta' + \beta_j$
    end for
end for
// DP procedure
$hc(n) = 0$
for $i = n - 1$ to $0$ do
    for $j = i + 1$ to $n$ do
        if $hc(i) > p_{ij} + hc(j)$ then
            $hc(i) = p_{ij} + hc(j)$
        end if
    end for
end for
return $hc(0)$
```

$$p = \begin{bmatrix} 0 & 8 & 10 & 18 & 31 \\ 0 & 0 & 8 & 10 & 17 \\ 0 & 0 & 0 & 6 & 9 \\ 0 & 0 & 0 & 0 & 2 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

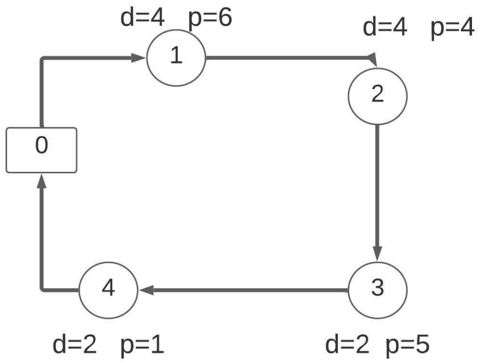


FIGURE 2.2: Final matrix p for the TSPPD-H using DP.

An example of the handling cost calculation is shown in Figure 2.2. At the bottom of the figure, a route with four customers and the depot with simultaneous pickup and delivery requirements is presented. To apply the exact approach, input data must be precalculated $\alpha = \{\alpha_0, 4, 4, 2, 2\}$ and $\beta = \{b_0, 6, 4, 5, 1\}$ with $\alpha_0 = \sum_{i \in V_c} \alpha_i = 12$ and $b_0 = \sum_{i \in V_c} \beta_i = 16$. Considering this example with $h_a = h_b = 1$, the output matrix p generated by Algorithm 1 is shown at the top of the figure.

With the matrix p and its candidate handling cost values, a recursive tree is generated and shown in Figure 2.3. Note that each value $hc(i)$ requires the values hc_j h_j $\forall\ j \in \{i+1, ..., n\}$ with $h(n) = 0$. For each operation, the minimum value is selected, which defines the definitive handling cost value for the route between the depot and each customer (colored boxes). Finally, an output vector $hc = \{8, 8, 6, 2, 0\}$ is generated and the optimal value of handling cost is $hc(0) = 8$.

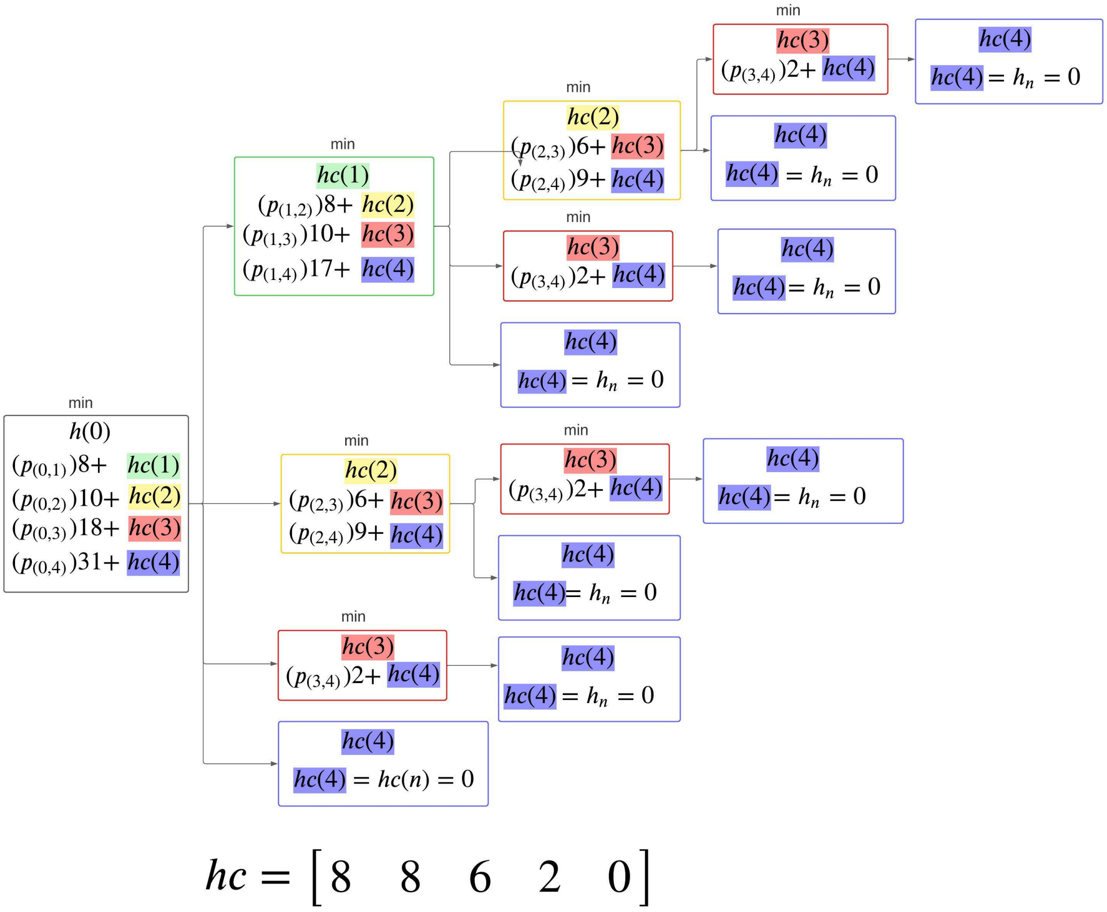


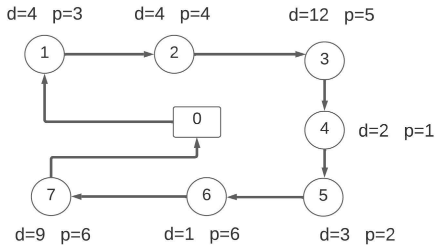
FIGURE 2.3: Recursive Tree of DP Approach.

2.2.2 Improved DP Algorithm

The original DP algorithm from [63] can be adapted in order to be used for local search methods. It should be noted that the improvement to the algorithm does not change the value and logic of the process; it only decreases the execution time. We show an example of this improvement using Figure 2.4. The handling cost vector is calculated as exemplified in the recursive tree of Figure 2.3.

Note that the value of each handling cost depends on its predecessor. In this way, original values of vector *hc* and matrix *p* can be maintained during a local search independently of the type of movement. For the example above, customers 1 and 2 could be swapped. However, the rest of the route maintains the handling costs calculated with the previous values of *hc* and *p*. As some initial arcs of the original route are changed, some rows of matrix *p* must be recalculated, and also some new values for vector *hc* must be recalculated. This principle allows to generate an improvement on the DP algorithm previously described. Figure 2.5 shows the details of this movement.

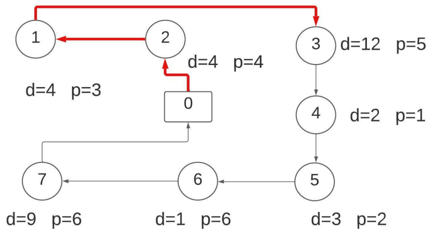
$$p = \begin{bmatrix} 0 & 31 & 30 & 25 & 35 & 45 & 59 & 71 \\ 0 & 0 & 31 & 23 & 30 & 37 & 48 & 57 \\ 0 & 0 & 0 & 19 & 22 & 25 & 32 & 37 \\ 0 & 0 & 0 & 0 & 25 & 23 & 25 & 25 \\ 0 & 0 & 0 & 0 & 0 & 12 & 13 & 12 \\ 0 & 0 & 0 & 0 & 0 & 0 & 12 & 9 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$



$$hc = \begin{bmatrix} 25 & 23 & 19 & 23 & 12 & 9 & 1 & 0 \end{bmatrix}$$

FIGURE 2.4: Original route.

$$p = \begin{bmatrix} 0 & 31 & 31 & 26 & 36 & 46 & 60 & 72 \\ 0 & 0 & 31 & 22 & 28 & 34 & 44 & 52 \\ 0 & 0 & 0 & 19 & 22 & 25 & 32 & 37 \\ 0 & 0 & 0 & 0 & 25 & 23 & 25 & 25 \\ 0 & 0 & 0 & 0 & 0 & 12 & 13 & 12 \\ 0 & 0 & 0 & 0 & 0 & 0 & 12 & 9 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$



$$hc = \begin{bmatrix} 26 & 22 & 19 & 23 & 12 & 9 & 1 & 0 \end{bmatrix}$$

limit

FIGURE 2.5: New route with improvement (red)

Algorithm 2 DP algorithm for Local Search

```
Input: $\alpha$,$\beta$,$h_a$,$h_b$,costCurrent, costRoute, $limit$, $hc$
Output: $hc_0$+ costRoute
 1: for $i = limit$ to 0 do
 2:     $\beta'$ = 0
 3:     $\theta$ = 0
 4:     $\alpha' = \alpha_0$
 5:     for $j = 1$ to $i$ do
 6:         $\alpha' = \alpha' - \alpha_j$
 7:     end for
 8:     for $j = i + 1$ to $n$ do
 9:         $\alpha'$ = $\alpha'$ - $\alpha_j$
10:         if $\beta'$ + $\beta_j > 0$ then
11:             $p_{i,j}$ = $\theta$ + ($h_a$*$\alpha'$) + ($h_b$*$\beta'$)
12:         else
13:             $p_{i,j}$ = $\theta$
14:         end if
15:         if $\alpha_j > 0$ then
16:             $\theta = \theta + (h_b * \beta')$
17:         end if
18:         $\beta' = \beta' + \beta_j$
19:     end for
20:     min = $\infty$
21:     //DP starts from route change
22:     for $k = i + 1$ to $n$ do
23:         if min $> hc(k) + p_{i,k}$ then
24:             min = $hc(k) + p_{i,k}$
25:         end if
26:     end for
27:     //If the current value has not decreased from the original path
28:     if costCurrent < (costRoute + min) then
29:         return costCurrent
30:     else
31:         $hc(i)$ = min
32:     end if
33: end for
34: return $hc(0)$+ costRoute
```

Algorithm 2 shows the details of the improvement. This process has new input parameters: *costCurrent* is the current cost of the original tour corresponding to the sum of the cost of the original tour and of the optimal handling cost of the original tour; *costRoute* is the cost of the tour applying the movements to the original tour; *limit* is an integer parameter defining the initial customer to be considered; and *hc* is the current vector of handling costs of the original tour. Note that the core of the algorithm (lines 2-19) is not changed but the considered customers depend on the value of *limit*. Then, for each considered customer (row) *i*, the DP routines are applied (lines 20 -26). Finally, a boundary condition for the algorithm is applied, for which the cost of the route *costRoute* is added to the handling cost on the customer *i* (*min*, line 28). If this value is less than the original cost *costCurrent*, the algorithm cut the execution and continues with another movement (returning *costCurrent*); otherwise, the algorithm continues calculating the optimal handling cost.

This algorithm will be used to calculate the handling costs, within the procedures defined for the ILS algorithm. In the next section, the metaheuristics will be defined in more details.

2.3 The Iterated Local Search Algorithm

2.3.1 Existing algorithm

Algorithm 3 ILS by [63]

```
Input: $s_0$, $N_{rand}$ , $N_{iter}$,
Output: $bestSolution$
 1: costCurrent = cost($s_0$)
 2: $bestSolution$ = $s_0$
 3: for $i = 1$ to $N_{iter}$ do
 4:     Tour = $bestSolution$
        $Perturbation$ :
 5:     for $r = 1$ to $N_{rand}$ do
 6:         $j$, $k$=RandomNumber(1,$|V_c|$)
 7:         if ($j == k$) then
 8:             $p$ =RandomNumber(1,$|V_c|$) / $p \neq j$
 9:             Tour = Relocate($Tour$,$j$,$p$)
10:         else
11:             Tour = Reverse($Tour$,$j$+1,$k$)
12:         end if
13:     end for
        $Local\ Search$ :
14:     while improvement do
15:         Tour = Relocate(Tour)
16:         Tour = 2-Opt(Tour)
17:     end while
18:     if ($cost(Tour) < costCurrent$) then
19:         costCurrent = cost(Tour)
20:         $bestSolution$ = Tour
21:     end if
22: end for
23: return $bestSolution$
```

The Iterated Local Search approach (ILS) was introduced by [27] and has been used for several routing problems related to TSP [68–70]. Algorithm 3 shows an ILS pseudocode designed in [63]: the algorithm receives an initial solution s_0 generated by the Symmetric TSP software Concorde [71], a number N_{rand} of iterations of perturbation and a number N_{iter} of iterations of ILS. The variable *costCurrent* is initialized with the cost of the initial solution s_0, and the best route or *bestSolution* is initialized with the solution s_0 (lines 1 and 2). Later, within the internal cycle of the ILS, two random numbers, j and k are generated. If $j = k$, a new random number p (with $p \neq j$) is generated and the customer j is relocated in position p; otherwise, a reverse function between the customers $j + 1$ and k is applied. Then, the 2-Opt [72] and Relocate improve heuristics are used on the perturbed route until no further improvement is found.

Relocate and Reverse moves are considered in the perturbation. Figure 2.6 shows an original route (with indices from 1 to 7 representing the sequence of its customers), and three random perturbation moves: Swap(i, j) moves the customer from position i to position j and vice versa, Relocate(i, j), which moves the customer from position i to position j, and Reverse(i, j) which changes a direction subroute (subroute between i, j). The moves considered in the ILS by [63] are Relocate and Reverse. Swap is considered by the new metaheuristics detailed in 2.3.3.

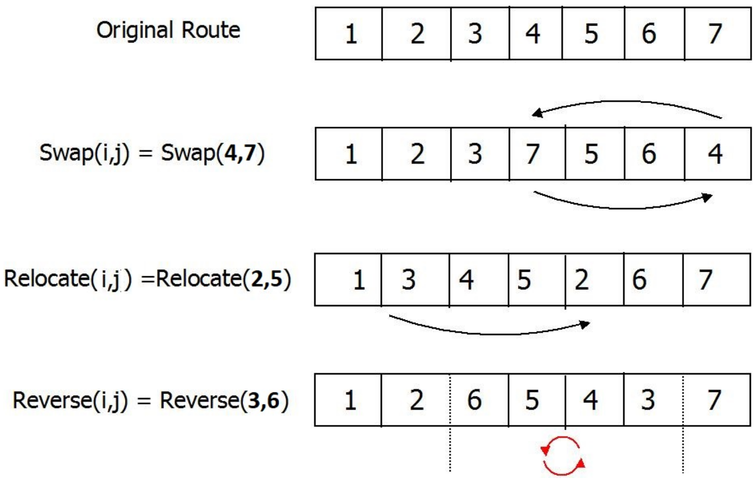


FIGURE 2.6: Perturbation moves

The previous algorithm can be used to define a new ILS. To do so, the two defined procedures must be taken into account.

2.3.2 Iterated Local Search using Frequency (ILS-F)

A new ILS called ILS-F is introduced to solve the TSPPD-H. The new approach considers new procedures that improve the performance of the metaheuristics and that are different from those used in the ILS of [63]:

- It is possible to generate an initial feasible solution using a framework that solves the TSP with Pickups and Delivery. Although Concorde can find good initial solutions for the TSP, it does not consider the demands of the customers, which are vital for the proper management of the handling costs.
- This initial solution can be improved by a local search based on the best neighborhood approach (called *GreedyBasedLS*).
- Subsequently, a perturbation based on a not crooked roulette (called *RoulettePerturbation*) can be applied. In this process, the perturbation must select a random movement without preference among a set of possible movements.
- A local search based on the frequency of use (called Local search with frequency or *LSF*) can be applied. This adaptive local search allows to improve the performance of a search. To do this, the algorithm considers whether a neighborhood improves the current solution; if this is true, the neighborhood increases its probability to be executed in the next search iteration.
- An acceptance criterion can be applied based on the Simulated Annealing approach. This criterion allows us to accept solutions that do not improve the objective function but can allow us to escape from local optima.

Note that the higher-level components listed above are not part of a standard ILS. ILS-F tries to guide the search with the same initial solution, not randomly perturbing it as done in the ILS from [63]. For ILS-F, a local search based on the best neighborhood is used, and a local search executes the best moves using frequency. Algorithm 4 shows the pseudocode of ILS-F.

Algorithm 4 ILS-F: Main Scheme

```
Input: $Data$, $N_{itrL}$ , $N_{per}$, $N_{iter}$,$\epsilon$,$t_0$
Output: $s_b$
 1: $s_0 = LKH(Data)$
 2: $s_* = GreedyBasedLS(s_0)$
 3: $t = t_0$
 4: $s_b = s_*$
 5: $costS_* = costS_b = cost(s_*)$
 6: $fls = equivalentFrequency()$
 7: $history_{val} = \emptyset$
 8: for $it = 1$ to $N_{iter}$ do
 9:     $s_+ = RoulettePerturbation(s_*, history_{val}, N_{per})$
10:     $(s_{++}, fls) = LSF(s_+, fls, N_{itrL})$
11:     if $(cost(s_{++}) < costS_b)$ then
12:         $s_b = s_* = s_{++}$
13:         $costS_b = costS_* = cost(s_{++})$
14:         $history_{val} = \emptyset$
15:         $history_{val}.enqueue(costS_b)$
16:         $fls = equivalentFrequency()$
17:         $t = t_0$
18:     else if $(cost(s_{++}) < costS_* || A(cost(s_{++}), costS_*, t, \epsilon))$ then
19:         $s_* = s_{++}$
20:         $costS_* = cost(s_{++})$
21:         $t = max(t * \epsilon, 0)$
22:     else
23:         $t = max(t * \epsilon, 0)$
24:     end if
25: end for
26: return $s_b$
```

ILS-F receives several input parameters for its execution and initial processing. The instance data (*Data*), the number of iterations N_{itrL} for the *LSF*, the number of iterations N_{per} for the perturbation, the corresponding number N_{iter} of iterations of the main loop, ϵ the value t_0 used for the acceptance criterion based on a cooling configuration in the decreasing variable t that is explained in detail in section 2.3.6. The initial solution s_0 is generated by the Lin-Kernighan TSP heuristic (LKH), version 2.0.9 [37]. Then, a *GreedyBasedLS* is performed, obtaining a solution s_*, which will be the current search solution. After that, the variables to be used in the main loop are updated: s_b represents the best known solution and $costS_*$ and $costS_b$ are the cost of the solution s_* and s_b respectively. The variable fls (line 6) is a triple of pairs of the form $(< 2 - Opt, \delta >, < Relocate, \zeta >, < Swap, \eta >) / \delta, \zeta, \eta \in [0,1], \delta + \zeta + \eta = 1$, where δ, ζ and η are the frequencies of moves "2-Opt", "Relocate" and "Swap", respectively (see section 2.3.4). Initially the three moves have equivalent frequencies

$(\delta = 1/3, \zeta = 1/3, \eta = 1/3)$. Finally, the variable $history_{val}$ is a queue structure that saves the values of the target function (pair $< TSPcost, HandlingCost >$) generated by the perturbation.

The main loop contains the perturbation and local search procedures defined for ILS-F. $RoulettePerturbation$ uses the current solution and generates a new neighboring solution s_+ in a number N_{per} of iterations. Subsequently, the routine local search with frequency LSF is used, which performs two main processes: LSF improves the perturbation solution s_+ by generating s_{++}, and it also updates the variable fls with new frequencies for the following search process. The new current solution may be accepted under three criteria:

- If the solution is the best from the search: All search variables are updated and the variable fls, the variable $history_{val}$, and the variable t are reset (lines 12-17).
- If a local optimum is found or an acceptance criterion is fulfilled (section 2.3.6, A function). Only the variables storing the current search solutions are updated (line 18-21).

A flow-chart of the algorithm is shown in Figure 2.7.

2.3.3 Roulette Perturbation

The $RoulettePerturbation$ algorithm makes random perturbations using three moves. Algorithm 5 details the process and receives a solution s_*, a queue with the values from the last perturbations $history_{val}$ and the number N_{per} of iterations for perturbation. The internal loop randomly performs 3 types of moves: $Reverse$, reverting a sub-route between positions $j+1$ and k; $Relocate$, selecting a customer j and putting it on position k, and $Swap$, exchanging the customer in position j with that in position k. It is worth noting that the number of moves made by the internal loop only depends on N_{per}. Once the random moves are applied, it is verified that the route cost and the handling cost from s_{per} are not contained within the variable $history_{val}$. The process behind this verification is based on the pair $< TSPcost, HandlingCost >$ having a high chance of being unique data for a route, i.e., a route generates a route cost and a handling cost unique to both together. Thus, routes containing both costs in $history_{val}$ are avoided for not applying a redundant local search. If the route is valid, the first pair input to $history_{val}$ (the oldest one on queue) is removed, and the new perturbation route is returned. Otherwise, the internal algorithm loop is executed again until a valid perturbation is found.

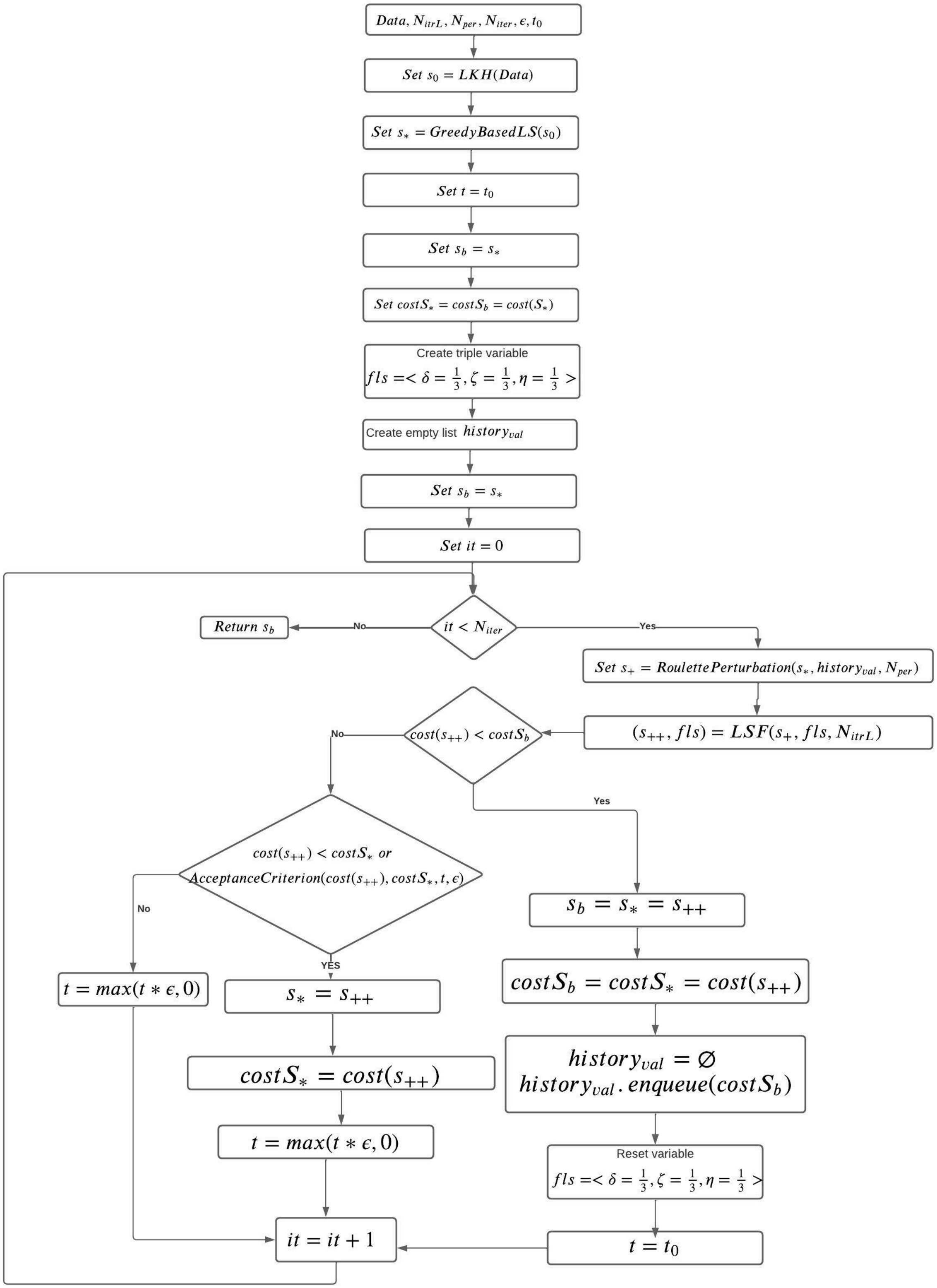

$Data, N_{itrL}, N_{per}, N_{iter}, \epsilon, t_0$
$Set\ s_0 = LKH(Data)$
$Set\ s_* = GreedyBasedLS(s_0)$
$Set\ t = t_0$
$Set\ s_b = s_*$
$Set\ costS_* = costS_b = cost(S_*)$
Create triple variable
$fls =< \delta = \frac{1}{3}, \zeta = \frac{1}{3}, \eta = \frac{1}{3} >$
Create empty list $history_{val}$
$Set\ s_b = s_*$
$Set\ it = 0$
$it < N_{iter}$
No
$Return\ s_b$
Yes
$Set\ s_+ = RoulettePerturbation(s_*, history_{val}, N_{per})$
$(s_{++}, fls) = LSF(s_+, fls, N_{itrL})$
$cost(s_{++}) < costS_b$
No
Yes
$cost(s_{++}) < costS_*\ or$
$AcceptanceCriterion(cost(s_{++}), costS_*, t, \epsilon)$
No
YES
$t = max(t * \epsilon, 0)$
$s_* = s_{++}$
$costS_* = cost(s_{++})$
$t = max(t * \epsilon, 0)$
$s_b = s_* = s_{++}$
$costS_b = costS_* = cost(s_{++})$
$history_{val} = \varnothing$
$history_{val}.enqueue(costS_b)$
Reset variable
$fls =< \delta = \frac{1}{3}, \zeta = \frac{1}{3}, \eta = \frac{1}{3} >$
$t = t_0$
$it = it + 1$

FIGURE 2.7: FlowChart of the ILS-F

Algorithm 5 RoulettePerturbation

```
Input: $s_*$,$history_{val}$, $N_{per}$,
Output: $s_+$
$s_+ = s_*$
$ValidPerturbation$ = $False$
while ($ValidPerturbation$==$False$) do
    $s_{per} = s_*$
    for $i = 1$ to $N_{per}$ do
        $per = RandomRoulette(1,2,3)$
        $(j,k) = RandomUniform(1,n)\ /\ j < k$
        if ($per == 1$) then
            $s_{per}$ = Reverse($s_{per}$,$j$+1,$k$)
        else if ($per == 2$) then
            $s_{per}$ = Relocate($s_{per}$,$j$,$k$)
        else if ($per == 3$) then
            $s_{per}$ = Swap($s_{per}$,$j$,$k$)
        end if
    end for
    if $(costTSP(s_{per}), costHC(s_{per}) \notin history_{val})$ then
        $ValidPerturbation$ = $True$
        $s_+ = s_{per}$
    end if
end while
$history_{val}.dequeue()$
return $s_+$
```

2.3.4 Elementary Heuristics

Given a route to apply a move, three heuristics may be described for TSPPD-H: Relocate, 2-Opt, and Swap. Relocate implies the best relocation of a single customer during the route, therefore, the number of possible relocations is $|V_c| * (|V_c| - 1)$. The move 2-Opt removes two arcs from the route and replaces them with two other arcs, thus, a single enhanced Hamiltonian cycle is formed, and the number of possible moves in this neighborhood is $((|V_c| + 1) * (|V_c| - 2))/2$. Swap consists of finding the best exchange of positions for two customers in the route, therefore, the number of exchange moves in each neighborhood is $|V_c| * (|V_c| - 1)/2$. For our ILS-F, each heuristic is executed until the first improvement is found, i.e., the total number of possible neighborhoods is only executed in a worst-case scenario.

The effectiveness study made in [63] suggests that it is better to use two of the three described heuristics. As a matter of fact, for this problem, Relocate and 2-Opt were the best among three considered heuristics. In this way, two neighborhoods are used in the metaheuristics of [63] and one (i.e. Swap) is completely discarded.

However, for the procedure LSF (described in section 2.3.5), we consider all three heuristic but, only two are performed during the local intensification search, and the third one is executed only when there is no possibility of improving the current solution. The two best moves are defined based on their use frequency during the searching. Our hypothesis is based on the effectiveness of the heuristics depending on the structure of the instance, given by the travel time between the customers, the commodities on-board, the commodities not on-board. Now, the use of the heuristics within the local search is explained.

2.3.5 Local search methods

Algorithm 6 GreedyBasedLS

```
Input: $s_0$
Output: $s_*$
1:  $s_* = s_0$
2:  $costS_* = cost(s_0)$
3:  $improve$ = $True$
4:  while ($improve$==$True$) do
5:      $s_\delta = 2 - OptHeu(s_*)$
6:      $s_\zeta = RelocateHeu(s_*)$
7:      $s_\eta = SwapHeu(s_*)$
8:      $cost_f = min(cost(s_\delta), cost(s_\zeta), cost(s_\eta))$
9:      $best_{route} = best(s_\delta, s_\zeta, s_\eta)$
10:     if ($cost_f < costS_*$) then
11:         $costS_* = cost_f$
12:         $s_* = best_{route}$
13:     else
14:         $improve$ = $False$
15:     end if
16: end while
17: return $s_+$
```

The algorithm *GreedyBasedLS* (see Algorithm 6) chooses the best solution by applying each heuristic to a current solution. The Algorithm 6 receives a solution s_0 as parameter. After the updating of the algorithm variables (current solution s_*, and $costS_*$), a loop is executed, where all the three described heuristics are applied to the current solution (lines 5-7) getting the routes s_δ, s_ζ and s_η . Then, the best route is selected (lines 8-9) and the solution improved is compared with the current solution (lines 10-15). The loop end criterion is applied when no further improvements over s_0 can be found.

The local search frequency (LSF) algorithm iteratively executes the heuristics based on the use of the last iterations. Algorithm 7 receives a solution s_+, the current use frequencies fls and the number of iterations of the local search N_{itrL}. As with the previous algorithms, the current solution variables of the algorithm must be updated. In addition, a variable fls_{new} to be used to update the new frequency values according to the local search to be performed is generated. First, the internal loop of the algorithm (lines 5-20) takes each heuristic *heu* from fls, assuming it is already ordered according to the prior iteration use frequency (it is worth noting that for the first iteration all the frequencies are equal. However, 2-Opt and Relocate are executed first, based on [63]). Then, the chosen heuristic is applied, and the solution is stored in s_{heu} (line 7). If the solution is feasible (line 9) and suffered changes due to the heuristic (line 10), 3 actions are performed (lines 11-14): the variable fls_{new} with the applied heuristic, the current solution variables are updated, and the counter of improvements is increased of one unit. It is also worth mentioning that the internal loop may execute all three moves if, and only if, any (or none) of the first two heuristics does not achieve an improvement of the current solution. Once the heuristic move loop is finished, the variable fls is updated with the new values (line 22, sort process) and it is verified that the new current solution is the best solution of the search. If this falls in a local optimum, the main loop is terminated without completing the N_{itrL} iterations (lines 23-28). The LSF algorithm returns the final solution found and the updated frequency.

Algorithm 7 LSF

```
Input: $s_+$,$fls$,$N_{itrL}$
Output: $s_{++}$,$fls_{new}$
 1: $s_{++} = s_{LSF} = s_+$
 2: $fls_{new} = fls$
 3: for ($it = 0$ to $N_{itrL}$) do
 4:     $improve$=0
        heuPair is a pair structure <frequency ratio, heuristic> :
 5:     for ($heuPair$ in $fls$) do
 6:         $heu = HeuPair.getHeuristic()$
 7:         $s_{heu} = applyHeu(heu, s_{LSF})$
 8:         $costS_{heu} = cost(s_{heu})$
 9:         if ($fesaible(s_{heu})$) then
10:             if ($costS_{heu} \neq costS_{LSF}$) then
11:                 $Update(fls_{new}, heu)$
12:                 $s_{LSF} = s_{heu}$
13:                 $costS_{LSF} = costS_{heu}$
14:                 $improve = improve + 1$
15:             end if
16:         end if
17:         if ($improve$==2) then
18:             break
19:         end if
20:     end for
21:     $fls = fls_{new}$
        The fls structure is sorted based on the new frequency ratio :
22:     $sort(fls)$
23:     if ($costS_{LSF} < costS_{++}$) then
24:         $s_{++} = s_{LSF}$
25:         $costS_{++} = costS_{LSF}$
26:     else
27:         break
28:     end if
29: end for
30: $sort(fls_{new})$
31: return $s_{++}, fls_{new}$
```

2.3.6 Acceptance Criterion

The acceptance criterion for the ILS-F is based on a Simulated Annealing cooling setup. This metaheuristic defined in [73] and [22], is based on the controlled cooling of materials. However, the cooling setups for this metaheuristic are different and have been used in several works of the literature [74, 75]. For this ILS-F, a configuration expressed in equation 2.3 has been defined, where t_0 and t_f are initial and final temperatures, respectively, N_{iter} is the number of iterations, ϵ is the cooling factor, s_{++} is the current solution and s_* is the new solution to consider accepting in the search.

$$\boxed{\begin{array}{ll} \epsilon = (\frac{t_f}{t_0})^{\frac{1}{N_{iter}}} & t_0 < t_f < 0 \\ t_{k+1} = \epsilon * t_k & 0 < k < N_{iter} \\ cost(s_*) < cost(s_{++}) - t_k * ln(U(0,1)) & \end{array}} \tag{2.3}$$

- t_0 and t_f are initial and final temperatures, respectively.
- N_{iter} is the number of iterations.
- ϵ is the cooling factor.
- s_{++} is the current solution.
- s_* is the new solution to consider for acceptance in the search.

The convergence of the temperature allows the change of neighborhood within the ILS. The criterion is applied for each iteration and it is necessary to check the cooling temperature each time a solution is obtained from the local search.

2.4 Granular Iterated Local Search

The Granular Iterated Local Search (GILS) is based on the ILS metaheuristic [27] and the granular search concept proposed by [32]. The essential idea of this algorithm is to use ILS but with a granular search within the neighborhoods in the local search. The concept of granularity is based on eliminating a priori arcs of the problem that may not be part of the optimal solution, reducing the search computation time. For example, the algorithm could calculate an average arc value and remove the arcs having values greater than threshold given by a fixed percentage of the average arc value. This idea proved to be effective for the VRP obtaining good results [32].

2.4.1 The granular approach

The granular search concept consists of evaluating only those edges which have larger probabilities to be part of a good solution. The classical granular approach proposed by [32] uses a travel time based criterion to decide which edges will be evaluated. Nevertheless, we propose an hybrid criterion, which combines the travel time based-criterion with the similarity of the demand between customers. Thus, a new granular cost $\lambda_{i,j}$ associated to each edge (i,j), connecting two customers i, j in the graph, is generated. The new granular cost will be used to guide the local search and the neighborhood exploration; it can be computed as follows:

$$\lambda_{ij} = \delta * (\frac{c_{ij}}{MAX_C}) + \eta * (\frac{|q_i - q_j|}{2 * MAX_Q}) \tag{2.4}$$

Where, MAX_C is the maximum traveling cost computed over all edges in the graph, q_i is the absolute value of the difference between the pickup and the delivery demand of a customer i (i.e. $|\alpha_i - \beta_i|$), MAX_Q represents the largest absolute difference between q_i and q_j computed out of all pairs of customers $(i,j) i \neq j, i, j \in V_c$. The parameters δ and η represent the weights respectively of the travel time and the demand in the final cost, with $\delta + \eta$=1. Note that if $\eta = 0$, the granular cost is reduced to a travel time based criterion. The idea of the hybrid criterion is based on the fact that the similarity between the customers requirements affects the handling sub-problem. Finally, the factor 2 is used in the second denominator since there are two demands considered (that is, two customers i and j).

The details of the main algorithm are presented in Algorithm 8. A feasible initial solution (line 1) generated with the well known LKH-3 heuristic [76] is used as starting point of the GILS. The obtained solution is improved by using a initial local search procedure (line 2), which is exclusively applied after the initial solution. Furthermore, the variables related to the acceptance criteria, expansion factor, and tabu solutions are initialized. The main cycle follows the original ILS, with a perturbation (line 12), and a local search (line 13). Then, the GILS updates the solution variables according to three criteria:

- If the solution found ($LSRoute$) is better than the global solution ($bestRoute$), the algorithm is restarted (i.e. all the parameters are updated), and the current solution ($currentCost$) is updated.
- If the solution found ($LSRoute$) is only better than the current solution ($currentCost$) or the solution found ($LSRoute$) fulfills the acceptance criteria (section 2.3.6), the current solution is updated.

The next step is to check if it is necessary to increase the granular factor ρ (lines 34-42). This factor is an input parameter for the granular local search and is defined as the neighbor ratio for the search. This check is applied through the variable *reset*. If the algorithm reaches *reset* iterations, ρ is increased to $\rho + \gamma$ (with γ constant throughout the search), the tabu list is cleaned and the variable *reset* is updated in order to increase again in the following iterations of the main cycle. When $\rho = 1$, the GILS algorithm runs without granularity, using all neighbors with $Niter$ iterations. Finally, the temperature and the iteration counter are updated.

The granular cost is used to sort the edges, and the ρ% cheapest ones are evaluated. The parameter ρ is initialized to 50%, nevertheless, if the algorithm remains in a local optimum for *reset* iterations, ρ is augmented to $\rho + \gamma$.

Algorithm 8 GILS Main Scheme

```
$(initialRoute, initialCost) = LKH3()$
$(bestRoute, bestCost) = GreedyLocalSearch(initialRoute, initialCost)$
$\rho = \rho_{initial}$
$sReset = N/ratioReset$
$reset = sReset$
$currentRoute = bestRoute$
$currentCost = bestCost$
$temp = sTemp$
$TabuSolutions = empty$
$it = 0$
while $it < NIter$ do
    $(perturbationRoute, perturbationCost) = Perturbation(currentRoute, currentCost, TabuSolutions)$
    $(LSRoute, LSCost) = LocalSearch(perturbationRoute, perturbationCost, \rho)$
    if $LSCost < bestRoute$ then
        $bestRoute = LSRoute$
        $bestCost = LSCost$
        $currentRoute = LSRoute$
        $currentCost = LSCost$
        $TabuSolutions.clear()$
        $UpdateTabuList(bestRoute, TabuSolutions)$
        $temp = sTemp$
        $it = 0$
        $ResetParameter = sResetParameter$
    else
        if $LSCost < currentCost$ then
            $currentRoute = LSRoute$
            $currentCost = LSCost$
        else
            if $AcceptanceCriteria(temp, LSRoute, LSCost)$ then
                $currentRoute = LSRoute$
                $currentCost = LSCost$
            end if
        end if
        if $reset == it$ then
            if $\rho + \gamma < 1.0$ then
                $\rho = \rho + \gamma$
                $TabuSolutions.clear()$
                $reset = reset + sReset$
            else
                $reset = NIter$
            end if
        end if
        $it = it + 1$
        $temp = max(temp * alpha, eTemp)$
    end if
end while
```

2.4.2 Neighborhoods, perturbation and local search

This subsection details the operators used as neighborhoods of a certain solution, the perturbation procedure used to escape from local optima, and the local search process.

Neighborhoods

The neighborhoods used in GILS are the same ones defined in chapter 2.3.4 and also defined in [67]. For this approach, each neighborhood works with the expansion factor (only in local search procedures) ρ, previously defined. This parameter defines the number of customers (within the neighborhoods) that can be used (i.e. $n * \rho$).

Perturbation

The perturbation algorithm is based on the method proposed in [67] (see Algorithm 9). It selects randomly a neighborhood, and a random move of the neighborhood is applied if it is feasible and is not contained in the *TabuSolutions* list. If one of the two mentioned conditions does not hold the whole process is repeated. That is, the neighborhood and the move are randomly selected again until the conditions are fulfilled. Once the perturbation is accepted, the tabu list and the objective function are updated. The perturbation process described is less-aggressive than that used in [67] since only feasible moves are accepted.

Using a tabu list within the perturbation avoids the redundant execution of the local search procedure for routes previously considered, with a reduction of the computation time to obtain the optimal value of the handling cost. For this reason, the output solution of this perturbation is a new route and avoids delivering to the local search process a solution that has been visited.

Algorithm 9 Perturbation for GILS

```
Input: $currentRoute, currentCost, TabuSolutions$
Output: $TabuSolutionsperturbationRoute, perturbationCost$
$noPerturbationFound = true$
$perturbationRoute = empty$
while ($noPerturbationFound$) do
    $perturbationRoute = currentRoute$
    $ipr = 0$
    $jpr = 0$
    while $ipr == jpr$ do
        $ipr = RandomIntegerNumber(0, N)$
        $jpr = RandomIntegerNumber(0, N)$
    end while
    $selectNeghboord = RandomIntegerNumber(1, 3)$
    if $selectNeghboord == 1$ then
        $perturbationRoute = SingleRelocate(currentRoute, ipr, jpr)$
    end if
    if $selectNeghboord == 2$ then
        $perturbationRoute = SingleSwap(currentRoute, ipr, jpr)$
    end if
    if $selectNeghboord == 3$ then
        $perturbationRoute = SingleTwoOpt(currentRoute, ipr, jpr)$
    end if
    if $NoTabu(perturbationRoute, TabuSolutions)$ and $isFeasible(perturbationRoute)$ then
        $noPerturbationFound = false$
    end if
end while
$UpdateTabuList(perturbationRoute, TabuSolutions)$
$perturbationCost = HandlingCost(perturbationRoute) + RouteCost(perturbationRoute)$
return ($TabuSolutions, perturbationRoute, perturbationCost$)
```

Local search

The proposed algorithm uses two local search procedures. The first one (*GreedyBasedLS*) is applied only once, after the construction of the initial solution. The objective of this local search is to improve the solution provided by the LKH-3 algorithm, because this algorithm seeks for minimizing the traveling cost without considering the handling cost. The second algorithm (*LocalSearch*) is used within the ILS, i.e. after the perturbation process. The *GreedyBasedLS* description was proposed in [67] and is described in detail in Algorithm 6.

Algorithm 10 LocalSearch

```
Input: $perturbationRoute, perturbationCost, \rho, N_{max}$
Output: $LSRoute, LSCost$
$iter = 0$
$LSRoute = perturbationRoute$
$LSCost = perturbationCost$
while $iter < N_{max}$ do
    $currentRoute = LSRoute$
    $currentCost = LSCost$
    for $selectNeghboord \in \{1, 2, 3\}$ do
        if $selectNeghboord == 1$ then
            $(RouteN, costRouteN) = TwoOpt(currentRoute, currentCost, \rho)$
        end if
        if $selectNeghboord == 2$ then
            $(RouteN, costRouteN) = Relocate(currentRoute, currentCost, \rho)$
        end if
        if $selectNeghboord == 3$ then
            $(RouteN, costRouteN) = Swap(currentRoute, currentCost, \rho)$
        end if
        if $currentCost > costRoute$ then
            $currentRoute = costRouteN$
            $currentCost = RouteN$
        end if
    end for
    if $isFeasible(currentRoute)$ then
        if $LSCost < currentCost$ then
            $LSRoute = currentRoute$
            $LSCost = currentCost$
            $iter = iter + 1$
        else
            break
        end if
    else
        break
    end if
end while
return $(LSRoute, LSCost)$
```

The *LocalSearch* procedure is detailed in algorithm 10. The input parameters are the following: a feasible route generated by the perturbation process, the cost of that route, ρ, N_{max}, and the maximum number of iterations. The internal cycle executes the three neighborhoods in a sequential way, one per iteration. Then, it verifies the feasibility of the solution, and if it is improved in order to replace the current solution. The change in the current neighborhood is performed regardless of the result in the internal cycle, that is, if the current solution is improving or not. Finally, within the outer cycle the global solution is updated if an improvement is found, otherwise, the cycle ends and the algorithm returns the best solution found and its cost.

2.5 Results

2.5.1 Instances

The computational experiment is performed by considering the instances presented by [63]. This set contains ten instances for each value of n =[20; 40; 60; 80; 100; 120; 140; 160; 180; 200]. Each of these instances was adapted from [77] and the pickups and deliveries were scaled from [42]. For each customer $i \in V_c$, the pickup value p_i is scaled using $p'_i = max\{1, p_i \ mod \ 20\}$. Finally, each value of α_i and β_i for the TSPPD-H is obtained using :

$$\beta_i = \lfloor p'_i(i \ mod \ 5)/5 \rfloor \ \ and \ \ \alpha_i = (p'_i - \beta_i) \tag{2.5}$$

The vehicle capacity is $Q = max\{\sum_{i \in V_c} \alpha i, \sum_{i \in V_c} \beta i\}$. Finally, an analysis in [63] suggests using $h = h_a = h_b$ in a constant way with value $h = 20/|V_c|$.

2.5.2 Parameters and Experiments for ILS-F

TABLE 2.1: Parameters Setting

Parameter	*Values*	*Final Value*
t_0	[50,100,200]	100
N_{iter}	[50,100,200,300]	200
N_{iterL}	[10,20,30,40,50]	10
N_{per}	[50,100,200,300,400]	200

In order to obtain appropriate values of the parameters, different ranges were used, executing ILS-F for the instances with n = 60. Table 2.1 shows a summary of the domain range used for each parameter and the chosen value. The ranges of the values were considered according to the performance of the algorithm, which was executed 5 times for each combination of the parameter values:

- t_0: The value t_0 = 100 was determined by considering the behavior of the algorithm. During different iterations, the allowed temperature and cooling scheduling change the current solution in an effective way inside the ILS-F.
- N_{iter}: If the number of iterations is large ($N_{iter} > 200$), the metaheuristic does not improve the current solution. On the other hand, if N_{iter} is small ($N_{iter} <$ 200), low quality solutions are obtained.
- N_{iterL}: The number of iterations of the local search can generate better solutions with a value of 10. Even the local search finishes the execution earlier when it does not improve a perturbation.
- N_{per}: If the number of iterations N_{per} is large ($N_{per} > 200$), the perturbation generates a neighbor that is too far from the current solution. As a consequence, the improvement procedure does not obtain better results with respect

to the current solution. On the other hand, if the number of iterations is low ($N_{per} < 200$), the perturbation generates a neighbor too near to the current solution. For this reason, the improvement procedure falls in a local optimum.

Metaheuristic ILS-F has been coded in C++ programming language, and the experiments were executed on an Intel Xeon E5-2660 processor of 2.2 Ghz and 8 GB.

2.5.3 Parameters and Experiments for GILS

The granular version of the ILS was calibrated using the Irace package [78]. This package iteratively and elitistically searches for the best parameters using a predefined range (or set) for each GILS parameter. Each iteration of Irace updates the parameters, thus obtaining high quality parameters. Small instances (n=20 and n=40) were used for the calibration, and the values related to the granular formula (i.e. δ and η) and the *ratioReset* parameter were fixed. The details are given in Tables 2.2 and 2.3.

TABLE 2.2: Fixed values for δ, η and *ratioReset*

δ	η	*ratioReset*
0.5	0.5	0.5
0.5	0.5	0.7
0.25	0.75	0.7
0.75	0.25	0.7

TABLE 2.3: Set of parameters from Irace

ID	γ	t_i	t_f	$NIter$	tabutenure
1	0.05	250	1	200	40
2	0.05	100	5	200	40
3	0.1	100	5	200	40
4	0.05	100	1	200	30
5	0.05	100	5	200	30

The final experiments were performed using the four sets of parameters reported in Table 2.2. No combination of parameters in this table generates results dominated for each considered instance, with respect to others, the details of these results are reported in Appendix A. In this way, we executed the algorithm four times, and we used only the best set of parameters from Irace, i.e., the set ID = 1. . The calibration experiments were performed on a multi-thread AMD Ryzen 7 2700X Eight-Core Processor running at 3.7 GHz with 64 GB RAM. Finally, the algorithm was run on the same computer as ILS-F to make a fair comparison. The calibration time was 296 hours.

2.5.4 Results and comparison

Tables 2.4, 2.5, 2.6 and 2.7 contain the computational results obtained by the algorithm Series [63], ALNS [65], ILS-F and GILS for the considered instances.The results of ALNS correspond to those reported in [65], and were obtained on an Intel Xeon Gold 6148 processor of 2.4 GHz with 16 GB (having characteristics similar to those of our computer). The tables report for each instance, the values of the best known solution (column BKS defined as the minimum value found by [63] and [65]) and of the best solution found in [63] (column Series). In addition, for the algorithms ALNS, ILS-F, and GILS the following values are reported:

- The best solution value using dynamic programming (DP column).
- The best solution value using linear heuristics (Heu column).
- The best solution value between DP and Heu (Best columns).
- The sum of the average times (expressed in seconds) required by DP and Heu (Time(s) column).
- The percentage gap between the best-known solution value (BKS) and Best, computed as Gap(%) = 100*((Best - BKS) / BKS).

At the end of each set, the average values of the corresponding columns are also shown. The ALNS results with $|V_c| = \{80, 100\}$ were not presented, as in [63], the reported values were abnormal, and therefore in [65], it was decided to avoid the comparison on these two sets of instances. However, we rectified those values presenting the best known value found by the metaheuristics presented in [63], and made a full comparison with ILS-F and GILS.

About the Iterated Local Search with frequency

Relocate, Swap and 2-Opt heuristics are used in the local search procedure. The two best (with respect to the frequency) heuristics always used and when these heuristics do not improve the current solution, the third heuristic is used. Figure 2.8 shows (for the instance with $n = 120$ and $l = 10$) a chart that contains the frequencies of use of the heuristics (from 0 to 1 on axis f) during 200 iterations (x axis) and a line chart (red) for the temperature (T axis). Our algorithm resets the initial temperature (100°) when a new best solution is found in the search process. When this happens, the frequencies of use are again equivalent $< 1/3, 1/3, 1/3 >$. Between iterations 1 and 28, the prioritized heuristics for the execution are 2-Opt and Relocate. However, when the first new best solution is found, Relocate starts to not enhance solutions in *LSF* and Swap begins to do so. Thus, the frequencies start to change in the following iterations. When a second best solution is found (iteration 73), Relocate has again a higher frequency during the remaining iterations. The studies presented in [63] can be confirmed in this case, as 2-Opt and Relocate are used in most iterations. However, Swap is necessary at some point of the search to find good results.

ALNS obtains better results than ILS-F. Our results are competitive only for small instances. However, for larger instance sets (particularly $|V_c| = \{180, 200\}$), ILS-F is bested by the metaheuristics of [63], and by ALNS. However, it should be taken into

TABLE 2.4: Computational Results for the TSPPD-H with the instances from [63]

N° cus	l	BKS	Series [63]	ALNS [65]					ILS-F [67]					GILS				
				DP	Heu	Best	Time(s)	GAP(%)	DP	Heu	Best	Time(s)	GAP(%)	DP	Heu	Best	Time(s)	GAP(%)
20	1	633.0	633.0	633.0	633.0	633.0	10	0.00	633.0	633.0	633.0	1.29	0.00	633.0	633.0	633.0	7.0	0.00
20	2	584.0	584.0	584.0	584.0	584.0	10	0.00	584.0	584.0	584.0	1.26	0.00	584.0	584.0	584.0	7.0	0.00
20	3	573.0	573.0	573.0	573.0	573.0	10	0.00	573.0	573.0	573.0	1.24	0.00	573.0	573.0	573.0	7.5	0.00
20	4	706.0	706.0	706.0	706.0	706.0	10	0.00	706.0	706.0	706.0	1.35	0.00	706.0	706.0	706.0	7.3	0.00
20	5	501.0	501.0	501.0	501.0	501.0	10	0.00	501.0	501.0	501.0	1.50	0.00	501.0	501.0	501.0	7.1	0.00
20	6	578.0	578.0	578.0	578.0	578.0	10	0.00	578.0	578.0	578.0	1.16	0.00	578.0	578.0	578.0	6.9	0.00
20	7	612.0	612.0	612.0	612.0	612.0	11	0.00	612.0	612.0	612.0	1.37	0.00	612.0	612.0	612.0	6.8	0.00
20	8	567.0	567.0	567.0	567.0	567.0	10	0.00	567.0	567.0	567.0	1.36	0.00	567.0	567.0	567.0	6.8	0.00
20	9	604.0	604.0	604.0	604.0	604.0	21	0.00	604.0	604.0	604.0	1.31	0.00	604.0	604.0	604.0	8.1	0.00
20	10	565.0	574.0	565.0	565.0	565.0	21	0.00	565.0	565.0	565.0	1.29	0.00	565.0	565.0	565.0	11.1	0.00
Average		592.30	593.20	592.30	592.30	592.30	12.30	0.00	592.30	592.30	592.30	1.31	0.00	592.30	592.30	592.30	7.6	0.00
40	1	909.5	909.5	913.5	909.5	909.5	79	0.00	909.5	909.5	909.5	12.42	0.00	909.5	909.5	909.5	91.7	0.00
40	2	883.0	885.0	883.0	883.0	883.0	81	0.00	883.0	883.0	883.0	13.76	0.00	883.0	883.0	883.0	74.2	0.00
40	3	815.5	815.5	815.5	815.5	815.5	79	0.00	815.5	815.5	815.5	12.51	0.00	815.5	815.5	815.5	91.0	0.00
40	4	898.0	898.0	898.0	898.0	898.0	83	0.00	898.0	898.0	898.0	12.77	0.00	898.0	898.0	898.0	71.0	0.00
40	5	743.5	743.5	745.0	743.5	743.5	81	0.00	743.5	743.5	743.5	12.36	0.00	743.5	743.5	743.5	82.6	0.00
40	6	883.5	901.0	883.5	883.5	883.5	82	0.00	883.5	883.5	883.5	13.39	0.00	883.5	883.5	883.5	89.9	0.00
40	7	798.5	798.5	798.5	798.5	798.5	83	0.00	798.5	798.5	798.5	8.06	0.00	798.5	798.5	798.5	96.2	0.00
40	8	795.0	795.0	795.0	795.0	795.0	82	0.00	795.0	795.0	795.0	13.48	0.00	795.0	795.0	795.0	80.7	0.00
40	9	876.5	876.5	876.5	876.5	876.5	82	0.00	876.5	876.5	876.5	12.30	0.00	876.5	876.5	876.5	79.4	0.00
40	10	862.5	866.0	862.5	862.5	862.5	81	0.00	862.5	862.5	862.5	13.36	0.00	862.5	862.5	862.5	82.3	0.00
Average		846.55	848.85	847.10	846.55	846.55	81.30	0.00	846.55	846.55	846.55	12.44	0.00	846.55	846.55	846.55	83.9	0.00
60	1	1051.0	1060.1	1051.0	1051.0	1051.0	316	0.00	1051.0	1056.7	1051.0	48.43	0.00	1051.0	1051.0	1051.0	232.5	0.00
60	2	1044.3	1051.1	1047.3	1044.3	1044.3	321	0.00	1047.3	1047.3	1047.3	52.41	0.29	1044.3	1046.7	1044.3	283.0	0.00
60	3	990.4	**990.4**	993.7	993.7	993.7	315	0.33	993.7	993.7	993.7	46.33	0.33	993.7	993.7	993.7	262.3	0.33
60	4	1061.5	**1061.5**	1066.0	1066.0	1066.0	322	0.43	1066.0	1066.0	1066.0	51.33	0.43	1066.0	1066.0	1066.0	259.6	0.43
60	5	986.9	**986.9**	989.7	989.7	989.7	322	0.28	989.7	989.7	989.7	17.01	0.28	989.7	989.7	989.7	225.0	0.28
60	6	1067.7	1086.3	1067.7	1073.3	1067.7	323	0.00	1080.7	1083.3	1080.7	55.58	1.21	1067.7	1074.3	1067.7	424.5	0.00
60	7	1005.4	**1005.4**	1007.3	1007.3	1007.3	315	0.19	1007.3	1007.3	1007.3	55.74	0.19	1007.3	1007.3	1007.3	310.1	0.19
60	8	1027.2	**1027.2**	1031.0	1031.0	1031.0	318	0.37	1031.0	1031.0	1031.0	50.59	0.37	1031.0	1031.0	1031.0	339.0	0.37
60	9	1001.4	**1001.4**	1004.3	1004.3	1004.3	318	0.29	1004.3	1004.3	1004.3	47.93	0.29	1004.3	1004.3	1004.3	296.5	0.29
60	10	1048.7	1062.1	1048.7	1048.7	1048.7	328	0.00	1053.3	1048.7	1048.7	48.17	0.00	1048.7	1048.7	1048.7	329.9	0.00
Average		1028.45	1033.25	1030.67	1030.93	1030.37	319.80	0.19	1032.43	1032.80	1031.97	47.35	0.34	1030.37	1031.27	1030.37	296.2	0.19

TABLE 2.5: Computational Results for the TSPPD-H with the instances from [63]

N° cus	l	BKS	Series [63]	ALNS [65]					ILS-F [67]					GILS				
				DP	Heu	Best	Time(s)	GAP(%)	DP	Heu	Best	Time(s)	GAP(%)	DP	Heu	Best	Time(s)	GAP(%)
80	1	1207.3	1207.3						1193.0	1193.3	1193.0	163.58	-1.18	1192.8	1192.3	**1192.3**	1057.7	**-1.24**
80	2	1191.5	1191.5						1186.0	1191.3	1186.0	165.06	-0.46	1186.0	1186.0	1186.0	1030.2	-0.46
80	3	1170.8	1170.8						1170.8	1170.8	1170.8	153.51	0.00	1170.8	1170.8	1170.8	747.6	0.00
80	4	1279.5	1279.5						1279.5	1278.3	1278.3	154.37	-0.10	1274.0	1275.5	**1274.0**	863.7	**-0.43**
80	5	1228.8	1228.8						1225.3	1220.0	1220.0	164.31	-0.71	1212.3	1211.5	**1211.5**	1269.2	**-1.40**
80	6	1278.0	1278.0						1273.3	1277.5	1273.3	172.67	-0.37	1262.3	1264.8	**1262.3**	1234.8	**-1.23**
80	7	1170.8	1170.8						1171.5	1172.5	1171.5	161.96	0.06	1170.0	1168.8	**1168.8**	846.8	**-0.17**
80	8	1220.0	1220.0						1220.0	1226.5	1220.0	158.34	0.00	1220.0	1220.5	1220.0	964.1	0.00
80	9	1205.0	1205.0						1198.0	1201.0	1198.0	153.72	-0.58	1194.3	1194.3	**1194.3**	959.0	**-0.89**
80	10	1197.3	1197.3						1198.8	1198.3	1198.3	155.61	0.08	1197.3	1197.8	1197.3	916.2	0.00
Average		1214.88	1214.88						1211.60	1212.93	1210.90	160.31	-0.33	1207.95	1208.20	**1207.70**	988.9	**-0.59**
100	1	1316.4	1316.4						1318.8	1326.0	1318.8	300.03	0.18	1310.6	1310.0	**1310.0**	2212.6	**-0.49**
100	2	1354.4	1354.4						1344.8	1346.4	1344.8	314.00	-0.71	1329.8	1335.8	**1329.8**	2048.4	**-1.82**
100	3	1370.4	1370.4						1340.6	1347.0	1340.6	328.58	-2.17	1326.0	1332.4	**1326.0**	2033.4	**-3.24**
100	4	1439.0	1439.0						1413.6	1428.2	1413.6	317.90	-1.77	1410.0	1411.6	**1410.0**	2377.3	**-2.02**
100	5	1312.6	1312.6						1326.6	1333.2	1326.6	321.77	1.07	1312.6	1315.4	1312.6	2141.7	0.00
100	6	1425.6	1425.6						1417.4	1418.2	1417.4	395.20	-0.58	1405.6	1408.4	**1405.6**	2644.8	**-1.40**
100	7	1358.8	1358.8						1352.2	1354.6	1352.2	349.58	-0.49	1337.8	1337.8	**1337.8**	2053.5	**-1.55**
100	8	1408.0	1408.0						1409.2	1401.4	1401.4	405.61	-0.47	1392.2	1392.2	**1392.2**	2090.3	**-1.12**
100	9	1381.8	1381.8						1378.2	1372.0	1372.0	367.19	-0.71	1372.0	1372.0	1372.0	1811.6	-0.71
100	10	1414.6	1414.6						1402.6	1404.6	1402.6	368.66	-0.85	1401.4	1401.6	**1401.4**	2295.6	**-0.93**
Average		1378.16	1378.16						1370.40	1373.16	1369.00	346.85	-0.65	1359.80	1361.72	**1359.74**	2170.9	**-1.34**
120	1	1436.7	1472.5	1442.6	1436.7	1436.7	3878	0.00	1462.3	1439.2	1439.2	698.90	0.17	1438.0	1435.7	**1435.7**	3382.5	**-0.07**
120	2	1455.5	1482.3	1459.6	1455.5	1455.5	3829	0.00	1476.5	1461.3	1461.3	682.94	0.40	1451.2	1453.0	**1451.2**	3877.4	**-0.30**
120	3	1465.5	1510.1	1483.9	1465.5	1465.5	3877	0.00	1480.8	1474.2	1474.2	588.37	0.59	1463.5	1463.5	**1463.5**	3086.7	**-0.14**
120	4	1543.7	1563.5	1549.5	1543.7	1543.7	3911	0.00	1546.0	1538.7	1538.7	630.35	-0.33	1525.7	1531.2	**1525.7**	5001.9	**-1.17**
120	5	1430.7	1457.1	1437.5	1430.7	1430.7	3896	0.00	1443.2	1442.8	1442.8	598.78	0.85	1425.2	1427.5	**1425.2**	3134.5	**-0.39**
120	6	1528.0	1546.3	1540.7	1528.0	1528.0	3964	0.00	1552.3	1538.8	1538.8	558.17	0.71	1525.7	1528.2	**1525.7**	4163.7	**-0.15**
120	7	1517.7	1557.3	1520.0	1517.7	1517.7	3981	0.00	1527.3	1527.0	1527.0	692.57	0.61	1510.7	1517.3	**1510.7**	3916.6	**-0.46**
120	8	1504.8	1524.1	1517.2	1504.8	**1504.8**	3963	**0.00**	1513.3	1518.5	1513.3	658.94	0.57	1507.3	1508.8	1507.3	3946.3	0.17
120	9	1528.9	1548.0	1528.9	1530.7	1528.9	4016	0.00	1546.0	1532.0	1532.0	591.64	0.20	1528.7	1521.0	**1521.0**	3934.9	**-0.52**
120	10	1540.8	1573.9	1548.7	1540.8	1540.8	4009	0.00	1532.0	1532.0	1532.0	653.78	-0.57	1531.8	1532.5	**1531.8**	3145.0	**-0.58**
Average		1495.23	1523.48	1502.86	1495.41	1495.23	3932.40	0.00	1507.98	1500.45	1499.93	635.44	0.32	1490.77	1491.87	**1489.77**	3758.9	**-0.37**

TABLE 2.6: Computational Results for the TSPPD-H with the instances from [63]

N° cus	l	BKS	Series [63]	ALNS [65]					ILS-F [67]					GILS				
				DP	Heu	Best	Time(s)	GAP(%)	DP	Heu	Best	Time(s)	GAP(%)	DP	Heu	Best	Time(s)	GAP(%)
140	1	1575.2	**1575.2**	1586.6	1589.0	1586.6	7285	0.72	1591.4	1600.1	1591.4	954.36	1.03	1580.3	1585.1	1580.3	6255.2	0.32
140	2	1584.9	1605.8	1596.8	1584.9	**1584.9**	7439	**0.00**	1597.3	1593.3	1593.3	1064.24	0.53	1599.9	1593.6	1593.6	8156.1	0.55
140	3	1571.3	1583.7	1572.8	1571.3	1571.3	7388	0.00	1583.3	1570.4	1570.4	1047.58	-0.06	1567.7	1570.9	**1567.7**	8125.4	**-0.23**
140	4	1682.7	1712.7	1694.1	1682.7	**1682.7**	7355	**0.00**	1692.0	1709.6	1692.0	1087.76	0.55	1688.7	1685.0	1685.0	9023.4	0.14
140	5	1542.9	1547.4	1542.9	1549.1	1542.9	7333	0.00	1558.3	1559.7	1558.3	1075.18	1.00	1541.1	1547.4	**1541.1**	7947.4	**-0.11**
140	6	1631.3	1662.3	1659.3	1631.3	**1631.3**	8361	**0.00**	1658.3	1662.0	1658.3	1106.05	1.65	1636.1	1633.9	1633.9	8601.1	0.16
140	7	1621.4	1664.4	1633.9	1621.4	1621.4	8367	0.00	1655.4	1637.0	1637.0	986.82	0.96	1629.6	1619.4	**1619.4**	7688.8	**-0.12**
140	8	1611.0	1646.1	1628.7	1611.0	**1611.0**	7432	**0.00**	1631.0	1612.1	1612.1	940.44	0.07	1620.0	1618.1	1618.1	7558.4	0.44
140	9	1629.1	**1629.1**	1643.5	1645.3	1643.5	7569	0.88	1647.1	1646.7	1646.7	1137.83	1.08	1645.0	1640.1	1640.1	7373.9	0.68
140	10	1671.9	1693.8	1683.1	1671.9	1671.9	7455	0.00	1689.3	1687.4	1687.4	1152.70	0.93	1665.3	1671.4	**1665.3**	7533.5	**-0.40**
Average		1612.17	1632.04	1624.17	1615.79	1614.75	7598.40	0.16	1630.35	1627.84	1624.70	1055.29	0.78	1617.37	1616.50	**1614.46**	7826.3	**0.14**
160	1	1712.6	1741.8	1713.2	1712.6	1712.6	12354	0.00	1728.4	1718.5	1718.5	1515.11	0.34	1712.5	1705.9	**1705.9**	11506.0	**-0.39**
160	2	1725.7	1773.6	1730.3	1725.7	**1725.7**	12544	**0.00**	1754.8	1745.8	1745.8	1662.23	1.16	1735.9	1731.4	1731.4	10164.3	0.33
160	3	1684.3	1690.9	1684.3	1689.7	**1684.3**	12220	**0.00**	1724.9	1727.1	1724.9	1300.47	2.41	1697.1	1710.4	1697.1	9656.3	0.76
160	4	1837.5	1858.1	1842.8	1837.5	1837.5	12486	0.00	1857.8	1829.5	1829.5	1277.91	-0.44	1827.3	1816.1	**1816.1**	15900.0	**-1.16**
160	5	1654.1	1667.8	1654.1	1658.0	1654.1	12424	0.00	1679.5	1647.8	1647.8	1455.22	-0.38	1637.0	1645.0	**1637.0**	10052.3	**-1.03**
160	6	1726.7	1813.0	1738.6	1726.7	**1726.7**	12646	**0.00**	1788.9	1783.8	1783.8	1557.13	3.30	1763.4	1751.9	1751.9	10944.8	1.46
160	7	1742.4	1774.3	1753.7	1742.4	**1742.4**	12585	**0.00**	1762.9	1764.5	1762.9	1074.67	1.18	1768.9	1753.6	1753.6	8625.2	0.64
160	8	1749.2	1770.8	1762.4	1749.2	**1749.2**	12459	**0.00**	1779.4	1789.6	1779.4	1260.07	1.73	1763.5	1763.4	1763.4	12879.5	0.81
160	9	1786.7	1800.9	1786.7	1793.2	1786.7	12724	0.00	1794.3	1796.1	1794.3	1220.66	0.42	1779.0	1782.5	**1779.0**	9775.0	**-0.43**
160	10	1756.3	1764.4	1756.3	1757.6	**1756.3**	12578	**0.00**	1757.4	1758.3	1757.4	1562.13	0.06	1757.4	1757.4	1757.4	7375.7	0.06
Average		1737.55	1765.54	1742.24	1739.26	**1737.55**	12502.00	**0.00**	1762.80	1756.09	1754.40	1388.56	0.98	1744.19	1741.75	1739.28	10687.9	0.10
180	1	1818.9	1854.2	1833.8	1818.9	1818.9	19122	0.00	1871.8	1842.3	1842.3	2703.05	1.29	1815.6	1818.1	**1815.6**	19539.9	**-0.18**
180	2	1838.9	1863.3	1851.4	1838.9	**1838.9**	19361	**0.00**	1873.8	1866.2	1866.2	2239.02	1.49	1847.2	1849.9	1847.2	12244.1	0.45
180	3	1822.1	1858.4	1822.1	1830.9	**1822.1**	19330	**0.00**	1859.8	1840.4	1840.4	1861.19	1.01	1829.7	1824.3	1824.3	17783.3	0.12
180	4	1942.8	1988.2	1943.9	1942.8	1942.8	19614	0.00	1996.9	1952.3	1952.3	1867.83	0.49	1955.2	1941.7	**1941.7**	22588.9	**-0.06**
180	5	1785.2	1795.8	1785.2	1785.4	**1785.2**	19506	**0.00**	1818.0	1818.2	1818.0	2040.62	1.84	1796.0	1793.1	1793.1	15695.4	0.44
180	6	1817.5	**1817.5**	1826.4	1820.4	1820.4	19451	0.16	1847.7	1842.1	1842.1	2445.59	1.36	1831.1	1829.1	1829.1	14847.6	0.64
180	7	1846.8	1868.0	1846.8	1848.3	1846.8	19552	0.00	1888.9	1868.9	1868.9	332.81	1.20	1831.8	1847.3	**1831.8**	16495.4	**-0.81**
180	8	1862.2	1883.4	1865.3	1862.2	**1862.2**	19493	**0.00**	1874.0	1909.7	1874.0	1667.39	0.63	1878.9	1875.9	1875.9	13818.8	0.74
180	9	1917.5	1931.4	1917.5	1935.0	**1917.5**	19754	**0.00**	1942.0	1942.2	1942.0	1694.72	1.28	1929.4	1930.7	1929.4	14477.7	0.62
180	10	1845.9	1852.5	1857.6	1845.9	**1845.9**	19464	**0.00**	1875.3	1882.8	1875.3	2150.64	1.59	1853.9	1857.1	1853.9	12320.5	0.43
Average		1849.78	1871.28	1855.00	1852.87	**1850.07**	19464.70	**0.02**	1884.81	1876.52	1872.17	1900.29	1.22	1856.88	1856.72	1854.20	15981.2	0.24

TABLE 2.7: Computational Results for the TSPPD-H with the instances from [63]

N° cus	I	BKS	Series [63]	ALNS [65]					ILS-F [67]					GILS				
				DP	Heu	Best	Time(s)	GAP(%)	DP	Heu	Best	Time(s)	GAP(%)	DP	Heu	Best	Time(s)	GAP(%)
200	1	1928.6	1976.3	1934.4	1928.6	**1928.6**	28988	**0.00**	1985.4	1977.9	1977.9	1955.05	2.56	1940.6	1957.1	1940.6	22595.1	0.62
200	2	1964.7	1982.0	1964.7	1975.2	**1964.7**	28869	**0.00**	2043.6	2024.7	2024.7	1118.04	3.05	1979.5	1973.8	1973.8	25238.2	0.46
200	3	1954.3	1976.7	1954.3	1958.8	**1954.3**	28568	**0.00**	1997.3	1983.6	1983.6	1813.00	1.50	1979.9	1968.4	1968.4	19022.7	0.72
200	4	2041.4	2119.5	2041.4	2061.5	**2041.4**	29237	**0.00**	2161.2	2068.7	2068.7	1244.31	1.34	2044.2	2052.7	2044.2	28940.0	0.14
200	5	1890.4	1905.6	1890.4	1902.1	1890.4	29101	0.00	1888.0	1915.1	**1888.0**	1364.15	**-0.13**	1888.8	1898.9	1888.8	20415.9	-0.08
200	6	1948.3	2011.1	1948.3	1957.6	**1948.3**	29309	**0.00**	2005.1	2003.5	2003.5	1875.17	2.83	1979.4	1973.1	1973.1	20941.4	1.27
200	7	1930.4	1983.0	1931.9	1930.4	**1930.4**	29248	**0.00**	1998.9	1977.7	1977.7	1250.51	2.45	1941.5	1937.4	1937.4	23937.4	0.36
200	8	1995.1	2000.3	1995.1	1995.8	**1995.1**	28777	**0.00**	2054.9	2011.8	2011.8	1745.23	0.84	2003.5	1997.4	1997.4	24317.2	0.12
200	9	2013.7	2052.7	2016.3	2013.7	**2013.7**	29607	**0.00**	2093.9	2043.6	2043.6	1449.19	1.48	2047.5	2043.8	2043.8	18442.5	1.49
200	10	1927.3	1977.9	1933.0	1927.3	**1927.3**	29265	**0.00**	1957.6	1957.7	1957.6	2116.96	1.57	1944.7	1939.4	1939.4	14123.4	0.63
Average		1959.42	1998.51	1960.98	1965.10	**1959.42**	29096.90	**0.00**	2018.59	1996.43	1993.71	1593.16	1.75	1974.96	1974.20	1970.69	21797.4	0.58
Final Avg1		1371.45	1385.92						1385.78	1381.51	1379.56	714.10	0.44	1372.11	1372.11	**1370.50**	6359.9	**-0.10**
Final Avg2		1390.18	1408.27	1394.42	1392.28	**1390.78**	9125.98	**0.05**	1409.48	1403.62	1401.97	829.23	0.67	1394.17	1393.89	1392.20	7554.9	0.11

Final Avg1 considers all instances
Final Avg2 does not consider instances with n=80 and n=100

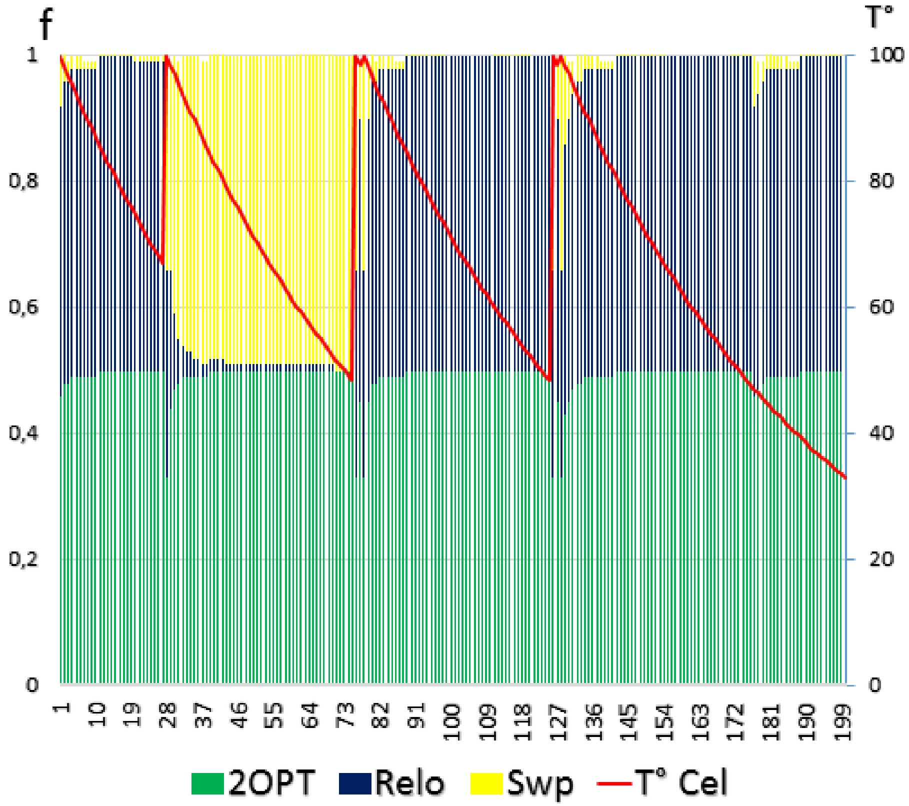


FIGURE 2.8: Frequency of use (200 iterations, instance with n=120 and l=10)

account that the results presented in [63], are obtained by executing three different metaheuristics, thus the computing time are even larger than that of ALNS. Also, considering the global average, the $|V_c| = 200$ set is the only one to present a percentage gap close to 1% (with respect to [63]). Therefore, our results are close to those shown by the previously proposed studies algorithms.

ILS-F is significantly faster than ALNS. As a matter of fact, it is faster on each instance. For $n = 200$, ALNS takes, on average, 7.5 hours against 27 minutes for ILS-F. This may have two explanations: first, the ALNS approach used a depth intensification process applying the three elementary heuristics proposed in [65], while we bet on the study presented in [63], and considered always to use two of the three heuristics, only using a third one if no enhancement could be found with the other two heuristics; second, our DP approach allows us to avoid redundant operations in the computation of the handling costs.

About the Granular Iterated Local Search

There are no significant differences between ALNS and GILS for the small instances. The value of %gap for ALNS and GILS is identical in Table 2.4, and the execution times are similar. Of course, the GILS was run 4 times, so the average time for a single run is considerably smaller than that of ALNS in this case. Another strange behavior is that in 6 instances for $n = 60$, ALNS, ILS-F, and GILS do not obtain results better than those found by the metaheuristics in [63]. Still, the best results among them are the same (for example, for $l = 9$, the metaheuristics from [63] get 1001.4, while the other metaheuristics find 1004.3).

For medium-size instances, the granular approach gets the best results in the literature (Table 2.5). Considering $n = 80$ and $n = 100$, GILS gets better (or equal) results than those found by the metaheuristics in [63] and by the ILS-F approach. Thus, the new best known solution values for these set are updated. For $n = 120$, algorithm GILS finds 9 new best known solutions compared to the algorithms in the literature.

For the large instances, ALNS is globally better than GILS (Tables 2.6 and 2.7). While GILS finds new best known solutions for 12 (over 40) instances, on average, ALNS appears to be better (although not significantly) than GILS. However, the run times of ALNS remain high with respect to GILS. As the number of customers increases, GILS loses quality compared to ALNS. For $n = 200$, GILS finds only one best known solution, and the worst value of %gap becomes 1.49%.

2.6 Conclusion and future directions

This study proposes a new Iterated Local Search algorithm (called ILS-F) for the solution of TSPPD-H based on the frequency of use during the iterations and a granular version (called GILS). We have improved the algorithm proposed by [63] for computation of the handling cost during the execution of the local search methods so as to avoid redundant computational processes of the dynamic programming approach. ILS-F uses a roulette wheel-type perturbation based on a queue-data history structure that allows us to generate new neighbors. Also, a local search using frequency, and an acceptance criterion based on Simulated Annealing were proposed. ILS-F seems to be competitive with the recently proposed algorithms, resulting 20 times faster than ALNS algorithm [65]. GILS was executed with four different groups of parameters concerning the proposed granular formula. The formula considers the travel time between the customers and also the pickup and delivery units to be handled at each customer. The granular approach finds several new best known solutions for the instances of the literature. Its performance for small and medium-sized instances is better than that of the state-of-the-art metaheuristics, but it loses quality for large-sized instances.

New experiments will be considered for the travel time granularity and a new calibration process for the large instances will be analyzed. In addition, a new problem based on Handling cost could be defined with Draft Limits or Time Windows.

Chapter 3

Formulations, Relaxations and Heuristics for the QMKP

3.1 Introduction

The (linear) *Multiple Knapsack Problem* (MKP) has been intensively studied in the last 40 years (see the relative chapters in the monographs by [79] and [50]).

The MKP is defined on n *items* and m *knapsacks*. Each knapsack $k \in M = \{1, \dots, m\}$ has a *capacity* C_k. Each item $i \in N = \{1, \dots, n\}$ has a *profit* p_i and a *weight* w_i. The objective is to select m disjoint subsets of items to be assigned to the knapsacks so that the total weight assigned to each knapsack does not exceed its capacity and the total profit of the selected items is maximized. By introducing nm binary variables x_{ik} $(i \in N, k \in M)$ taking the value 1 if and only if item i is assigned to knapsack k, the problem is formally defined by the *0-1 Linear Program*

$$\max \quad \sum_{i=1}^{n} \sum_{k=1}^{m} p_i x_{ik} \tag{3.1}$$

$$\text{s.t.} \quad \sum_{i=1}^{n} w_i x_{ik} \le C_k \qquad (k \in M) \tag{3.2}$$

$$\sum_{k=1}^{m} x_{ik} \le 1 \qquad (i \in N) \tag{3.3}$$

$$x \in \{0,1\}^{n \times m}, \tag{3.4}$$

where (3.2) and (3.3) are the classical *capacity* and *cardinality constraints*, respectively. The problem is a generalization of the famous *0-1 Knapsack Problem* (KP), in which $m = 1$. While the KP is ordinary $\mathcal{NP}$-hard and admits pseudo-polynomial time dynamic programming algorithms, the MKP is known to be strongly $\mathcal{NP}$-hard, as it can be seen by transformation from 3-partition (see, e.g., [79]). Strongly $\mathcal{NP}$-hard is a complexity class of decision problems which are still NP-hard even when all numbers in the input are bounded by some polynomial in the length of the input [80].

Although a problem with a similar flavor had been considered by [81] in 1975, to the best of our knowledge, the first *quadratic* version of a knapsack problem was introduced by [82]. In the (single) *Quadratic Knapsack Problem* (QKP) one is given a knapsack with capacity C and n items. Each item $i \in N$ has a profit p_i and a weight w_i. In addition, each pair of distinct items i, j gives a profit p_{ij} if both belong to the solution. (It is assumed that $p_{ji} = p_{ij}$.) The objective is to select a subset of items so that the total weight does not exceed the capacity, and the total profit (sum of the profits of the selected items and of their pairwise profits) is maximized. Formally,

$$\max \quad \sum_{i=1}^{n} p_i x_i + \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} p_{ij} x_i x_j \tag{3.5}$$

$$\text{s.t.} \quad \sum_{i=1}^{n} w_i x_i \leq C \tag{3.6}$$

$$x \in \{0,1\}^n, \tag{3.7}$$

where x_i is a binary variable taking the value 1 if and only if item i is selected. We refer the reader to monograph ([50], Chapter 12) for an extensive treatment of the QKP until 2003, and to [83], [84], [85], [86], [87], and [88] for later studies.

The *Quadratic Multiple Knapsack Problem* (QMKP), to which this study is devoted, was first introduced by [89], and ideally combines the objective function of the QKP and the constraints of the MKP. We have n items and m knapsacks. Each knapsack $k \in M$ has a capacity $C_k \in Z_+$, each item $i \in N$ has a profit $p_i \in Z_+$ and a weight $w_i \in Z_+$. Each pair of distinct items i, j produces a profit $p_{ij} \in Z_+$ (with $p_{ji} = p_{ij}$) if both are assigned to the same knapsack. The objective is to select m disjoint subsets of items to be assigned to the knapsacks, so that the total weight assigned to each knapsack does not exceed its capacity, and the total profit (sum of the profits of the selected items and of the pairwise profits of items assigned to the same knapsack) is maximized. Formally,

$$\max \quad \sum_{i=1}^{n} \sum_{k=1}^{m} p_i x_{ik} + \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \sum_{k=1}^{m} p_{ij} x_{ik} x_{jk} \tag{3.8}$$

$$\text{s.t.} \quad \sum_{i=1}^{n} w_i x_{ik} \leq C_k \qquad (k \in M) \tag{3.9}$$

$$\sum_{k=1}^{m} x_{ik} \leq 1 \qquad (i \in N) \tag{3.10}$$

$$x \in \{0,1\}^{n \times m}, \tag{3.11}$$

where x is defined as for the MKP. As the QKP is the special case of the QMKP arising when $m = 1$, the QMKP is strongly $\mathcal{NP}$-hard. In addition, all computational experiments reported so far in the literature indicate that it is extremely challenging to solve in practice.

Owing to its many practical applications, that range from project management to capital budgeting to product-distribution system design, as well as to its mathematical structure borrowing from well-studied combinatorial problems, the QMKP has received increasing attention in the literature over the last fifteen years. In their seminal work, [89] presented the first 60 benchmark instances and three heuristics. Their paper started a stream of research based on *meta-heuristic* techniques, that includes a genetic algorithm by [90]; an artificial bee colony algorithm by [91]; and a memetic algorithm by [92]. More recently, [93, 94] presented a strategic oscillation algorithm and a Tabu-enhanced iterated greedy approach. In [95] and [96] used, respectively, an iterative response threshold search algorithm, and an evolutionary path relinking approach, for which recent variations have been proposed by [97] and [98]. Other recently proposed methods are presented in [99] and [100].

Despite this growing stream of research on heuristics, no exact method for the QMKP was proposed in the literature until the recent contribution by [101], who presented the first exact solution approach to the QMKP, that uses a formulation based on an exponential-size number of variables, solved via a Branch-and-Price algorithm.

While the literature has been so far concentrating on exponential-size formulations and meta-heuristic approaches, our contribution consists of investigating several polynomial-size formulations, aiming at devising the relaxations that produce good upper bounds in reasonable computing times. In particular, our goal is to compare the effectiveness of the Lagrangian relaxation when applied to the quadratic formulation (3.8)-(3.11) and to a Level 1 Reformulation Linearization, that leads to a decomposable structure. We present the results of computational experiments on a large set of benchmark instances.

The chapter has the following structure. In Section 3.2, we derive several linear models for the QMKP, obtained from classical reformulations of 0-1 quadratic programs. Some theoretical properties and dominances among the resulting formulations are outlined. The surrogate relaxation of the quadratic model is discussed in Section 3.3. Section 3.4 is concerned with the Lagrangian relaxation of the quadratic model (3.8)-(3.11) and of a linear reformulation leading to a set of independent, well-structured subproblems.. Section 3.5 describe a Multi-Start Iterated Local Search. Section 3.6 present a Matheuristic approach for the problem.Section 3.7 presents the computational results and Section 3.8 contains some concluding remarks.

3.2 Linear Formulations

In this section we show how some *linear reformulations* for *0-1 Quadratic Programming* problems with linear constraints (01QP) can be specialized for the QMKP.

3.2.1 Classical Linear Formulations

In [102] proved that any integer-valued algebraic function can be transformed into a linear function by introducing auxiliary binary variables and linear *linking constraints*. In 1974 the idea was independently re-discovered and developed by [103] for 01QP. A direct application to the QMKP would result in *4-index* variables, each representing the product $x_{ik}x_{j\ell}$ for $i, j \in N$ and $k, \ell \in M$. We can observe, however, that our objective function (3.8) only includes products involving the same knapsack index, so it is sufficient to introduce *3-index* binary variables $\hat{y}_{ijk}$, taking the value one if and only if items i and j are assigned to the same knapsack k:

$$\hat{y}_{ijk} = x_{ik}x_{jk} \text{ for } i \in N \setminus \{n\}, j \in N \ (j > i), k \in M. \tag{3.12}$$

The *Fortet-Glover-Woolsey* (FGW) formulation for the QMKP is

$$\begin{aligned}
\text{(FGW)} \quad \max \quad & \sum_{i=1}^{n}\sum_{k=1}^{m} p_i x_{ik} + \sum_{i=1}^{n-1}\sum_{j=i+1}^{n}\sum_{k=1}^{m} p_{ij}\hat{y}_{ijk} && && (3.13)\\
\text{s.t.} \quad & \hat{y}_{ijk} \le x_{ik} && (i \in N \setminus \{n\}, j \in N(j > i), k \in M) && (3.14)\\
& \hat{y}_{ijk} \le x_{jk} && (i \in N \setminus \{n\}, j \in N(j > i), k \in M) && (3.15)\\
& \hat{y}_{ijk} \ge x_{ik} + x_{jk} - 1 && (i \in N \setminus \{n\}, j \in N(j > i), k \in M) && (3.16)\\
& \hat{y}_{ijk} \in \{0,1\} && (i \in N \setminus \{n\}, j \in N(j > i), k \in M) && (3.17)\\
& (3.9), (3.10), (3.11).
\end{aligned}$$

Constraints (3.14) and (3.15) ensure that variable $\hat{y}_{ijk}$ takes the value 0 when at least one of the two variables x_{ik} and x_{jk} is equal to 0. Constraints (3.16) force $\hat{y}_{ijk}$ to take the value 1 when both variables x_{ik} and x_{jk} are equal to 1.

We next show that an equivalent formulation can be obtained by removing constraints (3.16) and (3.17):

Lemma 1 *The optimal solution to the LP relaxation of* FGW *does not change if constraints* (3.16) *are removed.*

Proof. Let $(x^*, \hat{y}^*)$ be an optimal solution to the LP relaxation of FGW *without inequalities* (3.16). The second term of the objective function maximizes a linear function of $\hat{y}$ with coefficients $p_{ij} \ge 0$. It follows that *every variable* $\hat{y}^*_{ijk}$ will take the largest possible value, and hence, from (3.14)-(3.15), $\hat{y}^*_{ijk} = \min\{x^*_{ik}, x^*_{jk}\}$. Since $x^*_{ik} \le 1 \; \forall i \in N$ and $k \in M$, we have $\min\{x^*_{ik}, x^*_{jk}\} \ge x^*_{ik} + x^*_{jk} - 1$. □

Corollary 1 *If the LP relaxation of FGW satisfies* (3.11)*, constraints* (3.17) *are automatically satisfied.*

Proof. Assume that the optimal solution $(x^*, \hat{y}^*)$ to the LP relaxation of FGW satisfies $x^*_{ik} \in \{0,1\} \; \forall i \in N$ and $k \in M$. From the proof of Lemma 1 we have $\hat{y}^*_{ijk} = \min\{x^*_{ik}, x^*_{jk}\}$, and hence $\hat{y}^*_{ijk} \in \{0,1\}$. □

Proposition 1 *Constraints* (3.16) *and* (3.17) *are redundant for* FGW *and for its LP relaxation.*

Proof. Immediate from Lemma 1 and Corollary 1. □

Model FGW has $O(n^2m)$ variables and constraints. A more compact, $O(nm)$, linear model for 01QP was proposed in 1975 by [104], who introduced, for each original variable x_{ik}, a new continuous variable z_{ik} to represent its contribution to the objective function. For the QMKP, let us define, for each $i \in N$ and $k \in M$,

$$g_{ik}(x) = \begin{cases} p_i + \sum_{j=i+1}^{n} p_{ij} x_{jk} & \text{if } i \in N \setminus \{n\}; \\ p_n & \text{if } i = n. \end{cases} \tag{3.18}$$

The contribution of x_{ik} to the objective function is then

$$z_{ik} = g_{ik}(x) x_{ik} \quad (i \in N, k \in M), \tag{3.19}$$

and observe that z_{ik} will always take integer values as the profits are assumed to be integer. The resulting *Glover* model (GLOV) for the QMKP is

$$\text{(GLOV)} \quad \max \quad \sum_{i=1}^{n} \sum_{k=1}^{m} z_{ik} \tag{3.20}$$

$$\text{s.t.} \quad L_i x_{ik} \leq z_{ik} \leq U_i x_{ik} \qquad (i \in N, k \in M) \tag{3.21}$$

$$g_{ik}(x) - U_i(1 - x_{ik}) \leq z_{ik} \leq g_{ik}(x) - L_i(1 - x_{ik}) \quad (i \in N, k \in M) \tag{3.22}$$

$$(3.9), (3.10), (3.11),$$

where $g_{ik}(x)$ is defined in (3.18), while $L_i = p_i + \sum_{j=i+1}^{n} \min\{0, p_{ij}\}$, $U_i = p_i + \sum_{j=i+1}^{n} \max\{0, p_{ij}\}$ (for $i \in N \setminus \{n\}$), and $L_n = U_n = p_n$ are the smallest and largest values, respectively, that g_{ik} (and hence z_{ik}) can take. Note that, as we assume the pairwise profits p_{ij} to be non-negative, these values can be simplified to $L_i = p_i$, $U_i = p_i + \sum_{j=i+1}^{n} p_{ij}$ (for $i \in N$). Constraints (3.21) and (3.22) link variables x_{ik} and z_{ik}: constraints (3.21) impose $z_{ik} = 0$ when $x_{ik} = 0$, while constraints (3.22) impose $z_{ik} = g_{ik}(x)$ when $x_{ik} = 1$. (Note the similarity with the effect of (3.14)-(3.15) and (3.16), respectively.)

GLOV is indeed more compact than FGW, but, as proved by [105], its LP relaxation is weaker.

3.2.2 Reformulation Linearization Technique

In [106] strengthened FGW by proposing a new linearization method for 01QP. The idea was later extended to general 0-1 problems in [107]. The method, known as the *Reformulation Linearization Technique* (RLT), provides different *Levels* of representation with an increasingly stronger LP bound.

Let $\bar{n}$ denote the number of original binary variables appearing in each constraint. New quadratic constraints are added to the original formulation, to strengthen the resulting LP relaxation. At *Level 1*,

(i) each equality constraint results into $\bar{n}$ quadratic constraints obtained by multiplying it by each original binary variable;

(ii) each inequality constraint results into $2\bar{n}$ quadratic constraints obtained by multiplying it by each original binary variable and by its complement.

All the resulting quadratic constraints are then linearized by introducing auxiliary binary variables to represent the products of the original ones together with appropriate linking constraints. Higher levels are rarely used as the problem size increases so sharply that the bound computation becomes impractical.

In order to adapt the RLT to the QMKP, let us define binary variables y_{ijk} similarly to variables $\hat{y}_{ijk}$ of Section 3.2.1, but by considering *all* ordered pairs (i, j) with $i \neq j$, i.e.,

$$y_{ijk} = x_{ik} x_{jk} \text{ for } i \in N, j \in N \setminus \{i\}, k \in M. \tag{3.23}$$

A *Level 1 RLT model* (RLT1) for the QMKP is then

$$\text{(RLT1)} \quad \max \quad \sum_{i=1}^{n}\sum_{k=1}^{m} p_i x_{ik} + \tfrac{1}{2}\sum_{i=1}^{n}\sum_{\substack{j=1\\ j\neq i}}^{n}\sum_{k=1}^{m} p_{ij} y_{ijk} \tag{3.24}$$

$$\text{s.t.} \quad y_{ijk} \leq x_{ik} \qquad (i\in N, j\in N\backslash\{i\}, k\in M) \tag{3.25}$$

$$y_{ijk} = y_{jik} \qquad (i\in N\backslash\{n\}, j\in N(j>i), k\in M) \tag{3.26}$$

$$y_{ijk} \geq x_{ik} + x_{jk} - 1 \qquad (i\in N, j\in N\backslash\{i\}, k\in M) \tag{3.27}$$

$$\sum_{j\in N\backslash\{i\}} w_j y_{ijk} \leq (C_k - w_i)\, x_{ik} \qquad (i\in N, k\in M) \tag{3.28}$$

$$\sum_{j\in N\backslash\{i\}} w_j (x_{jk} - y_{ijk}) \leq C_k(1 - x_{ik}) \qquad (i\in N, k\in M) \tag{3.29}$$

$$y_{ijk} \in \{0,1\} \qquad (i\in N, j\in N\backslash\{i\}, k\in M) \tag{3.30}$$

$$(3.9), (3.10), (3.11).$$

(3.29) are the RTL constraints derived for each knapsack k ($k \in M$) from the corresponding capacity constraint (3.9) by multiplying both the left-hand side and right hand side times the binary variable x_{ik} and its complement (1 - x_{ik}) for each item $i(i \in N)$. For each pair (i,k), with $i \in N$ and $k \in M$, by multiplying (3.9) times x_{ik} we obtain:

$$x_{ik} \sum_{j\in N} w_j x_{jk} \leq C_k x_{ik} \quad (i \in N, k \in M) \tag{3.31}$$

From which we obtain:

$$\sum_{j\in N\backslash\{i\}} w_j x_{ik} x_{jk} \leq C_k x_{ik} - w_j x_{ik} \quad (i \in N, k \in M) \tag{3.32}$$

And then:

$$\sum_{j\in N\backslash\{i\}} w_j y_{ijk} \leq (C_k - w_j) x_i \quad (i \in N, k \in M) \tag{3.33}$$

Which corresponds to (3.28). For each pair (i,k), with $i \in N$ and $k \in M$, by multiplying (3.9) times (1 - x_{ik}) we obtain:

$$(1 - x_{ik}) \sum_{j\in N} w_j x_{jk} \leq C_k(1 - x_{ik}) \quad (i \in N, k \in M) \tag{3.34}$$

From which we obtain:

$$\sum_{j\in N\backslash\{i\}} w_j (x_{jk} - x_{ik} x_{jk}) + w_i (x_{ik} - x_{ik}^2) \leq C_k(1 - x_{ik}) \quad (i \in N, k \in M) \tag{3.35}$$

And then:

$$\sum_{j \in N \setminus \{i\}} w_j(x_{jk} - y_{ijk}) \leq C_k(1 - x_{ik}) \quad (i \in N, k \in M) \tag{3.36}$$

Which corresponds to (3.29).

We next show that if we drop the RLT constraints from RLT1, the LP relaxation of the resulting model is equivalent to the LP relaxation of FGW.

Proposition 2 *The polyhedra associated with the LP relaxation of* RLT1 *without the* RLT *constraints* (3.28)-(3.29)*, and the LP relaxation of* FGW *are isomorphic under the linear transformation*

$$\hat{y}_{ijk} = y_{ijk} = y_{jik} \ \forall\, i, j \in N\, (j > i),\, k \in M$$

(*with x unchanged*).

Proof. Inequalities (3.25) and (3.26) imply $y_{ijk} \leq x_{jk}\ \forall\, i, j \in N\, (i \neq j),\, k \in M$. By observing the different j-indexing in the two objective functions, it easily follows that the two solutions produce the same value. □

If, besides removing the RLT constraints, we also remove inequalities (3.27), the resulting LP bound is still as strong as the one produced by the LP relaxation of FGW:

Corollary 2 *The LP relaxation of* RLT1 without constraints (3.27)-(3.29) *is equivalent to the LP relaxation of* FGW*.*

Proof. According to Proposition 2, the polyhedra associated with the LP relaxations of the two models are isomorphic. Lemma 1 guarantees that inequalities (3.27) can be removed without changing the optimal value. □

Note that RLT1 does not include the RLT constraints obtained from cardinality constraints (3.10). The reason for this comes from our choice of having *3-index* variables. Indeed, by applying RLT to (3.10), we would obtain products involving different knapsacks, for which an additional index would be needed. On the one hand, this choice makes the LP relaxation of the resulting model weaker, but, on the other hand,

(i) it produces a more compact model, of size $O(n^2m)$ (instead of $O(n^2m^2)$), which lends itself to a much faster computation of the resulting LP bound;

(ii) RLT1 can be effectively decomposed, as shown in the next section.

A decomposable Level 1 RLT model

In this section we show how, starting from RLT1, we can construct a new linear reformulation that is amenable to a *decomposable Lagrangian relaxation* (to be examined in Section 3.4.2) that: (i) provides a stronger bound than the one given by its continuous relaxation, and (ii) can be computed with reasonable computational effort.

Point (i) obviously requires that the relaxed model does not have the integrality property (see, e.g., [108]). An effective way to pursue point (ii) is to obtain a "decomposable" Lagrangian problem, leading to a set of independent, well-structured subproblems. We generalize the approach presented by [109] for the single QKP. The same approach was later applied by [110] to the p-dispersion problem, then generalized by [111] to 0-1 quadratic problems with linear constraints, and recently adopted by [112] for a generalization of the quadratic assignment problem. Recall that the coefficients of the quadratic terms of the objective function (i.e., the pair-wise profits p_{ij}) are assumed to be non-negative, as it normally holds for the QMKP instances considered in the literature.

Let y_{ijk} be defined as in (3.23). A *Decomposable Level 1 RLT model* (DRLT1) for the QMKP can be obtained from RLT1 by eliminating constraints (3.27) and (3.29), i.e.,

$$\text{(DRLT1)} \quad \max \quad \sum_{i=1}^{n}\sum_{k=1}^{m} p_i x_{ik} + \frac{1}{2} \sum_{i=1}^{n}\sum_{\substack{j=1\\ j\neq i}}^{n}\sum_{k=1}^{m} p_{ij} y_{ijk} \tag{3.24}$$

$$\text{s.t.} \quad y_{ijk} \le x_{ik} \qquad (i \in N, j \in N \setminus \{i\}, k \in M) \tag{3.25}$$

$$y_{ijk} = y_{jik} \qquad (i \in N \setminus \{n\}, j \in N (j > i), k \in M) \tag{3.26}$$

$$\sum_{j \in N \setminus \{i\}} w_j y_{ijk} \le (C_k - w_i)\, x_{ik} \qquad (i \in N, k \in M) \tag{3.28}$$

$$y_{ijk} \in \{0, 1\} \qquad (i \in N, j \in N \setminus \{i\}, k \in M) \tag{3.30}$$

(3.9), (3.10), (3.11).

Note that the effect of RLT1 constraints (3.29) was purely to strengthen the continuous relaxation of the model. Moreover, as formally proved in Proposition 1 (also see [109] and [111]), constraints (3.27) are redundant when the coefficients of the quadratic term are non-negative. Therefore, DRLT1 is a valid (linear) reformulation for the QMKP.

The continuous relaxation of DRLT1 is weaker than that of RLT1 but *stronger* than that of FGW. In addition, it has the advantage that *dualizing* constraints (3.26) results in a *decomposable* Lagrangian relaxed problem, that does not have the integrality property, as we will show in Section 3.4.2.

Proposition 3 *The LP relaxation of* DRLT1 *is stronger than the LP relaxation of* FGW.

Proof. From Corollary 2, the LP relaxation of FGW is as strong as the LP relaxation of DRLT1 *without constraints* (3.28). Therefore, it is enough to show an example where inequalities (3.28) improve the LP bound. Consider an instance consisting of a single knapsack of capacity $C = 8$, and three items with $w_1 = 2$, $w_2 = 8$, $w_3 = 5$, $p_1 = 1$, $p_2 = 3$, $p_3 = 1$, and pairwise profits $p_{12} = 4$, $p_{13} = 2$, $p_{23} = 2$. The optimal solution of the LP relaxation of DRLT1 is $\bar{x}_1 = \bar{x}_3 = 1$, $\bar{x}_2 = 0.125$, $\bar{y}_{13} = \bar{y}_{31} = 1$ (all other $\bar{y}$ being 0) and has value 4.375. The optimal solution of the LP relaxation of FGW is instead $\bar{x}_1 = \bar{x}_2 = \bar{x}_3 = 0.5\bar{3}$, $\bar{y}_{12} = \bar{y}_{13} = \bar{y}_{23} = 0.5\bar{3}$ and has value $6.9\bar{3}$. □

Model DRLT1 can be improved by means of the following considerations:

(i) variables y_{ijk} for which $p_{ij} = 0$ can always be set to zero;

(ii) variables y_{ijk} for which $w_i + w_j > C_k$ must take the value zero;.

(iii) due to Corollary 1, constraints (3.30) can be relaxed in a continuous way.

By defining

$$S_{ik} = \{j \in N \setminus \{i\} \; : \; p_{ij} > 0 \text{ and } w_i + w_j \leq C_k\} \; (i \in N, k \in M); \tag{3.37}$$

$$T_{ik} = \{j \in N \setminus \{i\} \; : \; w_i + w_j > C_k\} \; (i \in N, k \in M); \tag{3.38}$$

$$R_{ik} = \{j \in N \; : \; j > i, p_{ij} > 0, \text{ and } w_i + w_j \leq C_k\} \; (i \in N \setminus \{n\}, k \in M), \tag{3.39}$$

we get the *Modified Decomposable Level 1 RLT model* (MDRLT1)

$$(\text{MDRLT1}) \quad \max \quad \sum_{i=1}^{n} \sum_{k=1}^{m} p_i x_{ik} + \frac{1}{2} \sum_{i=1}^{n} \sum_{k=1}^{m} \sum_{j \in S_{ik}} p_{ij} y_{ijk} \tag{3.40}$$

$$\text{s.t.} \quad y_{ijk} \leq x_{ik} \qquad (i \in N, k \in M, j \in S_{ik}) \tag{3.41}$$

$$y_{ijk} = y_{jik} \qquad (i \in N \setminus \{n\}, k \in M, j \in R_{ik}) \tag{3.42}$$

$$\sum_{j \in S_{ik}} w_j y_{ijk} \leq (C_k - w_i)\, x_{ik} \qquad (i \in N, k \in M) \tag{3.43}$$

$$y_{ijk} \geq 0 \qquad (i \in N, k \in M, j \in S_{ik}) \tag{3.44}$$

$$x_{ik} + x_{jk} \leq 1 \qquad (i \in N, k \in M, j \in T_{ik}) \tag{3.45}$$

(3.9), (3.10), (3.11).

3.3 Surrogate relaxation of the quadratic model

A classical relaxation technique for the (linear) MKP is obtained by surrogating the capacity constraints (3.9) with multipliers $\pi_k \geq 0$ $(k \in M)$. Its popularity comes from the fact that, as proved by [113], the optimal choice for the surrogate multipliers is to have them all equal to any positive number. We next show that such property carries through to the quadratic case.

For the QMKP, the surrogate relaxation of the quadratic model (3.8)-(3.11) is:

$$S(\pi) = \max \quad \sum_{i=1}^{n} \sum_{k=1}^{m} p_i x_{ik} + \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \sum_{k=1}^{m} p_{ij} x_{ik} x_{jk} \tag{3.46}$$

$$\text{s.t.} \quad \sum_{k=1}^{m} \pi_k \sum_{i=1}^{n} w_i x_{ik} \leq \sum_{k=1}^{m} \pi_k C_k \tag{3.47}$$

$$\sum_{k=1}^{m} x_{ik} \leq 1 \qquad (i \in N) \tag{3.48}$$

$$x \in \{0,1\}^{n \times m}. \tag{3.49}$$

Lemma 2 *There always exists an optimal solution to (3.46)-(3.49) that assigns all the selected items to the knapsack with smallest surrogate multiplier.*

Proof. Let $k^* = \arg\min\{\pi_k : k \in M\}$ and let x be a feasible solution to (3.46)-(3.49). Another feasible solution $\bar{x}$, not worse than x, can be obtained by setting $\bar{x}_{ik} = 0$ and $\bar{x}_{ik^*} = 1$ for each $i \in N$ such that $x_{ik} = 1$ and $k \neq k^*$. □

Proposition 4 *The optimal vector of multipliers for* (3.46)-(3.49) *is* $\pi_k = \bar{\pi}$ *(where* $\bar{\pi}$ *is any positive constant) for all* $k \in M$.

Proof. Using Lemma 2, (3.46)-(3.49) is equivalent to the (single) QKP

$$
\begin{aligned}
S(\pi) = \max \quad & \sum_{i=1}^{n} p_i x_{ik^*} + \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} p_{ij} x_{ik^*} x_{jk^*} \\
\text{s.t.} \quad & \sum_{i=1}^{n} w_i x_{ik^*} \leq \left\lfloor \sum_{k=1}^{m} \frac{\pi_k}{\pi_{k^*}} C_k \right\rfloor \\
& x_{ik^*} \in \{0,1\} \qquad\qquad (i \in N).
\end{aligned}
\tag{3.50}
$$

where k^* is the (knapsack) index corresponding to the smallest surrogate multiplier. Since $\left\lfloor \sum_{k=1}^{m} \frac{\pi_k}{\pi_{k^*}} C_k \right\rfloor \geq \sum_{k=1}^{m} C_k$, the choice $\pi_k = \bar{\pi}$ (any positive constant) for all $k \in M$ produces the minimum capacity and hence the minimum value of $S(\pi)$. □

3.4 Decomposable Lagrangian relaxations

In this section we study the Lagrangian relaxation when applied to the quadratic formulation (3.8)-(3.11) of Section 3.1, and to the DRLT1 formulation of Section 3.1.

3.4.1 Relaxing the Quadratic Model

A classical relaxation of the MKP is obtained by relaxing in a Lagrangian fashion the cardinality constraints (3.10) with multipliers $\lambda_i \geq 0$ $(i \in N)$. For the QMKP, such relaxation becomes:

$$
\begin{aligned}
L^Q(\lambda) = & \sum_{i=1}^{n} \lambda_i + \max \sum_{i=1}^{n} \sum_{k=1}^{m} (p_i - \lambda_i) x_{ik} + \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \sum_{k=1}^{m} p_{ij} x_{ik} x_{jk} \\
\text{s.t.} \quad & \sum_{i=1}^{n} w_i x_{ik} \leq C_k \qquad\qquad (k \in M) \\
& x \in \{0,1\}^{n \times m}.
\end{aligned}
\tag{3.51}
$$

As the objective function does not contain terms involving items assigned to different knapsacks, the problem decomposes into m independent QKPs (one for each knapsack $k \in M$).

It is worth mentioning that, if the knapsack set M is partitioned into t subsets $M_1, \ldots, M_t$, such that all knapsacks in M_h $(h = 1, \ldots, t)$ have the same capacity $\overline{C}_h$, the optimal solution to the above Lagrangian relaxation can be obtained by solving t independent QKPs. Indeed, for each subset M_h, it is enough to solve one single QKP and to sum up the optimal values. Such situation occurs, e.g., in the benchmark instances by [101], where all knapsacks have the same capacity.

In order to solve the *Lagrangian dual* problem, i.e., to find the best possible set of multipliers, λ^*, in our computational experiments we adopted the *proximal bundle method*, as implemented by [114]. The corresponding software is freely available at `https://gitlab.com/frangio68/ndosolver_fioracle_project` (as a part of the `NDOSolver/FiOracle` suite of `C++` solvers for NonDifferentiable Optimization, developed by the Department of Computer Science of the University of Pisa).

3.4.2 Relaxing DRLT1

A different Lagrangian relaxation can be obtained by the DRLT1 model introduced in Section 3.1. Let us dualize the symmetry equations (3.26) with multipliers $\lambda_{ijk} \lesseqgtr 0$. We get:

$$L^R(\lambda) = \max \quad \sum_{i=1}^{n}\sum_{k=1}^{m} p_i x_{ik} + \sum_{i=1}^{n}\sum_{\substack{j=1\\j\neq i}}^{n}\sum_{k=1}^{m}(\tfrac{1}{2}\,p_{ij} + \lambda_{ijk})y_{ijk} \tag{3.52}$$

$$\text{s.t.} \quad y_{ijk} \le x_{ik} \qquad (i\in N, j\in N\setminus\{i\}, k\in M) \tag{3.53}$$

$$\sum_{j\in N\setminus\{i\}} w_j y_{ijk} \le (C_k - w_i)\; x_{ik} \; (i \in N, k \in M) \tag{3.54}$$

$$y_{ijk} \in \{0,1\} \qquad (i\in N, j\in N\setminus\{i\}, k\in M) \tag{3.55}$$

$$(3.9), (3.10), (3.11).$$

Since the multipliers λ_{ijk} for the symmetry constraints (3.26) are only defined for $j > i$, we assume, for notational convenience, that $\lambda_{jik} = -\lambda_{ijk}$ in (3.52).

The main reason for relaxing (3.26) is that the resulting model has a *decomposable* structure. Observe indeed that constraints (3.53) allow a variable y_{ijk} to be 1 only if x_{ik} is 1. Moreover, for each pair i,k ($i \in N, k \in M$), variables y_{ijk} ($j \in N \setminus \{i\}$) only appear in capacity constraints (3.54) and in the objective function. Hence, if all x_{ik} variables are fixed, the relaxed problem consists of nm independent sub-problems, one for each pair i,k. More precisely, the relaxed problem decomposes into $nm+1$ sub-problems, that can be cascaded as follows:

(i) first we solve nm (linear) KPs, one for each pair i,k ($i \in N, k \in M$), of the form:

$$\begin{aligned} \max \quad & \sum_{\substack{j=1\\j\neq i}}^{n}(\tfrac{1}{2}\,p_{ij} + \lambda_{ijk})y_{ijk} \\ \text{s.t.} \; & \sum_{j\in N\setminus\{i\}} w_j y_{ijk} \le (C_k - w_i)x_{ik} \\ & y_{ijk} \in \{0,1\} \qquad (j \in N \setminus \{i\}) \\ & x_{ik} \in \{0,1\}. \end{aligned} \tag{3.56}$$

having only one x_{ik} variable and its associated $n-1$ auxiliary variables y_{ijk} ($j \in N \setminus \{i\}$) subject to a single capacity constraint (3.56) associated with the pair (i,k). We denote by v_{ik} the optimal solution value when $x_{ik} = 1$, while the optimal solution value is clearly 0 when $x_{ik} = 0$.

(ii) then we solve a unique (linear) *pseudo*-MKP with all the original x_{ik} variables subject to constraints (3.9)-(3.11):

$$\begin{aligned} \max \quad & \sum_{i=1}^{n} \sum_{k=1}^{m} \tilde{p}_{ik} x_{ik} && (3.57) \\ \text{s.t.} \quad & (3.9), (3.10), (3.11), \end{aligned}$$

where $\tilde{p}_{ik} = p_i + v_{ik}$ $(i \in N, k \in M)$.

Observe that, as it is known that the polyhedron of the 0-1 knapsack problem KP is not integral (see, e.g., [108]), our Lagrangian problem does not have the integrality property. Therefore, the Lagrangian bound, corresponding to the optimal dual multipliers λ^*, is not dominated by the standard continuous relaxation of DRLT1.

In this case too we performed our computational experiments by solving the Lagrangian dual problem by means of the proximal bundle method, as implemented by [114].

3.5 Meta-heuristic Approach: The Multi-Start Iterated Local Search

Several large size benchmark instances have been presented for the QMKP. Billionnet and Soutif originally proposed these instances for testing algorithms for QKP [83]. However, for the QMKP, a set of knapsacks with homogeneous capacities is defined, and the corresponding instances are very difficult to solve with exact methods. For solving larger instances, meta-heuristics have been the most used technique. Evolutionary Algorithms [89, 90, 115], Tabu Search [94, 97], Path Relinking [96],and heuristics based on thresholding [93, 96].

Large instances of QMKP remain a challenge for the state of the art algorithms. The current instances for the QMKP, based on the single QKP have been randomly generated, with number of items 100 and 200 and can be found `http://cedric.cnam.fr/soutif/QKP/QKP.html`. The latest results about these instances are based on time limit stopping criteria, so possibly the best results for the considered instances have not been explored yet. This section presents the Multi-Start Iterated Local Search (MS-ILS) algorithm for the solution of large instances of the problem.

The MS-ILS is based on the classic ILS described in the previous chapter. For an ILS, four essential processes must be defined: initial solution, perturbation, local search, and acceptance criteria. An initial solution can be generated randomly or by executing a constructive heuristic for the problem. A perturbation allows to diversify the search space by making random moves to a current solution; perturbation must control this process since a very aggressive move can generate a solution that does not help in the search as it is far from the optimum of the problem, while a less aggressive move can stall the process at a local optimum. The local search allows to intensify the search based on a candidate solution or current solution and determines if it is possible to change the neighborhood. For this, the local search strategy can be determined based on how the neighborhoods are applied (first improvement or best improvement) and the order in which they are applied (one neighborhood first or all simultaneously). Finally, the acceptance criterion allows to accept solutions no better than the current solution with a defined threshold to "navigate" in a larger problem-solution space.

The MS-ILS performs an ILS using more than one initial solution. The initial solution for the ILS can be changed during the execution and allows to diversify the search for the meta-heuristics. Generally, random processes based on properties of the problem or deterministic methods that allow to give stability to the algorithm considering the final result are used. In the following subsections, the meta-heuristic for QMKP is presented in detail, considering the critical ILS processes and initial solutions for multiple executions.

3.5.1 MS-ILS: Main Scheme

The MS-ILS approach generates initial solutions using different algorithms and applies Perturbations and local searches for the QMKP. The details of this metaheuristic are described in Algorithm 11. The algorithm receives as input parameters: $maxStart$ corresponding to the number of ILS executions with different initial solutions; $maxIter$ corresponding to the maximum number of iterations within the ILS; $r1$, $r2$ and $r3$ are numbers between 0 and 1 that allow to define in which iteration to use a specific initial solution; $maxPl$ corresponding to the maximum number of iterations; λ and $maxTry$ are parameters for the perturbation (see the details later in Algorithm 13); $tabuTenure_{inf}$ and $tabuTenure_{fea}$ corresponding to the length of different tabu lists used by the MS-ILS; M and β are parameters for the infeasible local search (see the details in Algorithm 14). Initially, we define the variable s to control the number of times the ILS is executed; we also define an empty list of tabu solutions that the algorithm will use during the execution of the MS-ILS. Finally, the $Quad - GreedyH()$ algorithm is executed (see the details in the following subsection).

Depending on the current value of s, an initial solution is generated and stored in $x_{initial}$ (lines 5 to 15), and a local search procedure is applied (line16). The first iteration of the MS-ILS allows to define x_{best}, corresponding to the best solution found by the metaheuristic.

The executions of a single ILS are carried out with the input parameters of the algorithm. The internal while-loop of the MS-ILS (lines 22-36) allows for a generation of perturbations based on thresholds and local searches in the feasible space of the problem. The tabu list solutions is also updated to avoid to run procedure $FeaLS$ redundantly. Also, if an improvement is found during the execution of the while-loop, the variables concerning the termination condition are reset (lines 29 and 30).

Finally, an infeasible local search process is executed. First, the list of tabu solutions is checked to avoid redundant executions of the algorithm (line 37). Subsequently, infeasible local search and repair algorithms are implemented. The global variables are updated before closing the main while-loop (lines 49-52).

Algorithm 11 MS-ILS: Main Scheme

Input: $maxStart, maxIter, r1, r2, r3, maxPl, \lambda, maxTry, tabuTenure_{inf}, tabuTenure_{fea}, M, \beta$
Output: BestProfit, x_{Best}

```
s = 1
tabuList = ∅
$x_{quad}$ = $Quad - GreedyH()$
while s ≤ maxStart do
    if 0 <(s/maxStart) ≤ $r1$ then
        $x_{initial}$ = $x_{quad}$
    end if
    if $r1$ <(s/maxStart) ≤ $r2$ then
        $x_{initial}$ = $Random - FeasibleH()$
    end if
    if $r2$ <(s/maxStart) ≤ $r3$ then
        $x_{initial}$ = $Greedy - RandomH()$
    else
        $x_{initial}$ = $x_{InfLS}$
    end if
    ($x_{ls}, tabuList, tabuCond$) = FeaLS($x_{initial}, tabuList, tabuTenure_{fea}$)
    if s==1 then
        $x_{best}$ = $x_{ls}$
    end if
    $f = 1$
    $pertLength = 1$
    while $f \leq maxIter$ and $pertLength \leq maxPl$ do
        $xp$ = Perturbation($x_{ls}$,pertLength,$\lambda$,maxTry,0.5,0.5)
        ($x_{feaLS}, tabuList, tabuCond$)=FeaLS($xp, tabuList, tabuTenure_{fa}$)
        if $tabuCond == 1$ then
            $pertLength = pertLength + 1$
        else
            if f($x_{feaLS}$) > f($x_{ls}$) then
                $x_{ls} = x_{feaLS}$
                $f = 1$
                $pertLength = 1$
            else
                $f = f + 1$
            end if
        end if
    end while
    if $x_{feaLS} \notin tabuList$ then
        ($xFeasible, xInfeasible$)=InfLS($x_{feaLS}, M, tabuTenure_{inf}, \beta$)
        $x_{feasible2}$ = Repair($xInfeasible, xFeasible, 10$)
        $x_{InfLS}$ = Best($xFeasible, x_{feasible2}$)
        if f($x_{InfLS}$) > f($x_{ls}$) then
            if $x_{InfLS} \notin tabuList$ then
                $x_{ls} = x_{InfLS}$
                $f = 1$
                $pertLength = 1$
            end if
        end if
    end if
    if f($x_{ls}$)> f($x_{best}$) then
        $x_{best}$ = $x_{ls}$
    end if
    s=s+1
end while
return $f(x_{best}), x_{best}$
```

Initial Solutions

Four ways to obtain initial solutions are used in the MS-ILS:

- *Quad-GreedyH* consists of optimally solving the QKP with the available items and the capacity of the homogeneous knapsack (i.e. C_k). This last process must be carried out m times always using the remaining available items that the algorithm did not select in the previous iterations. In order to solve each QKP, the *quadknap* algorithm was used.

- *Random-FeasibleH* consists of randomly selecting an item and assigning randomly it to a knapsack whenever the latter has available capacity.
- *Greedy-RandomH* consists of randomly selecting an item and assigning it to the best knapsack, improving the objective function for the QMKP.
- Use the best feasible solution found by the last MS-ILS run.

In Algorithm 11 (line 3), the *Quad-GreedyH* function is called only once, since it always generates a deterministic solution, while the remaining functions are called within the main loop since they use random processes and/or depend on past executions.

Feasible Local Search

The feasible local search works by considering on the Relocate and Exchange neighborhoods. *BestRelocateSolution* relocates each (available and unavailable) item between the different knapsacks, returning the best solution from the search neighborhood. *BestExchangeSolution* exchanges each pair of (available and unavailable) items between the different knapsacks (available and unavailable items), returning the best solution from the search neighborhood. All neighborhoods work based on the best improvement.

Another essential point of the feasible local search is how the neighborhoods are applied. Algorithm 12 receives as parameters: x_{**} the current solution, *TabuList* the list of the best solutions found, and $tabuTenure_{fea}$ the size of the tabu list. The algorithm begins by checking if the initial solution (x_{**}) is in the tabu list, in order to avoid to perform a redundant execution in the local search. Subsequently, a *VND* is executed: first, a neighborhood is applied; if this move does not improve the solution or if the solution found is tabu, it is changed to the second neighborhood. If the second neighborhood generates an improvement, it returns to the previous neighborhood (line 20). The above steps are done iteratively until no neighborhood can improve the solution. Finally, a list of local tabu solutions is considered for the *FeaLS* process, which stores all the solutions found (line 21) and updates the global tabu list *tabuList* at the end of the local search.

Algorithm 12 FeaLS

```
Input: $x_{**}, tabuList, tabuTenure_{fea}$
Output: $x_{ls}, tabuList, tabuCond$
$x_{ls} = x_{**}$
if $x_{ls} \in tabuList$ then
    $tabuCond = 1$
    return $x_{ls}, tabuList, tabuCond$
end if
$k = 1$
$tabuCond = 0$
$tabuListSolution = \emptyset$
while $k <= 2$ do
    if $k == 1$ then
        $x_m$ = BestExchangeSolution($x_{ls}$,$f(x_{ls})$)
    else
        $x_m$ = BestRelocateSolution($x_{ls}$,$f(x_{ls})$)
    end if
    if $x_m \in tabuList$ then
        $k = k + 1$
    else
        if $f(x_m) > f(x_{ls})$ then
            $x_{ls} = x_m$
            $k = 1$
            tabuListSolution.add($x_{ls}$)
        else
            $k = k + 1$
        end if
    end if
end while
for $solution \in tabuListSolution$ do
    $tabuList.update(solution, tabuTenure_{fea})$
end for
return $x_{ls}, tabuList, tabuCond$
```

Perturbation

The perturbation for the MS-ILS is based on the perturbation process defined in [116] for the VRP and [117] for the Generalized-QMKP. This perturbation is defined based on the concept of perturbation length ($pertLength$), the number of moves performed in the input solution; and a threshold (γ)that allows to control the impact of each move. In the perturbation process described in Algorithm 13, the variables of the process are initialized in the first lines; later, within the main loop, a random neighborhood is selected with probability *Neighborhood1* for Exchange and *Neighborhood2* for Relocate, and this move is iteratively applied.

Finally, if the solution is accepted with respect to $f(xp) * \gamma$ (where xp is the perturbed solution and, $f(xp)$ is profit of xp.), the search variables are replaced. Also, note that if the solution is not accepted, the counter variable *try* is used in order to update γ.

Algorithm 13 Perturbation

```
Input: $x_{ls}$,pertLength,$\lambda$,maxTry,$Neighborhood1$,$Neighborhood2$
Output: $xp$
$p = 1$
$\gamma = 1 - \lambda$
$xp = x_{ls}$
$try = 0$
while $p <= pertLength$ do
    $neighRandom$ = RandomNeighborhoodSelect($Neighborhood1$,$Neighborhood2$)
    $xpw$ =applySingleRandomMove($neighRandom, xp$)
    if $f(xpw) > f(xp) * \gamma$ then
        $xp = xpw$
        $\gamma = \gamma - \lambda$
        $p = p + 1$
        $try = 0$
    else
        $try = try + 1$
        if $try > maxTry$ then
            $\gamma = \gamma - \lambda$
            $try = 0$
        end if
    end if
end while
return $xp$
```

Infeasible local Search and the Repair procedure

The infeasible local search is based on the procedure presented by [97]. This search method is based on two major processes:

- The first process applies the *BestExchangeSolution* and *BestRelocateSolution* algorithms by respecting the capacity of each knapsack (we always assume homogeneous capacities). When the first process falls into a local optimum, i.e., the method cannot continue to improve the current feasible solution, the second process is executed.

- The second process consists of the relaxation of the capacity restriction, and the relocate neighborhood is used to explore infeasible areas. Generally, the relocation move of this process moves items from a heavy knapsack to a light knapsack.

The second process works stochastically, randomly choosing a knapsack and applying the relocation move. A function *Relocate(item,k1,k2)* is defined for the infeasibility process, this function returns:

$$Relocate(item, k1, k2) = (\Delta(item, k2) - \Delta(item, k1)) / w_i^{\beta} \tag{3.58}$$

where $\Delta(item, knapsack) = p_{item} + \sum_{j \in knapsack} p_{item,j}$. Algorithm 14 describes in detail the method from the [97], where at lines 23 and 24 the selection of the knapsack, and the definition of the corresponding weight are performed:

- If $W_{kinput} < C_{kinput}$, the algorithm selects the item $i \in k^{'}$, and the knapsack $k^{'} \in \{k_b \in K | W_{k_b} \geq W_{kinput}\}$ such that $max_{i \in k^{'}}\{Relocate(i, k^{'}, kinput)\}$ (procedure *RelocateInfGain1*).

- Otherwise, the algorithm selects the item $i \in kinput$ and the knapsack $k^{''} \in \{k_s \in K | W_{k_s} \leq W_{kinput}\}$ such that $max_{i \in kinput}\{Relocate(i, kinput, k^{''})\}$ is selected (procedure *RelocateInfGain2*).

Algorithm 14 InfLS

```
Input: $x_c, M, tabuTenure_{inf}, \beta$
Output: $xFeasible, x_s$ (i.e. $x_s$= the infeasible solution)
$m = 1$
$xFeasible = x_c$
$x_s = x_c$
$tabuListMovements = \emptyset$
$localOptFlag = False$
$feasibleFlag = True$
while $m <= M$ do
    if $(localOptFlag == False)$ and $(feasibleFlag == True)$ then
        $//FirstProcess$
        $(x_{exc}, \gamma_1)$ = BestExchangeSolution($x_s$,$f(x_s$,$TabuList)$
        $(x_{rel}, \gamma_2)$ = BestRelocateSolution($x_s$,$f(x_s$,$TabuList)$
        $(x_s, \gamma_{best})$ = SelectBest($x_{exc}$,$x_{rel}$,$\gamma_1$,$\gamma_2$)
        $tabuListMovements.update(\gamma_{best}, tabuTenure_{inf})$
        if $f(x_s) > f(xFeasible)$ then
            $xFeasible = x_s$
            $localOptFlag = False$
            $feasibleFlag = True$
        else
            $localOptFlag = True$
        end if
    else
        $//SecondProcess$
        $kinput$ = RandomKnapsack(1,K,0)
        $W_{kinput}$ = KnapsackWeight($kinput$)
        if $W_{kinput} < C_{kinput}$ then
            $(x_s, \gamma) = RelocateInfGain1(x_s, \beta, kinput, W_{kinput}, TabuList)$
        else
            $(x_s, \gamma) = RelocateInfGain2(x_s, \beta, kinput, W_{kinput}, TabuList)$
        end if
        $tabuListMovements.update(\gamma, tabuTenure_{inf})$
        if Feasible($x_s$)== $True$ then
            $xFeasible = x_s$
            $feasibleFlag = True$
            $localOptFlag = False$
        else
            $feasibleFlag = False$
        end if
    end if
    $m = m + 1$
end while
return $xFeasible, x_s$
```

All processes do not consider the capacity restriction, but consider the *TabuList* received as an input parameter. Finally, a repair procedure is applied. Three neighborhoods are used for this process: extraction, exchange, and relocation, that work on the same the infeasible solution. The move that generates the least infeasibility is selected, i.e., the move that produces the smallest overload with respect to the knapsack capacity. If there is a tie (i.e., more than one neighborhood generate the same degree of infeasibility), the algorithm selects the move that produces the highest profit (see Appendix A).

A second approach is presented in the next section. Previously described procedures concerning the Lagrangian relaxation and the local search processes are considered.

3.6 Matheuristics Approach

An effective solution process arises when heuristics and exact techniques are used in combination to solve a problem. This process gives rise to a solution approach known as Matheuristic, a field responsible for generating approaches that use mathematical programming and heuristics and/or metaheuristics, one within the other or both executed sequentially [118]. Consider the QKP described above: a Greedy heuristic can quickly find feasible solutions but of poor quality. On the other hand, an integer programming (IP) approach can find high-quality solutions in intractable computing times [51]. However, the combination of a pre-processing heuristic that allows to identify the items that are not part of a good solution for the QKP with the solution of the resulting subproblem by using an IP approach, can generate high-quality solutions. This process can be iterative, and the heuristics can deliver small subproblems to the IP model until a defined stop criterion is reached. Matheuristics have been studied in great depth in the field of operations research, and have been shown to be effective for various combinatorial optimization problems.

The Lagrangian relaxation of the quadratic model proposed for the QMKP process (see section 3.4.1) can generate reasonable solutions. Observe that a valid QKP solution and a valid upper bound UB are obtained at each iteration the bundle procedure. Thus, to solve the QMKP effectively, a new matheuristic algorithm can be considered. This approach is based on the methods previously described:

- The bundle procedure used to get the optimal multipliers for the Lagrangian relaxation of the quadratic model.
- The local search neighborhoods to improve the solutions in the MS-ILS.
- The search process proposed by [97] that considers infeasible moves throughout the thesis and that is also addressed in the MS-ILS.

This hybrid approach for the solution of QMKP is new and could be applied to other optimization problems, depending on the quality of the Lagrangian relaxation solutions determined during the executions of the bundle procedure. The following subsections describe the proposed matheuristic in detail considering the processes mentioned above.

Matheuristic: Main Scheme

Before executing the matheuristic, it is necessary to solve the Lagrangian relaxation of the quadratic model. Let us assume that the Bundle procedure requires t iterations. In this way, at the end of the bundle procedure, we have t solutions of the single QKP and t upper bound values.

The matheuristic receives the solutions generated by the bundle procedure and generates feasible solutions for the QMKP. Algorithm 15 details the input variables:

$solQKP$ a set of solutions for the single QKP obtained by the bundle procedure, UB a vector of upper bound values associated with each solution from $solQKP$, UBM a threshold value to generate a subset from $solQKP$ and a scalar δ used during the matheuristic.

Algorithm 15 Matheuristic 1

```
Input: $solQKP, UB, UBM, \delta$
Output: $bestSolution$
 1: $(A^*, s^*) = selectThreshold(solQKP, UB, UBM)$
 2: $(A, s) = DeleteRedundanceSolutions(A^*)$
 3: $b = GetNumberOfElements(A)$
 4: $bestSolution = \emptyset$
 5: for $r = 1, 2..m$ do
 6:     if $r > s$ then
 7:         $break$
 8:     end if
 9:     for $i = 1, 2..s$ do
10:         $g_i = b_i * \delta$
11:     end for
12:     $StopCriteria = False$
13:     $YY = \emptyset$
14:     $it = 1$
15:     while $StopCriteria == False$ do
16:         $y = SolveCCSP(A, g, r)$
17:         if $y \in YY$ then
18:             $StopCriteria = True$
19:         else
20:             $YY.add(y)$
21:             for $i = 1...s$ do
22:                 $g_i = g_i - y_i * (10 + it + 1)$
23:             end for
24:             $currentSolution = GetSolutions(A, y)$
25:             $nKnap = GetNomberOfKnapsack(currentSolution)$
26:             while $nKnap < m$ do
27:                 $availableItems = GetAvailableItems(currentSolution)$
28:                 $SolutionQKP = Quadknap(availableItems, C)$
29:                 $currentSolution.add(SolutionQKP)$
30:                 $nknap = nknap + 1$
31:             end while
32:             $LocalSearchSol = FeaLS(currentSolution)$
33:             $InfLocalSearchSol = InfLS(LocalSearchSol)$
34:             if $f(bestSolution) < f(InfLocalSearchSol)$ then
35:                 $bestSolution = InfLocalSearchSol$
36:             end if
37:             $it = it + 1$
38:         end if
39:     end while
40: end for
41: return $bestSolution$
```

The first phase of the matheuristic consists of selecting the QKP solutions that can be part of the final QMKP solution. Lines 1-3 of Algorithm 15 deletes the QKP solutions that are not "promising". First, *selectThreshold* is executed and returns a set $A^* = \{A^*_i | A^*_i \in solQKP \wedge UB_i <= UBM \,; \forall i \in \{1, 2..t\}\}$ and also a value $s^* = |A^*|$. Then, *DeleteRedundanceSolutions* deletes identical solutions from the subset of solutions in A^*, returning the set of solutions A and $s = |A|$. Finally, *GetNumberOfElements* returns a vector $b = \{b_i | b_i = \sum_{j=1}^{n} A_{i,j}; \forall i \in \{1, 2..s\}\}$.

The matheuristic explores various combinations of the matrix A and generates a feasible solution for the QMKP using hybrid approaches. Lines 5-40 define the main loop based on the number of knapsacks m. The value of r represents the maximum number of QKP solutions chosen by the matheuristic. Then, a vector g_i is defined using the number of selected items (b_i) and the parameter δ.

The matheuristic iteratively solves an IP model in order to identify the best QKP solutions that are part of the final QMKP solution. The model solves the Cardinality Constrained Set Packing Problem (CCSP, line 16) described in the following:

$$\max \quad \sum_{h=1}^{s} g_h y_h \tag{3.59}$$

$$\text{s.t.} \quad \sum_{h=1}^{s} A_{hi} y_h \leq 1 \qquad (i \in N) \tag{3.60}$$

$$\sum_{h=1}^{s} y_i \leq r \tag{3.61}$$

$$y_h \in \{0,1\}. \qquad h = 1, ..., s \tag{3.62}$$

For the h-th QKP solution $(h = 1, ..., s)$, the binary decision variable y_h takes the value 1 iff the solution is selected, and g_h represents the corresponding "gain". The objective function (3.59) maximizes the global "gain" of the selected solutions (i.e., the global number of items). The constraints (3.60) forbid each selected item to be repeated among the solutions (no overlap). Finally, the constraint (3.61) ensures that the model selects no more than r solutions (with $r \leq m$). In this way, the *SolveCCSP* function returns the (at most) r selected QKP solutions belonging to the QMKP solution.

A list YY of "tabu selections" is defined. Line 13 initializes the YY list. If the model generates a selection that has already been made (i.e., it is found in the tabu list), the external while-loop (lines 15-39) ends and the main loop continues (lines 5-40). The vector g is also updated allowing the model to explore new selections.

The selection of the QKP solutions performed by the model may be incomplete, since the model can choose at most r QKP solutions for the QMKP. Lines 26-31 fill the remaining knapsacks using the available items (returned on line 27) using for each knapsack the *quadknap* algorithm.

Improvement algorithms are applied until a local optimum is found. First, the *FeaLS* algorithm (Algorithm 12), and then $InfLS$ algorithm (Algorithm 14) are applied. Each local search process does not use the input tabu list; it only uses the local tabu list defined internally in each algorithm. Also, consider that the knapsack selection process for the infeasible relocation phase is carried out by examining all the knapsacks and selecting the move that generates the largest profit. Finally, the best solution found is returned.

A variation of the previous matheuristic (Matheuristic 2) is described in Algorithm 16. In this second matheuristic, the vector g is defined by considering the original profit of each QKP solution. In this way, line 2 calculates the profit for each QKP solution from the matrix A. The following steps are the same as those of Matheuristic 1, but in this case, there is no inner loop that updates the vector g.

Algorithm 16 Matheuristic 2

```
Input: $solQKP, UB, UBM$
Output: $bestSolution$
$(A^*, s^*) = selectThreshold(solQKP, UB, UBM)$
$(A, s) = DeleteRedundanceSolutions(A^*, s^*)$
$g = GetSingleProfit(A)$
$bestSolution = \emptyset$
for $r = 1, 2..m$ do
    if $r > s$ then
        $break$
    end if
    $Y = SolveCCSP(A, g, r)$
    $currentSolution = GetSolutions(A, Y)$
    $nKnap = GetNomberOfKnapsack(currentSolution)$
    while $nKnap < m$ do
        $availableItems = GetAvailableItems(currentSolution)$
        $SolutionQKP = Quadknap(availableItems, C)$
        $currentSolution.add(SolutionQKP)$
        $nknap = nknap + 1$
    end while
    $LocalSearchSol = FeaLS(currentSolution)$
    $InfLocalSearchSol = InfLS(LocalSearchSol)$
    if $f(bestSolution) < f(InfLocalSearchSol)$ then
        $bestSolution = InfLocalSearchSol$
    end if
end for
return $bestSolution$
```

3.7 Computational experiments

3.7.1 Formulations and the Relaxations

The formulations and the relaxations introduced in the previous sections were implemented in C++ language. In the present section, we report the outcome of computational experiments aimed at evaluating the quality of the upper bounds produced by the polynomial-size models and the relaxations we have introduced. All the experiments were performed on a single thread of an AMD Ryzen 7 2700X Eight-Core Processor running at 3.7 GHz with 64 GB RAM. In order to evaluate our models and relaxations, we used benchmark instances adopted by [101], for most of which his Branch-and-Price algorithm could find the optimal solution (available online, see below). For the sake of completeness, in the next section we describe the way in which the instances were generated. The solution of our mathematical models was obtained using different codes:

- the general purpose solver CPLEX 12.10;
- the open source C code quadknap, that implements the algorithm for the QKP developed by [109] and is available at the home page of D. Pisinger, http://hjemmesider.diku.dk/~pisinger/codes.html . This code works with integer parameters and non-negative pairwise profits p_{ij}: in Section 3.7.1 we detail how we handled this feature to solve Lagrangian subproblems;
- the open source Fortran code MT1R, that implements a variant of the KP algorithm MT1 by [79] (adapted to non-integer parameters), available at the home page of S. Martello, http://www.or.deis.unibo.it/knapsack.html .

Benchmark instances

[101] presented two sets of random instances, called HJ and SS, based on the generation schemes proposed, respectively, by [89] for the QMKP and by [115] for a generalization of the problem. However, as reported by [95], the known optimality gap for even the easiest of the HJ instances ($n = 100$) is enormous, so smaller instances were generated by [101] (with $n \in \{20, 25, 30, 35\}$) to test his exact approach. For our experiments, we considered the HJ instances, both because they have been specifically designed for the QMKP and because all the involved pairwise profits are non-negative.

All the instances can be downloaded from the INFORMS page as a zipfile at the address https://pubsonline.informs.org/doi/suppl/10.1287/ijoc.2018.0840/suppl_file/ijoc.2018.0840-instances.sm2.zip. The knapsacks have a common integer capacity C, with n ranging in $\{20, 25, 30, 35\}$ and m in $\{3, 5, 10\}$. Three different values $d \in (0, 1]$ were used for the density of the non-zero quadratic terms: $d \in \{0.25, 0.50, 0.75\}$. For each triple (n, m, d), 5 random instances were produced as follows. Linear profits p_i were generated as uniformly random integers from $[0, 100]$. For every pair $i, j \in N$, quadratic profits p_{ij} were set with probability d to a random integer value uniformly drawn from $[0, 100]$, and to 0 with probability $1 - d$.

The weights w_i were generated as uniformly random integers from $[1, 50]$, while the capacities C were all set to $\lfloor 0.8 \sum_{i \in N} w_i / m \rfloor$. In total, 180 instances were thus generated.

In addition, in order to analyze how the most promising reformulations and relaxations scale for larger values of n, we generated new HJ instances, using the instance generator provided by [101], that is available for download. The generator produces instances according to the scheme described above. In this case, we considered instances with n ranging in $\{40, 45, 50, 55, 60\}$, m in $\{3, 5, 10\}$, and $d \in \{0.25, 0.50, 0.75\}$.

Experiments

We first evaluate the polynomial-size formulations discussed in Sections 3.1 and 3.2, for what concerns both their performance on the computation of the optimal solution and the quality of the LP relaxation of the linear ones. Table 3.1 reports on the different formulations of the QMKP, when solved through CPLEX, with one hour time limit. The six groups, of three columns each, refer to the models we have obtained for the QMKP:

- CPLEX-QF: 0-1 quadratic formulation (Section 3.1);
- CPLEX-FGW: 0-1 linear formulation by Fortet, Glover, and Woolsey (Section 3.2);
- CPLEX-GLOV: mixed-integer linear formulation by Glover (Section 3.2);
- CPLEX-RLT1: Level 1 reformulation linearization by Sherali and Adams (Section 3.2.2);
- CPLEX-DRLT1: decomposable Level 1 reformulation (Section 3.1).
- CPLEX-MDRLT1: modified decomposable Level 1 reformulation (Section 3.1).

TABLE 3.1: CPLEX solution of the polynomial formulations within one hour. Average percentage optimality gap, number of instances solved to proven optimality, average number of nodes and CPU time over five instances. Time limit: 1 hour.

instance			CPLEX-QF			CPLEX-FGW			CPLEX-GLOV			CPLEX-RLT1			CPLEX-DRLT1			CPLEX-MDRLT1		
n	*m*	*d*	%gap (#)	nodes	t(s)	%gap (#)	nodes	t(s)	%gap (#)	nodes	t(s)	%gap (#)	nodes	t(s)	%gap (#)	nodes	t(s)	%gap (#)	nodes	t(s)
20	3	0.25	0.00 (5)	712	0.2	0.00 (5)	692	1.7	0.00 (5)	2091	0.8	0.00 (5)	209	4.8	0.00 (5)	411	0.2	0.00 (5)	448	0.2
20	5	0.25	0.00 (5)	1282	1.1	0.00 (5)	1698	9.5	0.00 (5)	4188	3.3	0.00 (5)	73	4.1	0.00 (5)	380	0.7	0.00 (5)	1877	2.5
20	10	0.25	0.00 (5)	58	0.2	0.00 (5)	43	1.2	0.00 (5)	127	0.2	0.00 (5)	3	0.4	0.00 (5)	12	0.0	0.00 (5)	682	0.3
20	3	0.50	0.00 (5)	6762	4.7	0.00 (5)	12958	42.5	0.00 (5)	9893	5.9	0.00 (5)	413	6.0	0.00 (5)	1062	1.6	0.00 (5)	1475	1.8
20	5	0.50	0.00 (5)	9263	13.5	0.00 (5)	43682	253.8	0.00 (5)	33478	25.6	0.00 (5)	123	4.9	0.00 (5)	453	1.4	0.00 (5)	1936	4.4
20	10	0.50	0.00 (5)	112	0.5	0.00 (5)	80	2.2	0.00 (5)	440	0.4	0.00 (5)	5	1.6	0.00 (5)	15	0.3	0.00 (5)	217	0.3
20	3	0.75	0.00 (5)	23199	25.4	0.00 (5)	32420	88.2	0.00 (5)	55320	28.2	0.00 (5)	895	10.0	0.00 (5)	2333	6.1	0.00 (5)	4230	5.3
20	5	0.75	0.00 (5)	76629	157.5	0.00 (5)	30042	127.2	0.00 (5)	242127	233.9	0.00 (5)	138	4.7	0.00 (5)	661	3.2	0.00 (5)	1175	3.8
20	10	0.75	0.00 (5)	229	1.1	0.00 (5)	199	3.3	0.00 (5)	1995	2.5	0.00 (5)	9	1.7	0.00 (5)	10	0.5	0.00 (5)	47	0.2
Avg(#tot)			0.00 (45)	13138	22.7	0.00 (45)	13535	58.8	0.00 (45)	38851	33.4	0.00 (45)	208	4.2	0.00 (45)	593	1.6	0.00 (45)	1343	2.1
25	3	0.25	0.00 (5)	2898	1.5	0.00 (5)	3183	18.2	0.00 (5)	5416	2.4	0.00 (5)	1328	32.7	0.00 (5)	2805	1.9	0.00 (5)	2346	1.5
25	5	0.25	0.00 (5)	5243	6.4	1.12 (4)	69566	1028.8	0.00 (5)	13960	14.1	0.00 (5)	790	44.4	0.00 (5)	2727	6.7	0.00 (5)	7103	12.0
25	10	0.25	0.00 (5)	900	2.0	0.00 (5)	812	14.7	0.00 (5)	2938	4.9	0.00 (5)	78	8.0	0.00 (5)	190	0.8	0.00 (5)	558918	641.9
25	3	0.50	0.00 (5)	39882	37.8	0.00 (5)	62716	320.6	0.00 (5)	161340	168.8	0.00 (5)	3480	84.9	0.00 (5)	10403	26.6	0.00 (5)	8912	19.2
25	5	0.50	0.00 (5)	286271	813.6	24.71 (1)	210054	3540.8	0.00 (5)	1966676	2676.9	0.00 (5)	1486	47.7	0.00 (5)	4907	39.3	0.00 (5)	14344	65.1
25	10	0.50	0.00 (5)	1776	6.9	0.00 (5)	41219	509.4	0.00 (5)	17082	39.6	0.00 (5)	35	8.9	0.00 (5)	83	1.7	0.00 (5)	4956	17.9
25	3	0.75	0.00 (5)	553050	1157.6	4.07 (3)	343601	2256.5	5.39 (2)	1671387	2430.5	0.00 (5)	5507	154.8	0.00 (5)	24292	163.4	0.00 (5)	18290	60.1
25	5	0.75	14.16 (2)	735365	3117.2	47.11 (0)	253386	3600.0	31.83 (0)	1417607	3600.0	0.00 (5)	4148	167.9	0.00 (5)	24286	295.2	0.00 (5)	32127	201.2
25	10	0.75	0.00 (5)	10690	56.1	10.05 (1)	189176	2907.7	1.78 (4)	323526	965.3	0.00 (5)	63	9.9	0.00 (5)	279	4.1	0.00 (5)	11074	61.4
Avg(#tot)			1.57 (42)	181786	577.7	9.67 (29)	130412	1577.4	4.33 (36)	619992	1100.3	0.00 (45)	1879	62.1	0.00 (45)	7775	60.0	0.00 (45)	73119	120.0
30	3	0.25	0.00 (5)	6935	5.6	0.00 (5)	12680	116.0	0.00 (5)	45366	39.0	0.00 (5)	2546	165.4	0.00 (5)	6301	6.5	0.00 (5)	5057	5.4
30	5	0.25	0.00 (5)	78649	162.3	16.36 (1)	142000	2988.1	0.00 (5)	548528	991.7	0.59 (4)	11848	1046.0	0.00 (5)	17744	69.6	0.00 (5)	33281	90.7
30	10	0.25	1.14 (4)	162474	745.5	9.03 (3)	39804	1637.6	0.00 (5)	67015	223.2	0.00 (5)	242	29.9	0.00 (5)	675	4.9	0.63 (4)	225539	1226.4
30	3	0.50	0.00 (5)	599912	1409.6	7.20 (2)	341303	3134.3	1.71 (4)	1706468	2267.9	0.00 (5)	12057	577.8	0.00 (5)	126844	464.4	0.00 (5)	73760	254.5
30	5	0.50	21.42 (0)	785381	3600.0	69.75 (0)	158100	3600.0	26.73 (0)	1294407	3600.0	3.63 (1)	32286	2949.1	0.00 (5)	70244	845.2	0.00 (5)	95814	776.9
30	10	0.50	20.04 (1)	360085	3013.4	50.21 (0)	80619	3600.0	15.28 (0)	711346	3600.0	0.00 (5)	612	64.2	0.00 (5)	1620	35.4	0.00 (5)	11446	185.5
30	3	0.75	20.45 (0)	1043048	3600.0	31.10 (0)	425252	3600.0	22.80 (0)	1845088	3600.0	0.00 (5)	14281	789.1	1.17 (3)	127112	2061.4	0.00 (5)	135415	816.1
30	5	0.75	65.62 (0)	419412	3600.0	84.78 (0)	207738	3600.0	51.48 (0)	922215	3600.0	0.00 (5)	10027	884.7	0.00 (5)	46904	1038.0	0.00 (5)	56323	657.6
30	10	0.75	52.40 (1)	204383	3050.9	77.71 (0)	73858	3600.0	61.36 (0)	622603	3600.0	0.00 (5)	230	39.4	0.00 (5)	7222	196.4	0.00 (5)	4525	148.5
Avg(#tot)			20.12 (21)	406698	2131.9	38.46 (11)	164595	2875.1	19.93 (19)	862560	2391.3	0.47 (40)	9348	727.3	0.13 (43)	44963	524.7	0.07 (44)	71240	462.4
35	3	0.25	0.00 (5)	49383	48.2	0.17 (4)	82912	1322.0	0.00 (5)	195385	282.0	0.00 (5)	8045	815.9	0.00 (5)	41148	51.4	0.00 (5)	30827	41.4
35	5	0.25	1.50 (4)	638998	2124.0	21.63 (0)	104089	3600.0	7.80 (0)	1181369	3600.0	0.00 (5)	12511	2029.2	0.39 (4)	206468	1344.8	0.00 (5)	241679	1109.0
35	10	0.25	4.64 (3)	223228	1771.7	42.40 (0)	39541	3600.0	8.83 (2)	456966	2882.6	0.00 (5)	970	201.3	0.00 (5)	6786	96.8	1.63 (3)	116274	2015.3
35	3	0.50	10.02 (1)	1641493	3255.5	26.09 (0)	224323	3600.0	15.65 (0)	1239528	3600.0	2.45 (2)	29491	2816.6	5.19 (1)	542010	3358.3	2.86 (2)	489435	2924.5
35	5	0.50	49.95 (0)	523292	3600.0	78.72 (0)	111527	3600.0	46.77 (0)	703265	3600.0	10.61 (0)	21284	3600.0	8.96 (0)	168929	3600.0	7.90 (0)	229339	3600.0
35	10	0.50	32.74 (0)	211563	3600.0	105.18 (0)	35657	3600.0	65.90 (0)	379695	3600.0	0.00 (5)	2946	713.2	1.59 (2)	46683	2283.3	0.67 (4)	47960	2161.0
35	3	0.75	30.51 (0)	927451	3600.0	40.86 (0)	237571	3600.0	25.66 (0)	1128057	3600.0	0.00 (4)	21997	1980.4	5.06 (1)	187245	3081.6	1.97 (2)	197177	2437.0
35	5	0.75	86.50 (0)	277265	3600.0	108.75 (0)	111032	3600.0	60.77 (0)	625095	3600.0	4.90 (0)	15068	3600.0	6.24 (0)	98240	3600.0	5.19 (0)	139215	3600.0
35	10	0.75	135.39 (0)	112848	3600.0	155.78 (0)	34545	3600.0	119.48 (0)	319078	3600.0	1.43 (3)	7874	2011.1	5.33 (1)	29577	2965.4	1.90 (2)	43900	3126.7
Avg(#tot)			39.03 (13)	459759	2799.9	64.40 (4)	92953	3346.9	38.99 (7)	637548	3151.6	2.15 (29)	12274	1974.2	3.64 (19)	142480	2264.6	2.46 (23)	167329	2335.0
Ov.Avg(#tot)			15.18(121)	278337	1383.1	28.13 (89)	104391	1964.6	15.81(107)	553363	1669.2	0.66(159)	6197	692.0	0.94(152)	50196	712.7	0.63(157)	79087	729.9

Each line refers to a triple (n, m, d). For each formulation, the three entries in the table report (over the corresponding 5 instances),

- `%gap` = average percentage optimality gap of the best solution value z obtained by CPLEX within one CPU hour with respect to its best found upper bound u, computed as $100\,(u - z)/z$. In parentheses #, total number of instances solved to proven optimality;
- `nodes` = average number of nodes of the CPLEX branch-decision tree;
- `t(s)` = average CPU time expressed in seconds.

The average values of `%gap`, `nodes`, `t(s)` and the total value of # for each value of n (45 instances) are also reported, as well as the overall values over the 180 instances.

The table shows that the direct use of the models to provide solutions to the QMKP through a general purpose solver like CPLEX (we also tried Gurobi 9, with similar results) can only be effective for small size instances. We observe that the quadratic formulation QF and the two linear formulations FGW and GLOV obtain worse results than those obtained by the three Level 1 RLT formulations.

For $n \leq 25$, RLT1, DRLT1, and MDRLT1 could solve all 90 instances to optimality. DRLT1 turned out to be the fastest method, although it requires a higher number of CPLEX decision nodes than RLT1. For $n = 30$, the same models could solve, respectively, 40, 43, and 44 instances out of 45, with CPU times of few hundred seconds. The models look instead inadequate for instances with $n = 35$.

As previously mentioned, all the considered instances but two were efficiently solved to optimality by [101] Branch-and-Price algorithm, referred to as BBP in the following. (He used Gurobi 7.5.1 with one hour time limit on a computer similar to ours, namely an Intel Core i7-4770 running at 3.40 GHz with 32 GB RAM.) Although a direct comparison between the CPLEX solution of polynomial-size models and a specialized Branch-and-Price algorithm may be questionable, we can observe that the Level 1 RLT models appear to perform better for $n = 20$ and $n = 25$, while BBP is more effective for $n = 30$, and much better for $n = 35$. More specifically,

- for $n = 20$, the three Level 1 reformulations and BBP solved all instances, with DRLT1 and MDRLT1 taking smaller times (on average, 1.6 and 2.1 seconds, respectively, versus 4.2 seconds of RLT1 and 3.6 seconds of BBP);
- for $n = 25$, RLT1 and DRLT1 solved all 45 instances (with average times 62.1 and 60.0 seconds, respectively) while BBP solved one instance less with average time 95.8 seconds. MDRLT1 solved all instances, but required a much higher, anomalous, time;
- for $n = 30$, MDRLT1 solved 44 instances with an average time of 462.4 seconds, while BBP solved all 45 instances with an average time of 151.2 seconds;
- for $n = 35$, RLT1 solved 29 instances with an average time of 1974.2 seconds, while BBP solved 44 instances with an average time of 455.2 seconds.

Table 3.2 examines the quality of the upper bounds computed through the LP relaxations of the linear models considered in Table 3.1. The five groups, of two columns each, refer to:

TABLE 3.2: Upper bounds computed through LP relaxation of the linear formulations. Average percentage optimality gap and CPU time over 5 instances. Time limit: 1 hour.

instance			LP-FGW		LP-GLOV		LP-RLT1		LP-DRLT1		LP-MDRLT1	
n	*m*	*d*	%gap	t(s)	%gap	t(s)	%gap	t(s)	%gap	t(s)	%gap	t(s)
20	3	0.25	37.75	0.01	39.21	0.00	29.83	0.01	29.83	0.00	29.83	0.00
20	5	0.25	64.50	0.01	66.31	0.00	29.47	0.01	29.48	0.00	29.30	0.00
20	10	0.25	163.12	0.01	166.21	0.00	19.31	0.02	19.32	0.01	11.94	0.00
20	3	0.50	68.66	0.01	70.94	0.00	31.59	0.01	31.63	0.00	31.63	0.00
20	5	0.50	126.35	0.01	129.41	0.00	29.31	0.01	29.32	0.01	28.05	0.00
20	10	0.50	265.74	0.01	270.76	0.01	24.72	0.02	24.72	0.01	16.72	0.00
20	3	0.75	90.80	0.01	95.75	0.00	28.76	0.01	28.77	0.01	28.77	0.00
20	5	0.75	156.59	0.01	163.41	0.00	24.26	0.02	24.26	0.01	23.67	0.01
20	10	0.75	310.87	0.01	322.10	0.01	19.36	0.02	19.37	0.01	12.06	0.01
	Avg		142.71	0.01	147.12	0.00	26.29	0.01	26.30	0.01	23.55	0.00
25	3	0.25	41.19	0.01	42.97	0.00	36.28	0.02	36.28	0.00	36.28	0.00
25	5	0.25	65.55	0.01	67.69	0.00	33.27	0.02	33.27	0.01	33.27	0.00
25	10	0.25	138.13	0.02	141.35	0.01	24.82	0.04	24.85	0.01	19.86	0.01
25	3	0.50	78.55	0.01	81.90	0.00	40.42	0.02	40.49	0.01	40.49	0.01
25	5	0.50	132.64	0.01	137.10	0.01	33.33	0.03	33.34	0.01	33.34	0.01
25	10	0.50	263.46	0.02	270.41	0.01	27.69	0.04	27.74	0.02	20.17	0.01
25	3	0.75	93.43	0.01	97.98	0.00	34.57	0.02	34.59	0.01	34.59	0.01
25	5	0.75	163.96	0.01	170.19	0.01	30.60	0.03	30.60	0.01	30.60	0.01
25	10	0.75	310.82	0.02	320.75	0.01	23.95	0.05	23.98	0.02	20.03	0.01
	Avg		143.08	0.01	147.81	0.00	31.66	0.03	31.68	0.01	29.85	0.01
30	3	0.25	43.53	0.01	45.56	0.00	39.37	0.02	39.37	0.01	39.37	0.00
30	5	0.25	69.84	0.02	72.27	0.00	39.87	0.03	39.91	0.01	39.91	0.01
30	10	0.25	135.77	0.03	139.28	0.01	28.45	0.07	28.53	0.02	24.19	0.01
30	3	0.50	77.83	0.01	82.43	0.00	47.04	0.03	47.10	0.01	47.10	0.01
30	5	0.50	125.58	0.02	131.49	0.01	37.88	0.04	37.88	0.01	37.88	0.01
30	10	0.50	250.12	0.03	259.62	0.01	29.97	0.07	30.01	0.02	26.12	0.02
30	3	0.75	104.83	0.02	110.95	0.00	36.65	0.04	36.65	0.01	36.65	0.01
30	5	0.75	175.09	0.02	183.45	0.01	28.54	0.05	28.54	0.02	28.54	0.01
30	10	0.75	348.85	0.04	362.68	0.01	26.20	0.08	26.20	0.03	20.71	0.02
	Avg		147.94	0.02	154.19	0.01	34.89	0.05	34.91	0.02	33.39	0.01
35	3	0.25	50.01	0.02	53.32	0.00	43.19	0.04	43.19	0.01	43.19	0.01
35	5	0.25	77.38	0.02	81.28	0.01	44.89	0.05	44.90	0.02	44.90	0.01
35	10	0.25	137.54	0.04	142.89	0.01	31.79	0.09	31.79	0.03	29.81	0.01
35	3	0.50	82.83	0.02	87.35	0.00	49.75	0.04	49.79	0.02	49.79	0.01
35	5	0.50	135.69	0.03	141.55	0.01	42.82	0.05	42.85	0.02	42.85	0.01
35	10	0.50	248.47	0.05	257.14	0.01	30.51	0.09	30.51	0.03	28.35	0.02
35	3	0.75	101.86	0.02	108.36	0.01	36.15	0.06	36.16	0.02	36.16	0.02
35	5	0.75	178.30	0.03	187.29	0.01	32.10	0.07	32.10	0.02	32.10	0.02
35	10	0.75	345.25	0.05	360.17	0.01	24.95	0.11	24.95	0.04	23.37	0.04
	Avg		150.81	0.03	157.70	0.01	37.35	0.07	37.36	0.02	36.72	0.02
	Ov.Avg		146.14	0.02	151.71	0.01	32.55	0.04	32.56	0.01	30.88	0.01

- LP-FGW: LP relaxation of FGW (Section 3.2.1);
- LP-GLOV: LP relaxation of GLOV (Section 3.2.1);
- LP-RLT1: LP relaxation of RLT1 (Section 3.2.2);
- LP-DRLT1: LP relaxation of DRLT1 (Section 3.1);
- LP-MDRLT1: LP relaxation of MDRLT1 (Section 3.1);

The LP relaxations were solved through CPLEX. Each line refers to a triple (n, m, d). For each formulation, the two entries in the table report the values (over the corresponding 5 instances) of:

- `%gap` = average percentage gap of the upper bound u obtained within one CPU hour with respect to the best known solution value z, computed as $100\,(u - z)/z$. The value of z is optimal for 179 instances out of 180: 178 were provided by Bergman [101], one more was found by the Level 1 RLT models (see the comments on Table 3.1);

- `t(s)` = average CPU time expressed in seconds.

We have seen in Table 3.1 that the linear models (DRLT1 in particular) can provide good solutions for instances of limited size. Table 3.2 shows that the CPU times for computing their LP relaxations are very small, but the quality of the upper bounds they provide is poor, especially for what concerns GLOV and FGW. The performances of RLT1 and DRLT1 are very similar to each other. Although the continuous relaxation of DRLT1 is weaker than that of RLT1 (as observed in Section 3.1), the quality of the bounds they produce is practically the same, while DRLT1 is faster. The best performance was obtained by MDRLT1. In particular: (i) for $m = 10$, MDRLT1 produced the smallest percentage gaps, thanks to the addition of constraints (3.45); (ii) MDRLT1 was slightly faster than DRLT1, probably due to the use of sets R_{ik} and S_{ik}.

In any case, the results of Table 3.2 indicate that the LP relaxations are inadequate to be embedded in an enumerative approach. We next show that much better results can be obtained from Lagrangian relaxations.

In Table 3.3 we analyze the quality of the upper bounds obtained by the surrogate and Lagrangian relaxations studied in Sections 3.3-3.4. For the surrogate relaxations, the optimal multipliers are known (see Proposition 4). For the Lagrangian relaxations, the search of the best multipliers was always performed via the proximal bundle method [114]. The columns provide information on the different ways we solved the relaxed subproblems (either CPLEX, or Quadknap [109], or MT1R [79]). We also consider both the case where separability due to equal capacities is exploited (see Section 3.4.1) and where it is not. The eight groups, of two columns each, refer to:

- `Srg CPLEX`: surrogate bound $S(\pi)$ (Section 3.3) solved through CPLEX;

- `Srg Qknap`: surrogate bound $S(\pi)$ (Section 3.3) solved through `quadknap` [109];

- `Lgr QP CPLEX`: Lagrangian bound $L^Q(\lambda)$ (Section 3.4.1) solved through CPLEX;

- `S-Lgr QP CPLEX`: Lagrangian bound $L^Q(\lambda)$ exploiting separability, with the single QKP solved through CPLEX;

- `S-Lgr QP Qknap`: Lagrangian bound $L^Q(\lambda)$ exploiting separability, with the single QKP solved through `quadknap` [109]. Note that `quadknap` works with non-negative pairwise profits p_{ij} (which holds in our formulation) and integer coefficients. Since our Lagrangian linear profits $p_i - \lambda_i$ $(i \in N)$ can assume non-integer values, we multiplied all profits by 100, rounded each resulting

TABLE 3.3: Upper bounds computed through surrogate and Lagrangian relaxations. Average percentage gap and CPU time over 5 instances. Time limit: 1 hour.

instance			Srg CPLEX		Srg Qknap		Lgr QP CPLEX		S-Lgr QP CPLEX		S-Lgr QP Qknap		S-Lgr QPL CPLEX		D-Lgr DRLT1 CPLEX		D-Lgr DRLT1 MT1R	
n	*m*	*d*	%gap	t(s)	%gap	t(s)	%gap	t(s)	%gap	t(s)	%gap	t(s)	%gap	t(s)	%gap	t(s)	%gap	t(s)
20	3	25	34.45	0.01	34.45	0.00	0.24	1.37	0.24	0.71	0.24	0.38	0.24	0.64	26.04	5.29	26.04	1.58
20	5	25	59.80	0.01	59.80	0.00	0.09	2.11	0.09	1.12	0.09	0.45	0.09	1.05	18.90	13.09	18.90	2.46
20	10	25	156.18	0.01	156.18	0.00	0.00	1.78	0.00	0.81	0.00	0.43	0.00	0.74	3.89	4.35	3.89	1.50
20	3	50	62.53	0.03	62.53	0.00	0.44	5.80	0.44	1.73	0.44	0.45	0.44	1.81	25.24	33.06	25.24	6.67
20	5	50	119.77	0.02	119.77	0.00	0.02	13.90	0.02	1.73	0.02	0.46	0.02	1.77	18.18	34.55	18.18	7.27
20	10	50	253.45	0.02	253.45	0.00	0.47	2.43	0.47	1.24	0.47	0.40	0.47	0.90	8.74	16.02	8.74	2.61
20	3	75	85.14	0.04	85.14	0.00	1.07	7.20	1.07	3.14	1.07	0.49	1.07	2.86	22.51	48.52	22.51	9.16
20	5	75	149.76	0.02	149.76	0.00	0.52	10.44	0.52	2.09	0.52	0.45	0.52	1.87	14.42	52.50	14.42	9.87
20	10	75	297.79	0.04	297.79	0.00	0.00	3.68	0.00	1.39	0.00	0.44	0.00	0.91	5.31	21.50	5.31	3.94
Avg			135.43	0.02	135.43	0.00	0.32	5.41	0.32	1.55	0.32	0.44	0.32	1.39	15.91	25.43	15.91	5.01
25	3	25	38.84	0.01	38.84	0.00	0.41	5.16	0.41	1.89	0.41	0.63	0.41	1.86	34.23	7.80	34.23	3.20
25	5	25	63.01	0.01	63.01	0.00	0.24	7.91	0.24	1.62	0.24	0.73	0.24	1.68	23.59	35.99	23.59	7.24
25	10	25	134.29	0.01	134.29	0.00	0.19	4.49	0.19	1.42	0.19	0.66	0.19	1.15	10.66	20.27	10.66	4.00
25	3	50	74.39	0.03	74.39	0.00	0.45	12.67	0.45	6.58	0.45	0.81	0.45	7.18	35.72	52.57	35.71	14.37
25	5	50	127.88	0.02	127.88	0.00	0.31	12.38	0.31	3.51	0.31	0.71	0.31	3.60	25.06	85.20	25.08	20.13
25	10	50	256.45	0.02	256.45	0.00	0.00	11.24	0.00	1.75	0.00	0.73	0.00	1.23	10.98	60.98	10.99	17.12
25	3	75	87.19	0.07	87.19	0.00	0.46	54.57	0.46	42.07	0.46	0.88	0.46	21.49	30.31	80.56	30.31	22.51
25	5	75	155.37	0.06	155.37	0.00	0.86	22.97	0.86	7.86	0.86	0.70	0.86	6.98	22.93	122.87	22.93	31.31
25	10	75	298.84	0.05	298.84	0.00	0.28	42.17	0.28	1.93	0.28	0.78	0.28	1.61	11.07	91.47	11.12	31.12
Avg			137.36	0.03	137.36	0.00	0.36	19.29	0.36	7.63	0.36	0.74	0.36	5.20	22.73	61.97	22.74	16.78
30	3	25	41.91	0.03	41.91	0.00	0.19	12.16	0.19	4.13	0.19	1.23	0.19	4.50	37.40	16.90	37.40	7.46
30	5	25	68.10	0.04	68.10	0.00	0.20	20.86	0.20	3.31	0.20	1.01	0.20	3.49	33.05	55.19	33.04	15.61
30	10	25	133.05	0.04	133.05	0.00	0.16	20.25	0.16	1.75	0.16	1.01	0.16	1.53	14.09	65.79	14.10	17.79
30	3	50	73.90	0.04	73.90	0.01	0.73	64.11	0.73	50.42	0.74	1.75	0.73	39.86	43.55	83.00	43.56	30.48
30	5	50	120.64	0.05	120.64	0.00	0.04	27.11	0.04	11.32	0.04	1.17	0.04	10.93	31.42	163.19	31.38	60.97
30	10	50	242.59	0.05	242.59	0.00	0.05	47.13	0.05	3.28	0.06	1.05	0.05	2.70	17.37	179.19	17.30	87.92
30	3	75	98.86	0.12	98.86	0.01	1.07	496.64	1.07	456.88	1.11	1.96	1.07	138.98	33.45	143.83	33.44	53.25
30	5	75	167.51	0.08	167.51	0.00	0.25	74.82	0.25	58.34	0.26	1.05	0.25	17.79	23.53	223.65	23.53	79.20
30	10	75	337.06	0.09	337.06	0.01	0.23	32.14	0.23	4.56	0.23	0.84	0.23	3.09	13.53	221.99	13.56	104.73
Avg			142.63	0.06	142.63	0.00	0.32	88.36	0.32	66.00	0.33	1.23	0.32	24.76	27.49	128.08	27.48	50.82
35	3	25	47.94	0.09	47.94	0.01	0.51	20.84	0.51	10.30	0.51	3.30	0.51	11.41	41.86	32.58	41.86	17.53
35	5	25	75.42	0.06	75.42	0.01	0.23	23.08	0.23	7.07	0.24	1.40	0.23	8.05	39.50	87.92	39.50	31.58
35	10	25	135.16	0.03	135.16	0.01	0.16	54.57	0.16	2.86	0.16	1.11	0.16	2.93	18.14	144.62	18.13	57.52
35	3	50	81.15	0.04	81.15	0.01	0.49	377.56	0.49	342.35	0.49	4.61	0.49	261.28	46.65	143.11	46.66	67.96
35	5	50	133.57	0.04	133.57	0.01	0.36	149.04	0.36	119.59	0.36	1.59	0.36	63.66	37.17	266.52	37.16	125.57
35	10	50	245.30	0.04	245.30	0.01	0.15	47.62	0.15	6.74	0.15	1.02	0.15	5.73	21.11	331.66	21.10	167.77
35	3	75	98.13	0.16	98.13	0.01	0.54	2192.90	0.50	2023.31	0.50	3.97	0.50	381.58	33.66	268.22	33.66	137.15
35	5	75	173.40	0.14	173.40	0.01	0.30	792.43	0.30	689.64	0.31	1.93	0.30	134.80	28.10	401.31	28.07	182.57
35	10	75	335.82	0.13	335.82	0.01	0.36	103.01	0.36	21.78	0.36	1.25	0.36	13.03	17.41	495.15	17.24	293.53
Avg			147.32	0.08	147.32	0.01	0.35	417.89	0.34	358.18	0.34	2.24	0.34	98.05	31.51	241.23	31.49	120.13
Ov.Avg			140.68	0.05	140.68	0.00	0.34	132.74	0.33	108.34	0.34	1.16	0.33	32.35	24.41	114.18	24.40	48.18

value a to $\lceil a \rceil$, and correspondingly divided the solution value by 100 (thus obtaining a valid upper bound on the optimal QKP solution);

- `S-Lgr QPL CPLEX`: Lagrangian bound $L^Q(\lambda)$ exploiting separability, with the single QKP linearized through the MDRLT1 formulation with $m = 1$ and solved through CPLEX;
- `D-Lgr DRLT1 CPLEX`: Lagrangian bound $L^R(\lambda)$ exploiting the decomposable structure (see (i)-(ii) of Section 3.4.2) with single KPs solved through CPLEX;
- `D-Lgr DRLT1 MT1R`: Lagrangian bound $L^R(\lambda)$ exploiting the decomposable structure (see (i)-(ii) of Section 3.4.2) with single KPs solved through `MT1R` [79].

Preliminary computational experiments showed that the exact solution of the linear pseudo-MKP (point (ii) in Section 3.4.2), performed at each iteration of the bundle procedure, takes a large computing time, so we replaced it with its LP relaxation (solved through CPLEX). The entries in the table are the same as for Table 3.2. The results indicate that:

- despite using optimal multipliers, the surrogate relaxation is very weak: it takes very short CPU times, but the upper bounds are extremely loose;
- all Lagrangian relaxations provide much better bounds, although $L^R(\lambda)$ is considerably weaker than $L^Q(\lambda)$ (with `D-Lgr DRLT1 MT1R` requiring much smaller CPU times than `D-Lgr DRLT1 CPLEX`);
- all versions of $L^Q(\lambda)$ are by far the best approaches:
  - they produce an average gap of 0.34%, with individual gaps rarely exceeding 1%;
  - their gaps are identical, with the only exception of `S-Lgr QP Qknap` due to the non-optimal solution of the Lagrangian subproblems imposed by `quadknap`, which can result in a suboptimal Lagrangian dual (for 7 instances out of 180, its value is higher by one unit);
  - `Lgr QP CPLEX`, which does not exploit separability, has the highest CPU times;
  - the second highest times are those of `S-Lgr QP CPLEX`, which solves the single QKP Lagrangian subproblem through CPLEX on the standard quadratic formulation. By linearizing the QKP through the MDRLT1 formulation, `S-Lgr QPL CPLEX` reduces the computational effort by two thirds;
  - the best approach is by far `S-Lgr QP Qknap`, which directly solves the QKP through `quadknap`. It provides very tight upper bounds in short CPU times, with a much smaller growth rate with respect to n than that of the other $L^Q(\lambda)$ methods;
  - by comparing the obtained upper bounds with the optimal solution values, it turns out that they are frequently identical: it happens for 94 instances out of 180 (92 instances for `S-Lgr QP Qknap`).

Finally, Table 3.4 shows how the two best performing reformulations RLT1 and MDRLT1, as well as three of our relaxations scale for larger values of n, up to 60. The five groups, of two columns each, refer to:

TABLE 3.4: CPLEX solution and upper bounds for larger instances. Average percentage gap and CPU time over 5 instances. Time limit: 1 hour (3 hours for CPLEX runs).

instance			CPLEX-RLT1				CPLEX-MDRLT1				LP-MDRLT1		S-Lgr QP Qknap		D-Lgr DRLT1 MT1R	
n	m	d	%gap	%gapL	%gapU	t(s)	%gap	%gapL	%gapU	t(s)	%gap	t(s)	%gap	t(s)	%gap	t(s)
40	3	25	0.00	0.00	0.00	2476.2	0.00	0.00	0.00	191.6	45.26	0.01	0.11	11.47	44.23	34.24
40	5	25	4.72	0.27	4.43	9577.2	2.65	0.12	2.53	8368.1	48.91	0.01	0.55	1.81	44.14	71.77
40	10	25	5.49	0.29	5.18	9308.6	0.00	0.00	0.00	2010.2	35.23	0.02	0.43	0.59	25.22	156.48
40	3	50	2.18	0.01	2.17	9154.0	8.32	0.04	8.28	10559.8	54.21	0.02	1.05	25.30	51.79	129.77
40	5	50	11.12	2.24	8.67	10793.2	11.12	0.00	11.11	10800.0	46.99	0.02	1.03	3.39	42.73	247.10
40	10	50	9.55	0.22	9.31	10792.9	6.60	0.04	6.56	9335.4	32.12	0.03	0.83	0.71	24.86	428.60
40	3	75	1.71	0.05	1.65	9265.2	8.81	0.20	8.59	10795.0	40.76	0.03	0.45	26.12	38.88	233.18
40	5	75	11.13	0.66	10.41	10793.2	10.82	0.29	10.49	10794.2	37.36	0.03	1.76	3.79	33.84	310.38
40	10	75	8.23	1.31	6.84	10792.8	6.96	0.00	6.96	10794.8	25.19	0.05	0.70	0.67	19.53	434.95
Avg			6.01	0.56	5.41	9217.0	6.14	0.08	6.06	8183.2	40.67	0.02	0.77	8.21	36.14	227.39
45	3	25	3.91	0.62	3.25	8889.6	0.00	0.00	0.00	1548.8	50.35	0.01	0.23	83.92	49.53	61.97
45	5	25	11.78	1.89	9.69	10792.8	12.27	0.00	12.27	10800.0	55.36	0.01	0.68	10.12	51.30	138.89
45	10	25	14.03	2.39	11.35	10792.7	10.27	0.00	10.28	10800.0	38.67	0.02	0.45	0.91	29.74	342.53
45	3	50	7.26	0.81	6.39	10117.3	16.17	0.00	16.17	10800.0	56.59	0.03	1.25	151.94	54.57	236.79
45	5	50	17.64	2.77	14.45	10792.9	19.00	0.00	19.00	10800.0	50.84	0.03	1.59	12.01	47.34	413.74
45	10	50	15.30	2.62	12.37	10792.5	11.23	0.00	11.23	10800.0	35.22	0.04	1.31	1.27	29.29	701.21
45	3	75	4.91	0.02	4.89	9354.8	12.09	0.13	11.95	10800.0	42.57	0.04	0.98	125.74	41.00	395.29
45	5	75	15.42	2.63	12.43	10792.7	12.93	0.00	12.94	10800.0	38.83	0.04	1.84	12.48	36.21	533.15
45	10	75	13.81	2.02	11.55	10792.3	11.16	0.00	11.16	10800.0	30.06	0.06	1.72	1.48	24.64	854.63
Avg			11.56	1.75	9.60	10346.4	11.68	0.01	11.67	9772.1	44.28	0.03	1.12	44.43	40.40	408.69
50	3	25	10.05	1.98	7.91	10792.3	3.91	0.00	3.91	7977.8	53.29	0.01	0.57	702.10	52.61	172.38
50	5	25	18.44	2.56	15.52	10792.1	21.03	0.00	21.03	10800.0	62.60	0.02	2.00	44.53	59.12	411.04
50	10	25	19.87	2.02	17.51	10792.2	15.51	0.00	15.51	10800.0	46.75	0.03	1.48	2.61	37.67	977.58
50	3	50	13.94	1.81	11.92	10792.5	24.23	0.00	24.23	10800.0	59.71	0.04	1.24	1459.54	58.13	390.89
50	5	50	24.33	2.65	21.16	10792.4	28.86	0.12	28.70	10800.0	56.69	0.04	3.09	58.10	53.61	610.32
50	10	50	21.84	4.36	16.73	10792.3	17.82	0.00	17.82	10800.0	37.30	0.05	1.98	2.66	31.87	919.93
50	3	75	10.47	2.40	7.88	10792.4	17.20	0.00	17.20	10800.0	44.24	0.05	0.84	1134.35	42.95	873.14
50	5	75	16.60	2.52	13.70	10792.5	19.73	0.00	19.73	10800.0	42.59	0.06	2.59	76.35	40.38	1455.24
50	10	75	17.85	2.14	15.41	10792.6	17.10	0.35	16.69	10800.0	35.77	0.08	3.94	4.32	31.11	2317.03
Avg			17.04	2.49	14.19	10792.4	18.38	0.05	18.31	10486.4	48.77	0.04	1.97	387.17	45.27	903.06
55	3	25	12.91	2.13	10.54	10796.4	9.29	0.08	9.21	10800.0	57.31	0.01	0.79	2527.69	57.13	169.40
55	5	25	23.26	3.66	18.89	10795.7	28.92	0.00	28.92	10800.0	69.31	0.02	1.37	183.96	66.30	389.81
55	10	25	26.92	3.03	23.21	10793.7	23.66	0.00	23.66	10800.0	51.07	0.03	2.52	3.27	43.72	793.73
55	3	50	19.81	2.15	17.29	10796.4	35.11	0.11	34.96	10800.0	67.15	0.05	3.88	3659.88	65.70	656.78
55	5	50	31.03	4.54	25.34	10795.6	36.68	0.00	36.68	10800.0	62.47	0.05	3.52	334.96	59.98	1019.73
55	10	50	27.17	2.87	23.61	10793.6	25.52	0.26	25.20	10800.0	44.25	0.06	4.91	6.31	39.04	1422.40
55	3	75	13.53	2.55	10.70	10796.3	23.18	0.00	23.17	10800.0	46.56	0.07	1.15	3570.32	45.46	1049.39
55	5	75	16.38	1.67	14.44	10795.6	23.28	0.07	23.19	10800.0	42.76	0.07	2.26	207.63	40.89	1237.25
55	10	75	20.57	1.64	18.61	10793.5	21.29	0.23	21.02	10800.0	36.90	0.08	5.35	8.54	33.14	1581.45
Avg			21.29	2.69	18.07	10795.2	25.21	0.08	25.11	10800.0	53.09	0.05	2.86	1166.95	50.15	924.44
60	3	25	18.19	3.64	14.04	10795.7	18.67	0.00	18.67	10800.0	62.73	0.02	3.43	3682.17	62.54	415.61
60	5	25	33.09	7.33	23.99	10794.7	38.30	0.00	38.30	10800.0	76.75	0.03	1.73	1791.48	74.11	1056.09
60	10	25	38.17	7.80	28.16	10793.7	31.99	0.00	31.99	10800.0	58.47	0.05	3.29	12.37	51.91	2284.78
60	3	50	23.04	4.10	18.20	10795.8	37.20	0.00	37.20	10800.0	67.25	0.06	5.93	3841.32	66.00	1033.00
60	5	50	33.26	4.59	27.42	10794.7	42.16	0.28	41.76	10800.0	65.77	0.06	4.01	1259.54	63.47	1798.34
60	10	50	32.69	3.60	28.17	10793.5	31.13	0.01	31.11	10800.0	51.20	0.07	6.92	23.76	46.63	1795.76
60	3	75	15.06	3.60	11.09	10795.6	25.10	0.00	25.10	10800.0	46.46	0.10	2.94	3735.64	45.55	1990.74
60	5	75	20.42	3.16	16.80	10794.9	27.21	0.07	27.12	10800.0	44.87	0.09	3.48	1211.01	43.47	2031.27
60	10	75	20.93	1.34	19.32	10793.3	23.82	0.31	23.44	10800.0	38.25	0.13	6.66	19.91	35.09	2214.12
Avg			26.10	4.35	20.80	10794.7	30.62	0.07	30.52	10800.0	56.86	0.07	4.26	1730.80	54.31	1624.41
Ov.Avg			16.40	2.37	13.61	10389.1	18.41	0.06	18.33	10008.3	48.73	0.04	2.20	667.51	45.25	817.60

- CPLEX-RLT1: Level 1 reformulation linearization by Sherali and Adams (Section 3.2.2);
- CPLEX-MDRLT1: modified decomposable Level 1 reformulation (Section 3.1);
- LP-MDRLT1: LP relaxation of MDRLT1 (Section 3.1);
- S-Lgr QP Qknap: Lagrangian bound $L^Q(\lambda)$ exploiting separability, with the single QKP solved through quadknap [109];
- D-Lgr DRLT1 MT1R: Lagrangian bound $L^R(\lambda)$ exploiting the decomposable structure (see (i)-(ii) of Section 3.4.2) with single KPs solved through MT1R [79].

Since for most of these instances the optimal value is unknown, we ran CPLEX, both for RLT1 and MDRLT1, with a time limit of three hours (instead of one our, as for all other runs): the percentage gaps of the relaxations were thus computed with respect to the best solution value z obtained by the two CPLEX executions. The %gap and t(s) values have the same meaning as in the previous tables. For CPLEX-RLT1 and CPLEX-MDRLT1, two additional columns provide:

- %gapL = average percentage gap of the lower bound L obtained within three hours with respect to z, computed as $100\,(z-L)/L$;
- %gapU = average percentage gap of the upper bound U obtained within three hours with respect to z, computed as $100\,(U-z)/z$.

The table shows that the computing time for the three relaxations was generally below one hour, apart from a few cases that took slightly longer for the Lagrangian relaxation S-Lgr QP Qknap. The Lagrangian relaxation of the quadratic model still provides fairly good bounds, if compared to the others. Yet, the computing time of the bundle procedure may be rather large for $n > 50$. The bounds provided by D-Lgr DRLT1 MT1R are pretty close to those found by LP-MDRLT1 despite a considerably larger computational effort required by the bundle method. Considering the size of gaps for the LP relaxation, it comes as no surprise that CPLEX cannot find an optimal solution for most of the instances. The %gapL values confirm that the MDRLT1 reformulation leads to a better CPLEX performance.

We finally mention that it turned out to be impossible to execute the experiments with even larger instances, as already for $n = 60$ CPLEX required more than 30 GB of memory to solve an instance. Observe that, in any case, the %gap values in the table are still likely to overestimate the real gaps.

3.7.2 Multi-Start Iterated Local Search

General descriptions and Parameter Setting

TABLE 3.5: Best sets of parameters from Irace for MS-ILS

ID	$maxPl$	$maxIter$	$maxStart$	$r1$	$r2$	$r3$	λ	$maxTry$	$tabuTenure_{inf}$	M	$tabuTenure_{fea}$	β
1	60	90	20	0.03	0.19	0.36	0.8	150	100	80	80	0.97
2	60	90	20	0.07	0.1	0.19	0.79	200	100	80	60	0.92
3	60	90	20	0.04	0.09	0.14	0.72	200	100	80	60	0.85
4	60	90	20	0.04	0.05	0.09	0.62	200	100	80	60	0.81
5	60	90	20	0.03	0.28	0.38	0.81	200	100	80	80	0.99

All the experiments were performed on a single thread of an AMD Ryzen 7 2700X Eight-Core Processor running at 3.7 GHz with 64 GB RAM. The meta-heuristic was programmed in C ++ using the code *quadknap* that was programmed in C in [109]. Finally, we used the classic instances of the literature available in `http://cedric.cnam.fr/soutif/QKP/QKP.html` with size $n = \{100, 200\}$, $d = \{25, 75\}$ and $m = \{3, 5, 10\}$. Each group has five instances, globally obtaining 60 instances.

The MS-ILS was calibrated using Irace [78]. This framework iteratively and elitistically searches for the best parameters using a predefined range (or set) for each MS-ILS parameter. Each iteration of Irace updates the parameters, thus obtaining high quality parameters. An instance was used for each group (globally 12 instances). The details of the obtained parameters are shown in Table 3.5. The set of parameters corresponding to $ID = 1$ has been used since it is the one that empirically obtained the best results. The total duration of the calibration was 163944 seconds.

Comparison of results without time limit

The proposed algorithm is compared with four metaheuristics from the literature. The first approach is an Iterated Responsive Threshold search (IRTS) presented by [96]; also a Tabu-Enhanced Iterated Greedy Algorithm (TIG) presented by [94], a Strategic Oscillation (SO) presented by [93], and finally a Hybridization of Tabu Search (HTS) presented by [97] are considered. The results for the different metaheuristic presented in the Tables 3.6 and 3.7 are taken from [97]. However, a scale factor was used with respect to the original computing times since the reported results are obtained in 15 seconds for $n = 100$ and 90 seconds for $n = 200$, but using a computer with less computing power. The reported computing times (expressed in seconds) are scaled with respect to our computer. The results of CPLEX 12.4, which will not be analyzed due to their poor performance, are also added.

The MS-ILS metaheuristic approach is not compared with the matheuristic described in section 3.6, because the latter requires (as shown in Tables 3.4 and 3.9) long computing time to execute the bundle procedure for the instances with $n > 50$ (mainly when the number of knapsacks is small).

Tables 3.6 and 3.7 report, for each instance, the value of the best known solution (BKS) and, for each metaheuristic algorithm, the following values:

- Max: value of the best solution found by executing 40 runs;

- % Gap: percentage gap between BKS and Max;
- Avg: average value of the solutions found by executing 40 runs;
- Time: average computing time for each run.

MS-ILS appears to be competitive for the instances with $n = 100$. Our approach is not better in solution quality (considering the maximum and average values) than HTS and IRTS, which are the current best approaches in the literature, and are extremely stable despite the stochastic component: HTS obtains an average value equal to the best solution value in 20 of the 30 instances, and IRTS gets an average value equal to the best solution value in 11 of the 30 instances. With respect to the SO and TIG metaheuristics, MS-ILS finds a better maximum value for 5 instances; it seems to be more competitive for the remaining instances, not having significant gaps. Regarding the execution times, our algorithm does not manage to use the time limit specified by the state-of-the-art algorithms, being competitive with all the approaches in the literature. Indeed, in the best case, MS-ILS uses an average of 2.03 seconds (for the 100.75.3 group), and in the worst case, it uses an average of 5.94 seconds (for the 100.25.3 group).

MIS-ILS obtains encouraging results for the largest instances of the literature with $n = 200$ (see Table 3.7). The proposed algorithm is not able to obtain competitive results with respect to HTS and IRTS. However, both algorithms lose stability for this group of instances; HTS obtains an average value equal to the best solution value in 8 of the 30 instances, while IRTS obtains an average value equal to the best solution value in 2 of the 30 instances. Regarding SO and TIG metaheuristics, MS-ILS finds a better maximum value for 2 instances. Regarding the computation time, MS-ILS is three times faster in the best case (for group 200,75,3) and uses a similar computation time in the worst-case (group 200,25,10).

A statistical comparison is performed between MS-ILS and the other metaheuristics. Wilcoxon signed-rand test was applied to check the average performance of each algorithm; the details are reported in Appendix B. Using two groups of instances per number of items ($n = \{100, 200\}$) to perform the analysis, MS-ILS fails to have competitive performance, and in all hypothesis tests, the algorithm does not prove to be better than any of the algorithms of the literature. Figures B.1 and B.2 show two box plots for the average performance of each metaheuristic (see Appendix B).

3.7.3 Matheuristic Experiments

General descriptions and Parameter Setting

The thresholds for the proposed matheuristic were defined based on their performance. The procedures executed after bundle procedure are fast, so the matheuristic process is executed with different thresholds and the best value is selected. The thresholds considered (i.e. *selectThreshold* for the Algorithms 15 and 16) are the following:

- UBM: (1 + 0.001) * UB

TABLE 3.6: Computacional Results for the MS-ILS with the instances from [83] with $n = 100$

n	d	m	l	BKS	MS-ILS				HTS [97]				TIG				SO [93]				IRTS [96]				CPLEX 12.4			
					Max	%Gap	Avg.	Time(s)	Max	%Gap	Avg.	Time(s)	Max	%Gap	Avg.	Time(s)	Max	%Gap	Avg.	Time(s)	Max	%Gap	Avg.	Time(s)	LB	UB	Gap%	Time(s)
100	25	3	1	29286	29286	0.00	29006.6	2.06	29286	0.00	29286.0	7.79	29286	0.00	29027.90	7.79	29286	0.00	29201.70	7.79	29286	0.00	29286.00	7.79	28077	53963.23	92.2	1868.85
100	25	3	2	28491	28491	0.00	28087.0	7.94	28491	0.00	28491.0	7.79	28491	0.00	28470.70	7.79	28491	0.00	28488.32	7.79	28491	0.00	28491.00	7.79	28169	52942.22	87.9	1868.85
100	25	3	3	27179	27164	0.06	26894.4	5.64	27179	0.00	27179.0	7.79	27095	0.31	27015.90	7.79	27179	0.00	27175.20	7.79	27179	0.00	27179.00	7.79	26492	51010.72	92.6	1868.85
100	25	3	4	28593	28556	0.13	28258.5	7.33	28593	0.00	28593.0	7.79	28593	0.00	28593.00	7.79	28593	0.00	28580.75	7.79	28593	0.00	28593.00	7.79	27793	53382.60	92.1	1868.85
100	25	3	5	27892	27830	0.22	27721.9	6.71	27892	0.00	27892.0	7.79	27892	0.00	27885.33	7.79	27892	0.00	27821.98	7.79	27892	0.00	27892.00	7.79	27058	53083.61	96.2	1868.85
Avergage					28265.40	0.08	27993.65	5.94	28288.20	0.00	28288.20	7.79	28271.40	0.06	28198.57	7.79	28288.20	0.00	28253.59	7.79	28288.20	0.00	28288.20	7.79	27517.80	52876.48	92.19	1868.85
100	25	5	1	22581	22581	0.00	22346.3	2.51	22581	0.00	22579.2	7.79	22413	0.74	22273.98	7.79	22509	0.32	22403.50	7.79	22581	0.00	22530.68	7.79	21194	55784.57	163.2	1868.85
100	25	5	2	21704	21566	0.64	21404.9	4.91	21704	0.00	21687.9	7.79	21678	0.12	21648.00	7.79	21678	0.12	21622.43	7.79	21704	0.00	21667.00	7.79	20725	54647.80	163.7	1868.85
100	25	5	3	21239	21101	0.65	21003.1	4.44	21239	0.00	21239.0	7.79	21181	0.27	21099.30	7.79	21188	0.24	21153.00	7.79	21239	0.00	21235.95	7.79	19674	52614.27	167.4	1868.85
100	25	5	4	22181	22003	0.80	21919.4	7.36	22181	0.00	22181.0	7.79	22181	0.00	22180.42	7.79	22181	0.00	22164.32	7.79	22181	0.00	22180.90	7.79	20644	55098.88	166.9	1868.85
100	25	5	5	21669	21562	0.49	21305.8	4.29	21669	0.00	21669.0	7.79	21669	0.00	21663.85	7.79	21669	0.00	21567.00	7.79	21669	0.00	21656.42	7.79	20054	54887.58	173.7	1868.85
Avergage					21762.60	0.52	21595.88	4.70	21874.80	0.00	21871.22	7.79	21824.40	0.23	21773.11	7.79	21845.00	0.14	21782.05	7.79	21874.80	0.00	21854.19	7.79	20458.20	54606.62	166.98	1868.85
100	25	10	1	16221	16187	0.21	16014.5	3.06	16221	0.00	16201.8	7.79	16157	0.39	16057.60	7.79	16065	0.96	15996.83	7.79	16221	0.00	16200.53	7.79	14804	57710.44	289.8	1868.85
100	25	10	2	15700	15612	0.56	15475.2	3.48	15700	0.00	15700.0	7.79	15700	0.00	15557.68	7.79	15617	0.53	15446.40	7.79	15700	0.00	15665.65	7.79	14191	56294.00	296.7	1868.85
100	25	10	3	14927	14798	0.86	14614.8	3.19	14927	0.00	14892.0	7.79	14832	0.64	14736.23	7.79	14760	1.12	14648.43	7.79	14927	0.00	14852.00	7.79	13560	54274.15	300.3	1868.85
100	25	10	4	16181	16145	0.22	15967.8	3.31	16181	0.00	16181.0	7.79	16181	0.00	16168.50	7.79	16159	0.14	16082.68	7.79	16181	0.00	16181.00	7.79	14630	56871.67	288.7	1868.85
100	25	10	5	15326	15182	0.94	15066.7	3.49	15326	0.00	15326.0	7.79	15289	0.24	15189.45	7.79	15196	0.85	15094.89	7.79	15326	0.00	15293.00	7.79	14142	56609.11	300.3	1868.85
Avergage					15584.80	0.56	15427.78	3.31	15671.00	0.00	15660.16	7.79	15631.80	0.25	15541.89	7.79	15559.40	0.72	15453.85	7.79	15671.00	0.00	15638.44	7.79	14265.40	56351.87	295.16	1868.85
100	75	3	1	69977	69977	0.00	69736.8	1.39	69977	0.00	69977.0	7.79	69935	0.06	69935.00	7.79	69935	0.06	69935.00	7.79	69977	0.00	69977.00	7.79	69010	157543.51	128.29	1868.85
100	75	3	2	69504	69462	0.06	69207.4	2.23	69504	0.00	69504.0	7.79	69504	0.00	69504.00	7.79	69504	0.00	69497.40	7.79	69504	0.00	69499.60	7.79	68157	158390.17	132.39	1868.85
100	75	3	3	68832	68832	0.00	68742.9	1.17	68832	0.00	68832.0	7.79	68832	0.00	68816.20	7.79	68832	0.00	68813.00	7.79	68832	0.00	68832.00	7.79	67681	157395.07	132.55	1868.85
100	75	3	4	70028	69879	0.21	69725.1	3.46	70028	0.00	70028.0	7.79	70028	0.00	70028.00	7.79	70028	0.00	70028.00	7.79	70028	0.00	70028.00	7.79	69717	154667.08	121.85	1868.85
100	75	3	5	69692	69596	0.14	69474.3	1.91	69692	0.00	69692.0	7.79	69692	0.00	69681.30	7.79	69692	0.00	69652.22	7.79	69692	0.00	69692.00	7.79	68638	160393.28	133.68	1868.85
Avergage					69549.20	0.08	69377.29	2.03	69606.60	0.00	69606.60	7.79	69598.20	0.01	69592.90	7.79	69598.20	0.01	69585.12	7.79	69606.60	0.00	69605.72	7.79	68640.60	157677.82	129.75	1868.85
100	75	5	1	49421	49240	0.37	49173.9	1.84	49421	0.00	49421.0	7.79	49421	0.00	49295.60	7.79	49363	0.12	49238.83	7.79	49421	0.00	49365.98	7.79	48270	161306.19	234.17	1868.85
100	75	5	2	49400	49315	0.17	49133.7	2.03	49400	0.00	49386.0	7.79	49360	0.08	49266.80	7.79	49320	0.16	49226.60	7.79	49365	0.07	49350.60	7.79	48643	162654.44	234.38	1868.85
100	75	5	3	48495	48495	0.00	48358.5	2.03	48495	0.00	48495.0	7.79	48495	0.00	48474.20	7.79	48495	0.00	48360.85	7.79	48495	0.00	48495.00	7.79	44474	161403.85	262.92	1868.85
100	75	5	4	50246	50084	0.32	49979.0	2.24	50246	0.00	50246.0	7.79	50246	0.00	49966.60	7.79	50246	0.00	50124.20	7.79	50246	0.00	50141.50	7.79	48756	159342.88	226.82	1868.85
100	75	5	5	48753	48650	0.21	48388.2	2.28	48753	0.00	48753.0	7.79	48752	0.00	48735.20	7.79	48752	0.00	48718.38	7.79	48753	0.00	48749.10	7.79	47286	164701.44	248.31	1868.85
Avergage					49156.80	0.21	49006.62	2.08	49263.00	0.00	49260.20	7.79	49254.80	0.02	49147.68	7.79	49235.20	0.06	49133.77	7.79	49256.00	0.01	49220.44	7.79	47485.80	161881.76	241.32	1868.85
100	75	10	1	30296	29980	1.04	29791.4	3.54	30296	0.00	30232.4	7.79	30138	0.52	29900.84	7.79	30018	0.92	29897.80	7.79	30296	0.00	30240.20	7.79	28124	165782.37	489.47	1868.85
100	75	10	2	31207	31009	0.63	30879.4	3.16	31207	0.00	31056.7	7.79	31092	0.37	30969.15	7.79	30973	0.75	30914.00	7.79	31207	0.00	31095.80	7.79	29436	167227.44	468.11	1868.85
100	75	10	3	29908	29657	0.84	29493.2	3.20	29908	0.00	29896.3	7.79	29812	0.32	29662.00	7.79	29765	0.48	29638.80	7.79	29908	0.00	29894.75	7.79	27340	165637.23	505.84	1868.85
100	75	10	4	31762	31476	0.90	31303.5	3.32	31762	0.00	31710.9	7.79	31672	0.28	31491.82	7.79	31634	0.40	31481.30	7.79	31762	0.00	31706.50	7.79	27916	163759.49	486.62	1868.85
100	75	10	5	30507	30165	1.12	30004.2	3.32	30507	0.00	30450.9	7.79	30188	1.05	30046.10	7.79	30348	0.52	30055.42	7.79	30507	0.00	30458.50	7.79	27003	168992.47	525.83	1868.85
Avergage					30457.40	0.91	30294.30	3.31	30736.00	0.00	30669.44	7.79	30580.40	0.51	30413.98	7.79	30547.60	0.61	30397.46	7.79	30736.00	0.00	30679.15	7.79	27963.80	166279.80	495.17	1868.85

TABLE 3.7: Computacional Results for the MS-ILS with the instances from [83] with $n = 200$

n	d	m	l	BKS	MS-ILS				HTS [97]				TIG				SO [93]				IRTS [96]				CPLEX 12.4			
					Max	%Gap	Avg.	Time(s)	Max	%Gap	Avg.	Time(s)	Max	%Gap	Avg.	Time(s)	Max	%Gap	Avg.	Time(s)	Max	%Gap	Avg.	Time(s)	LB	UB	Gap%	Time(s)
200	25	3	1	101471	100992	0.47	100376.4	17.90	101471	0.00	101396.0	46.72	100372	1.08	100207.00	46.72	101218	0.25	100776.00	46.72	101471	0.00	101441.00	46.72	94992	222051.26	133.8	1868.85
200	25	3	2	107958	107901	0.05	107604.4	12.90	107958	0.00	107958.0	46.72	107927	0.03	107814.00	46.72	107958	0.00	107663.00	46.72	107958	0.00	107958.00	46.72	105978	222937.93	110.4	1868.85
200	25	3	3	104589	104329	0.25	103955.4	15.63	104589	0.00	104579.0	46.72	104532	0.05	104445.00	46.72	104538	0.05	104365.00	46.72	104589	0.00	104559.00	46.72	98214	219794.08	123.8	1868.85
200	25	3	4	100136	99002	1.13	98388.4	24.92	100136	0.00	100103.0	46.72	99000	1.13	98836.80	46.72	99559	0.58	99170.50	46.72	100098	0.04	100098.00	46.72	93693	219675.07	134.5	1868.85
200	25	3	5	102311	102041	0.26	101719.5	17.33	102311	0.00	102311.0	46.72	101999	0.30	101877.00	46.72	102084	0.22	101792.00	46.72	102311	0.00	102310.00	46.72	94818	217794.90	129.7	1868.85
Avergage					102853.00	0.43	102408.80	17.73	103293.00	0.00	103269.40	46.72	102766.00	0.52	102635.96	46.72	103071.40	0.22	102753.30	46.72	103285.40	0.01	103273.20	46.72	97539.00	220450.65	126.41	1868.85
200	25	5	1	75623	75458	0.22	74802.3	32.57	75623	0.00	75479.5	46.72	74682	1.24	74361.52	46.72	74665	1.27	74389.82	46.72	75623	0.00	75554.10	46.72	67464	226198.30	235.3	1868.85
200	25	5	2	80033	79596	0.55	79099.7	26.94	80033	0.00	80033.0	46.72	79604	0.54	79459.33	46.72	79473	0.70	79244.40	46.72	80033	0.00	80023.40	46.72	71876	226791.30	215.5	1868.85
200	25	5	3	78043	77341	0.90	77164.9	26.85	78043	0.00	78043.0	46.72	77795	0.32	77720.58	46.72	77695	0.45	77570.82	46.72	78043	0.00	78028.95	46.72	70259	223882.05	218.7	1868.85
200	25	5	4	74140	73335	1.09	72831.7	32.99	74140	0.00	74024.2	46.72	73189	1.28	72984.40	46.72	73405	0.99	73005.00	46.72	74140	0.00	74061.29	46.72	65940	223859.76	239.5	1868.85
200	25	5	5	76610	75944	0.87	75776.9	25.81	76610	0.00	76607.7	46.72	76137	0.62	75905.14	46.72	76037	0.75	75829.90	46.72	76610	0.00	76597.62	46.72	69523	222056.15	219.4	1868.85
Avergage					76334.80	0.72	75935.07	29.03	76889.80	0.00	76837.48	46.72	76281.40	0.80	76086.19	46.72	76255.00	0.83	76007.99	46.72	76889.80	0.00	76853.07	46.72	69012.40	224557.51	225.67	1868.85
200	25	10	1	52293	51294	1.91	50818.2	45.44	52293	0.00	52114.8	46.72	51592	1.34	51298.40	46.72	51389	1.73	51043.93	46.72	52293	0.00	52158.50	46.72	43054	230387.32	435.1	1868.85
200	25	10	2	54830	54105	1.32	53828.5	35.51	54830	0.00	54539.7	46.72	54290	0.98	54077.30	46.72	54102	1.33	53831.20	46.72	54830	0.00	54666.25	46.72	45774	231059.61	404.8	1868.85
200	25	10	3	53678	52912	1.43	52416.5	37.67	53678	0.00	53594.8	46.72	52985	1.29	52791.49	46.72	52841	1.56	52483.48	46.72	53661	0.03	53588.28	46.72	44187	227911.74	415.8	1868.85
200	25	10	4	51302	50330	1.89	49942.5	45.34	51302	0.00	51097.6	46.72	50577	1.41	50282.50	46.72	50371	1.81	50002.82	46.72	51297	0.01	51078.20	46.72	41608	228108.72	448.2	1868.85
200	25	10	5	53621	53110	0.95	52768.2	40.67	53621	0.00	53605.2	46.72	53337	0.53	52856.38	46.72	52596	1.91	52395.10	46.72	53621	0.00	53532.24	46.72	43847	226045.81	415.5	1868.85
Avergage					52350.20	1.50	51954.76	40.93	53144.80	0.00	52990.42	46.72	52556.20	1.11	52261.21	46.72	52259.80	1.67	51951.31	46.72	53140.40	0.01	53004.69	46.72	43694.00	228702.64	423.89	1868.85
200	75	3	1	270718	269965	0.28	269261.5	15.39	270718	0.00	270681	46.72	270718	0.00	270718.00	46.72	270718	0.00	270697.00	46.72	270718	0.00	270685.00	46.72	258740	644499.87	149.09	1868.85
200	75	3	2	257288	257277	0.00	256481.8	15.84	257288	0.00	257288	46.72	257288	0.00	257099.00	46.72	257277	0.00	256931.00	46.72	257288	0.00	257273.00	46.72	248139	645348.35	160.08	1868.85
200	75	3	3	270069	269319	0.28	268663.7	15.70	270069	0.00	270069	46.72	270069	0.00	270069.00	46.72	270069	0.00	270028.00	46.72	270069	0.00	269926.00	46.72	262806	649880.92	147.29	1868.85
200	75	3	4	246993	246130	0.35	245134.8	16.21	246993	0.00	246881	46.72	246882	0.04	246684.00	46.72	246882	0.04	246555.00	46.72	246993	0.00	246877.00	46.72	227193	633617.30	178.89	1868.85
200	75	3	5	279598	277959	0.59	277630.8	14.94	279598	0.00	279598	46.72	279598	0.00	279598.00	46.72	279598	0.00	279598.00	46.72	279598	0.00	279570.00	46.72	270979	655098.56	141.75	1868.85
Avergage					264130.00	0.30	263434.49	15.62	264933.20	0.00	264903.40	46.72	264911.00	0.01	264833.60	46.72	264908.80	0.01	264761.80	46.72	264933.20	0.00	264866.20	46.72	253571.40	645689.00	155.42	1868.85
200	75	5	1	185493	183946	0.83	183380.1	26.42	185493	0.00	185443	46.72	184984	0.27	184774.00	46.72	184882	0.33	184641.00	46.72	185493	0.00	184904.00	46.72	164519	654784.78	298	1868.85
200	75	5	2	174836	173935	0.52	173349.3	25.71	174836	0.00	174836	46.72	174776	0.03	174642.00	46.72	174682	0.09	174445.00	46.72	174836	0.00	174688.00	46.72	160577	656046.05	308.56	1868.85
200	75	5	3	186774	185839	0.50	185115.1	24.01	186782	0.00	186732	46.72	186674	0.05	186507.00	46.72	186619	0.08	186352.00	46.72	186774	0.00	186674.00	46.72	175943	661330.18	275.88	1868.85
200	75	5	4	167142	165500	0.98	164754.4	30.29	167142	0.00	166918	46.72	166832	0.19	166487.00	46.72	166584	0.33	166246.00	46.72	166990	0.09	166747.00	46.72	158807	643762.33	305.37	1868.85
200	75	5	5	193310	192498	0.42	191771.7	22.68	193310	0.00	193263	46.72	193255	0.03	193002.00	46.72	193138	0.09	192836.00	46.72	193310	0.00	193217.00	46.72	173698	665834.00	283.33	1868.85
Avergage					180343.60	0.65	179674.12	25.82	181512.60	0.00	181438.40	46.72	181304.20	0.12	181082.40	46.72	181181.00	0.18	180904.00	46.72	181480.60	0.02	181246.00	46.72	166708.80	656351.47	294.23	46.72
200	75	10	1	113324	112398	0.82	111825.2	21.57	113324	0.00	113061	46.72	112591	0.65	112330.00	46.72	112457	0.77	112258.00	46.72	113139	0.16	112809.00	46.72	95514	665339.06	596.59	46.72
200	75	10	2	105959	105062	0.85	104311.4	31.93	105959	0.00	105800	46.72	105297	0.62	105064.00	46.72	105260	0.66	104947.00	46.72	105807	0.14	105437.00	46.72	88469	667546.03	654.55	46.72
200	75	10	3	114860	113453	1.22	113056.2	20.60	114860	0.00	114599	46.72	114237	0.54	113930.00	46.72	114007	0.74	113717.00	46.72	114596	0.23	114367.00	46.72	94768	672239.84	609.35	46.72
200	75	10	4	99309	97875	1.44	97137.1	26.16	99309	0.00	98850.3	46.72	98556	0.76	98219.93	46.72	98285	1.03	97885.95	46.72	99106	0.20	98851.55	46.72	88580	653941.84	638.25	46.72
200	75	10	5	117309	115975	1.14	115558.4	16.30	117110	0.17	116940	46.72	116725	0.50	116266.00	46.72	116298	0.86	116031.00	46.72	117309	0.00	116947.00	46.72	98288	676442.05	588.22	46.72
Avergage					108952.60	1.09	108377.64	23.31	110112.40	0.03	109850.06	46.72	109481.20	0.61	109161.99	46.72	109261.40	0.81	108967.79	46.72	109991.40	0.15	109682.31	46.72	93123.80	667101.76	617.39	46.72

- UBM: (1 + 0.01) * UB
- UBM: (1 + 0.02) * UB
- UBM: UB + 1.0
- UBM: UB + 2.0

Where UB is the Upper Bound associated with the candidate QKP solution used within the matheuristic process.

Variants of the presented algorithms arise when the processes are executed using different time limits. First, the *quadknap* function is bounded with a *QTL* time limit for the execution of the Lagrangian process; that is, each iteration of the bundle procedure can take a maximum time equal to *QTL*. Second, the global time for the execution of the bundle procedure can be limited with a time limit equal to *BTL*. From this perspective, we define $QTL=\{3, 5\}$ seconds and $BTL=\{100, 300, 500\}$ seconds.

A variant of Algorithm 15 is presented. The current stopping criterion of the matheuristic is satisfied when the generated solution is inside the tabu list. However, an alternative execution is presented with a new stopping criterion. Specifically, at line 18, the matheuristic is stopped if all the values in g are negative and if the algorithm has been executed a specific number of iterations. The results of this variant are shown in the tables 3.8 and 3.9 with the suffix *scit* (stop criteria using iterations).

The experiments were performed on a single thread of Intel(R) Core(TM) i7-8700K CPU @ 3.70GHz with 32 GB RAM (characteristics similar to those of the system used for the BP algorithm by [101]).

Two groups of instances were used. The first group considers the instances presented by [101], with the number of items $n = \{20, 25, 30, 35\}$, densities $d = \{25, 50, 75\}$ and $m = \{3, 5, 10\}$; each triple (n, d, m) has five instances. The second group was described in the previous sections and published in [59], with the number of items $n = \{40, 45, 50, 55, 60\}$, densities $d = \{25, 50, 75\}$ and $m = \{3, 5, 10\}$, each triple (n, d, m) has five instances.

Comparison

Tables 3.8 and 3.9 show a comparison of the proposed matheuristic with the Branch & Bound (B&B) algorithm presented in [119]. This algorithm is available at `https://sites.google.com/view/kfleszar/research` and was executed for all the instances with a time limit of one hour. The considered versions of the proposed matheuristic correspond to the pairs *(BTL,QTL)*= (100,3), (100,5), (300,3), (300,5), (500,3), (500,5), and to the variant of Algorithm 15 denoted as *scit* (with BTL=500). The values of BTL and QTL are expressed in second.

Tables 3.8 and 3.9 report, for each triple (m, m, d) and for each algorithm, the averages (computed with respect to the five instances of the triple) of the following values;

- Value: value of the best solution found (or value of the upper bound for the bundle procedure);

- Time: computing time (expressed in seconds);
- Gap %: percentage gap between the solution value of the Branch & Bound (B&B) algorithm and the best solution value found by the matheuristic.

For the Bergman instances (with $n \leq 35$), the matheuristic is competitive in terms of solution values and computing time with respect to B&B. There are no significant differences in time and quality of solution for the different versions of the matheuristic for this group of instances, indeed, the matheuristics find the optimal solution for 13 out of 36 triples of instances regardless of the time limit used. The matheuristic *(500,scit)* finds optimal solution in 21 out of 36 triples in much shorter times than B&B. For example, for the triple $(35, 50, 5)$, B&B finds the optimal solutions with an average time 1412.79 seconds, while the matheuristic *(500,scit)* takes on average 4 seconds.

For the instances in [59] ($n \geq 40$), the matheuristic is competitive in result quality and vastly superior in computational time. Table 3.9 shows that B&B reaches the time limit of 1 hour for most of the instance triples. The versions of the matheuristic with smaller time limits worsen their solution quality in many instances, but not in a significant way. For specific instance triples, the matheuristic versions with smaller time limits may have large % gaps with respect to B&B (4.06 % gap for triple (60.25.5)), but the average & gap of the total set of instances (that is, the average of the average gaps) does not reach 1% in any case. The matheuristic version *(500,5)* found the average best known solution (BKS) on 8 out of 45 triples. The matheuristic version *(500,scit)* finds good results in this set of instances, but with a longer time than the other versions. This metaheuristic version finds the average best known solution in 14 out of 45 triples in much shorter times than B&B.

A comparison with the metaheuristics proposed in the literature is not performed, since no computational results obtained by these metaheuristics are reported for instances (having values of n in the range $\{20, ..60\}$) considered in this section.

TABLE 3.8: Average values over the 5 instances of each triple (n,m,d) for the Bergman Instances.

			Branch & Bound		Bundle *BTL*=3600sg		Matheuristic (100,3)			Matheuristic (100,5)			Matheuristic (300,3)			Matheuristic (300,5)			Matheuristic (500,3)			Matheuristic (500,5)			Matheuristic (500,scit)		
n	d	m	Value	Time	Value	Time	Value	Gap %	Time	Value	Gap %	Time	Value	Gap %	Time	Value	Gap %	Time	Value	Gap %	Time	Value	Gap %	Time	Value	Gap %	Time
20	25	3	2289.2	0.03	2295.5	0.04	2282.8	0.31	0.10	2283	0.31	0.10	2282.8	0.31	0.10	2282.8	0.31	0.11	2282.8	0.31	0.10	2282.8	0.31	0.10	2289.2	0.00	1.08
20	25	5	1916.2	0.07	1918.31	0.04	1905.4	0.48	0.08	1905	0.48	0.08	1905.4	0.48	0.08	1905.4	0.48	0.08	1905.4	0.48	0.08	1905.4	0.48	0.08	1911	0.23	0.46
20	25	10	1199	0.01	1199.09	0.04	1199	0.00	0.07	1199	0.00	0.07	1199	0.00	0.07	1199	0.00	0.07	1199	0.00	0.07	1199	0.00	0.07	1199	0.00	0.82
20	50	3	3079.8	0.09	3093.8	0.05	3079.8	0.00	0.08	3080	0.00	0.08	3079.8	0.00	0.08	3079.8	0.00	0.08	3079.8	0.00	0.08	3079.8	0.00	0.08	3079.8	0.00	0.30
20	50	5	2296.6	0.21	2297.21	0.05	2296.6	0.00	0.12	2297	0.00	0.12	2296.6	0.00	0.12	2296.6	0.00	0.12	2296.6	0.00	0.12	2296.6	0.00	0.12	2296.6	0.00	0.88
20	50	10	1427	0.01	1433.68	0.04	1427	0.00	0.08	1427	0.00	0.08	1427	0.00	0.08	1427	0.00	0.08	1427	0.00	0.08	1427	0.00	0.08	1427	0.00	1.06
20	75	3	3553.8	0.17	3592.79	0.05	3522.8	0.84	0.08	3523	0.84	0.08	3522.8	0.84	0.08	3522.8	0.84	0.08	3522.8	0.84	0.09	3522.8	0.84	0.09	3543.8	0.27	0.31
20	75	5	2640.6	0.16	2654.26	0.05	2640.6	0.00	0.11	2641	0.00	0.11	2640.6	0.00	0.11	2640.6	0.00	0.11	2640.6	0.00	0.11	2640.6	0.00	0.11	2640.6	0.00	0.91
20	75	10	1639.6	0.02	1639.6	0.04	1639.6	0.00	0.07	1640	0.00	0.07	1639.6	0.00	0.07	1639.6	0.00	0.07	1639.6	0.00	0.07	1639.6	0.00	0.07	1639.6	0.00	0.80
25	25	3	3002.4	0.20	3014.6	0.11	2999.4	0.10	0.16	2999	0.10	0.17	2999.4	0.10	0.16	2999.4	0.10	0.17	2999.4	0.10	0.16	2999.4	0.10	0.17	2999.4	0.10	0.69
25	25	5	2559	0.67	2566	0.09	2559	0.00	0.14	2559	0.00	0.15	2559	0.00	0.14	2559	0.00	0.15	2559	0.00	0.15	2559	0.00	0.15	2559	0.00	0.92
25	25	10	1781.2	0.08	1784.4	0.08	1773.8	0.44	0.23	1774	0.44	0.23	1773.8	0.44	0.23	1773.8	0.44	0.23	1773.8	0.44	0.23	1773.8	0.44	0.23	1773.8	0.44	1.30
25	50	3	4275	1.07	4295.2	0.15	4267.8	0.17	0.23	4268	0.17	0.22	4267.8	0.17	0.22	4267.8	0.17	0.22	4267.8	0.17	0.22	4267.8	0.17	0.22	4275	0.00	0.79
25	50	5	3272.8	2.49	3283	0.10	3272.8	0.00	0.15	3273	0.00	0.15	3272.8	0.00	0.16	3272.8	0.00	0.15	3272.8	0.00	0.16	3272.8	0.00	0.16	3272.8	0.00	1.06
25	50	10	2093.2	0.14	2093.2	0.07	2093.2	0.00	0.13	2093	0.00	0.12	2093.2	0.00	0.13	2093.2	0.00	0.13	2093.2	0.00	0.13	2093.2	0.00	0.12	2093.2	0.00	1.36
25	75	3	5155	3.46	5178.8	0.22	5126.2	0.57	0.31	5126	0.57	0.31	5126.2	0.57	0.31	5126.2	0.57	0.31	5126.2	0.57	0.31	5126.2	0.57	0.31	5151.6	0.07	0.89
25	75	5	3774.8	5.99	3807.6	0.14	3728.8	1.22	0.23	3729	1.22	0.23	3728.8	1.22	0.23	3728.8	1.22	0.23	3728.8	1.22	0.24	3728.8	1.22	0.23	3774.8	0.00	1.38
25	75	10	2420.2	0.16	2427	0.11	2416.2	0.17	0.17	2416	0.17	0.17	2416.2	0.17	0.17	2416.2	0.17	0.17	2416.2	0.17	0.17	2416.2	0.17	0.17	2417.4	0.12	1.61
30	25	3	4194.8	0.73	4203.4	0.32	4162.4	0.79	0.47	4162	0.79	0.47	4162.4	0.79	0.47	4162.4	0.79	0.46	4162.4	0.79	0.46	4162.4	0.79	0.46	4194.8	0.00	1.40
30	25	5	3550.4	6.51	3558	0.18	3537.6	0.38	0.28	3538	0.38	0.28	3537.6	0.38	0.28	3537.6	0.38	0.28	3537.6	0.38	0.29	3537.6	0.38	0.29	3550.4	0.00	1.66
30	25	10	2558.4	10.86	2562.4	0.15	2558.4	0.00	0.26	2558	0.00	0.26	2558.4	0.00	0.26	2558.4	0.00	0.26	2558.4	0.00	0.26	2558.4	0.00	0.26	2558.4	0.00	2.44
30	50	3	5858.6	19.10	5903.4	0.69	5819.4	0.67	0.94	5819	0.67	0.94	5819.4	0.67	0.95	5819.4	0.67	0.95	5819.4	0.67	0.95	5819.4	0.67	0.95	5850.8	0.13	1.97
30	50	5	4609.4	21.39	4611.2	0.23	4609.4	0.00	0.34	4609	0.00	0.34	4609.4	0.00	0.34	4609.4	0.00	0.34	4609.4	0.00	0.34	4609.4	0.00	0.33	4609.4	0.00	1.58
30	50	10	2963.2	6.35	2965	0.16	2963.2	0.00	0.27	2963	0.00	0.27	2963.2	0.00	0.27	2963.2	0.00	0.27	2963.2	0.00	0.27	2963.2	0.00	0.27	2963.2	0.00	2.34
30	75	3	7210.2	42.82	7291.2	0.83	7209.4	0.01	1.15	7209	0.01	1.16	7209.4	0.01	1.16	7209.4	0.01	1.15	7209.4	0.01	1.16	7209.4	0.01	1.16	7209.4	0.01	2.43
30	75	5	5362	30.57	5376	0.22	5362	0.00	0.33	5362	0.00	0.33	5362	0.00	0.33	5362	0.00	0.33	5362	0.00	0.33	5362	0.00	0.33	5362	0.00	1.58
30	75	10	3283.4	2.16	3292.2	0.12	3264.2	0.62	0.23	3264	0.62	0.22	3264.2	0.62	0.23	3264.2	0.62	0.22	3264.2	0.62	0.23	3264.2	0.62	0.22	3281	0.08	1.70
35	25	3	5277.8	10.74	5304.4	1.98	5259.2	0.37	2.72	5259	0.37	2.71	5259.2	0.37	2.71	5259.2	0.37	2.71	5259.2	0.37	2.71	5259.2	0.37	2.71	5272.4	0.11	4.16
35	25	5	4463	87.35	4473.4	0.44	4453.2	0.23	0.66	4453	0.23	0.67	4453.2	0.23	0.67	4453.2	0.23	0.66	4453.2	0.23	0.67	4453.2	0.23	0.66	4462.8	0.00	4.42
35	25	10	3318.6	45.29	3324.2	0.23	3313.6	0.15	0.44	3314	0.15	0.43	3313.6	0.15	0.43	3313.6	0.15	0.43	3313.6	0.15	0.43	3313.6	0.15	0.43	3317.6	0.03	4.68
35	50	3	7730.6	443.02	7768.2	3.06	7715.8	0.19	4.16	7716	0.19	4.18	7715.8	0.19	4.17	7715.8	0.19	4.17	7715.8	0.19	4.17	7715.8	0.19	4.19	7721.4	0.12	5.75
35	50	5	6000.2	1412.79	6021	0.68	5982.6	0.31	1.02	5983	0.31	1.01	5982.6	0.31	1.01	5982.6	0.31	1.02	5982.6	0.31	1.03	5982.6	0.31	1.02	6000	0.00	3.90
35	50	10	4062.2	86.65	4068.6	0.21	4021.6	1.03	0.42	4022	1.03	0.42	4021.6	1.03	0.42	4021.6	1.03	0.42	4021.6	1.03	0.42	4021.6	1.03	0.42	4056.2	0.14	5.23
35	75	3	9944.4	271.80	9994.4	2.62	9906.2	0.42	3.53	9906	0.42	3.53	9906.2	0.42	3.55	9906.2	0.42	3.53	9906.2	0.42	3.54	9906.2	0.42	3.54	9944.4	0.00	5.08
35	75	5	7204.2	731.79	7227.2	0.89	7149	0.75	1.34	7149	0.75	1.35	7149	0.75	1.34	7149	0.75	1.35	7149	0.75	1.35	7149	0.75	1.35	7194.6	0.13	4.71
35	75	10	4482.6	260.15	4498.4	0.34	4478	0.10	0.69	4478	0.10	0.69	4478	0.10	0.69	4478	0.10	0.69	4478	0.10	0.70	4478	0.10	0.69	4478.8	0.08	5.79

TABLE 3.9: Average values over the 5 instances of each triple (n,m,d) for the instances by [59]

			Branch & Bound		Bundle *BTL*=3600sg		Matheuristic (100,3)			Matheuristic (100,5)			Matheuristic (300,3)			Matheuristic (300,5)			Matheuristic (500,3)			Matheuristic (500,5)			Matheuristic (500,scit)		
n	d	m	Value	Time	Value	Time	Value	Gap %	Time	Value	Gap %	Time	Value	Gap %	Time	Value	Gap %	Time	Value	Gap %	Time	Value	Gap %	Time	Value	Gap %	Time
40	25	3	6789.8	378.20	6797.2	9.58	6772.4	0.25	12.86	6772	0.25	12.79	6772.4	0.25	12.89	6772.4	0.25	12.83	6772.4	0.25	12.81	6772.4	0.25	12.84	6788.8	0.01	15.43
40	25	5	5755.2	3108.17	5777.6	1.45	5712.2	0.72	2.18	5712	0.72	2.19	5712.2	0.72	2.19	5712.2	0.72	2.19	5712.2	0.72	2.18	5712.2	0.72	2.19	5741.2	0.24	6.11
40	25	10	4381.6	3213.97	4400.6	0.41	4347	0.80	0.84	4347	0.80	0.84	4347	0.80	0.84	4347	0.80	0.83	4347	0.80	0.84	4347	0.80	0.83	4372.4	0.21	7.99
40	50	3	9869.4	3600.34	9969.8	21.20	9814	0.57	28.28	9814	0.57	28.38	9814	0.57	28.29	9814	0.57	28.34	9814	0.57	28.44	9814	0.57	28.29	9856	0.14	31.01
40	50	5	7659.6	3600.26	7684.2	2.66	7593.8	0.85	3.68	7594	0.85	3.67	7593.8	0.85	3.66	7593.8	0.85	3.67	7593.8	0.85	3.68	7593.8	0.85	3.67	7656	0.05	6.93
40	50	10	5173.2	2973.07	5207.4	0.50	5148.4	0.47	0.93	5148	0.47	0.94	5148.4	0.47	0.94	5148.4	0.47	0.94	5148.4	0.47	0.95	5148.4	0.47	0.93	5169.2	0.08	7.31
40	75	3	12170.8	3600.30	12222	22.30	12168	0.02	29.67	12168	0.02	29.69	12168	0.02	29.71	12168	0.02	29.62	12168	0.02	29.70	12168	0.02	29.71	12176	-0.05	32.08
40	75	5	8863.2	3600.04	8917.6	3.20	8759.4	1.18	4.41	8759	1.18	4.39	8759.4	1.18	4.41	8759.4	1.18	4.38	8759.4	1.18	4.41	8759.4	1.18	4.40	8853	0.11	7.52
40	75	10	5695.6	2880.46	5729.4	0.49	5678	0.34	1.05	5678	0.34	1.06	5678	0.34	1.06	5678	0.34	1.06	5678	0.34	1.05	5678	0.34	1.07	5682.4	0.26	14.31
45	25	3	8290	3089.54	8309.4	70.47	8274.4	0.19	79.45	8275	0.18	79.68	8275	0.18	104.55	8275	0.18	94.22	8275	0.18	104.55	8275	0.18	94.14	8280.4	0.12	97.64
45	25	5	6894.6	3600.44	6956.2	8.31	6917.6	-0.32	11.19	6918	-0.32	11.25	6917.6	-0.32	11.23	6917.6	-0.32	11.22	6917.6	-0.32	11.24	6917.6	-0.32	11.20	6933	-0.55	19.57
45	25	10	5252.6	3600.59	5285.6	0.67	5230.6	0.42	1.63	5231	0.42	1.64	5230.6	0.42	1.64	5230.6	0.42	1.64	5230.6	0.42	1.64	5230.6	0.42	1.61	5256.2	-0.07	21.11
45	50	3	12282.4	3600.40	12377.2	129.04	12272.8	0.09	93.33	12249	0.28	93.48	12282	0.01	156.05	12265.6	0.15	173.83	12276	0.06	156.16	12285.2	-0.02	165.90	12278.4	0.04	174.08
45	50	5	9443.4	3600.54	9513.4	10.22	9405.4	0.41	13.72	9405	0.41	13.71	9405.4	0.41	13.66	9405.4	0.41	13.68	9405.4	0.41	13.71	9405.4	0.41	13.74	9443	0.00	19.17
45	50	10	6453.6	3600.52	6493.4	0.96	6458.2	-0.07	1.81	6458	-0.07	1.83	6458.2	-0.07	1.83	6458.2	-0.07	1.82	6458.2	-0.07	1.82	6458.2	-0.07	1.83	6459.8	-0.10	11.63
45	75	3	15722.2	3600.35	15775.6	106.92	15699.2	0.14	88.94	15696	0.17	90.31	15716.2	0.04	109.74	15722.2	0.00	138.70	15716.2	0.04	109.66	15722.2	0.00	138.90	15722.2	0.00	146.88
45	75	5	11288.2	3600.08	11447.2	10.59	11346	-0.52	14.34	11346	-0.52	14.37	11346	-0.52	14.35	11346	-0.52	14.32	11346	-0.52	14.35	11346	-0.52	14.34	11383.8	-0.86	25.26
45	75	10	7118.4	3599.50	7230	1.11	7169	-0.76	1.81	7169	-0.76	1.80	7169	-0.76	1.83	7169	-0.76	1.81	7169	-0.76	1.82	7169	-0.76	1.81	7175.2	-0.85	17.73
50	25	3	9684.4	3600.15	9693.8	537.09	9537.8	1.54	97.19	9466	2.29	103.93	9647.8	0.38	207.18	9668.6	0.17	278.78	9647.8	0.38	213.19	9667	0.18	335.91	9684.4	0.00	416.39
50	25	5	8095.4	3600.59	8117	32.81	8081.8	0.18	41.78	8082	0.18	41.66	8081.8	0.18	41.71	8081.8	0.18	41.90	8081.8	0.18	41.79	8081.8	0.18	41.68	8086.6	0.12	59.30
50	25	10	6105.2	3600.63	6156.6	1.85	6116.6	-0.17	2.93	6117	-0.17	2.94	6116.6	-0.17	2.93	6116.6	-0.17	2.92	6116.6	-0.17	2.94	6116.6	-0.17	2.93	6131	-0.40	26.02
50	50	3	14636.4	3600.55	14723	1304.93	14550.6	0.60	100.69	14433	1.38	102.49	14537.4	0.68	269.33	14517	0.81	303.33	14555.8	0.55	301.85	14565.8	0.49	464.67	14610.6	0.18	528.78
50	50	5	11116	3600.66	11239.2	53.81	11057	0.54	67.13	11057	0.54	67.16	11057	0.54	67.09	11057	0.54	67.15	11057	0.54	67.08	11057	0.54	66.97	11105.4	0.09	78.24
50	50	10	7568.2	3600.55	7642	2.25	7602.4	-0.46	3.83	7602	-0.46	3.87	7602.4	-0.46	3.85	7602.4	-0.46	3.84	7602.4	-0.46	3.78	7602.4	-0.46	3.84	7606.4	-0.51	30.40
50	75	3	19162.6	3600.45	19342.2	1010.29	19020.2	0.71	91.26	19103	0.32	103.00	19108.4	0.23	207.77	19121.2	0.21	252.31	19096	0.30	236.03	19032	0.70	305.84	19232.8	-0.39	512.58
50	75	5	13852.6	3600.62	13993.4	59.49	13788.4	0.46	66.00	13788	0.46	65.98	13777	0.53	73.84	13777	0.53	74.01	13777	0.53	74.00	13777	0.53	73.83	13856.2	-0.02	83.48
50	75	10	8807.8	3600.67	8888	3.33	8773	0.40	4.99	8773	0.40	4.98	8773	0.40	4.99	8773	0.40	5.01	8773	0.40	4.99	8773	0.40	5.00	8813.2	-0.06	29.55
55	25	3	11215.6	3600.68	11266.6	2416.59	11070	1.32	101.63	11071	1.34	103.17	11155.8	0.53	294.45	11161	0.49	302.90	11137.6	0.69	304.80	11179.4	0.34	467.71	11098.8	1.05	591.30
55	25	5	9230	3600.69	9310.6	170.18	9084.2	1.58	101.93	9107	1.34	101.90	9170.6	0.63	195.81	9170.2	0.64	204.30	9170.6	0.63	196.02	9170.2	0.64	210.43	9206.6	0.26	252.88
55	25	10	6978.6	3600.84	7043.2	2.80	6931.4	0.69	4.92	6931	0.69	4.98	6931.4	0.69	4.97	6931.4	0.69	4.92	6931.4	0.69	4.98	6931.4	0.69	4.96	6985.8	-0.09	47.86
55	50	3	16923	3600.54	17200.6	3627.66	16626	1.75	101.75	16520	2.36	102.92	16809.4	0.68	301.76	16745.2	1.06	303.07	16736.8	1.11	446.57	16760.4	0.96	477.03	16466	2.69	695.96
55	50	5	12913	3600.61	12989.2	310.83	12633.8	2.14	102.21	12702	1.63	102.83	12679	1.83	257.49	12821.4	0.69	273.88	12732.6	1.41	274.57	12821.4	0.69	301.43	12897.4	0.12	401.11
55	50	10	8860.2	3600.79	8936.4	5.56	8796.4	0.73	8.12	8796	0.73	8.14	8796.4	0.73	8.06	8796.4	0.73	8.15	8796.4	0.73	8.16	8796.4	0.73	8.10	8850	0.13	47.63
55	75	3	22295.6	3600.50	22426.4	3521.85	21898.8	1.78	83.21	21628	2.96	97.16	22113.6	0.80	203.80	21980.2	1.41	216.11	22041.2	1.14	284.33	22103.8	0.85	336.21	22118.2	0.81	561.37
55	75	5	16039.4	3600.63	16196	191.23	15868.6	1.06	102.80	15966	0.45	102.54	15911.6	0.79	176.58	15989.8	0.30	226.38	15905.2	0.83	176.57	15989.8	0.30	226.40	16019.8	0.12	266.57
55	75	10	10175.4	3600.77	10297	7.86	10180.4	-0.05	10.53	10180	-0.05	10.50	10180.4	-0.05	10.50	10180.4	-0.05	10.53	10180.4	-0.05	10.45	10180.4	-0.05	10.47	10218.8	-0.43	37.36
60	25	3	12743.2	3600.64	13203.4	3768.14	12508	1.86	101.68	12429	2.46	103.08	12569.2	1.37	301.58	12603.4	1.10	303.25	12656.2	0.68	420.91	12575.8	1.31	502.16	12379.6	2.85	591.27
60	25	5	10489.6	3600.68	10598.2	1787.99	10198.6	2.77	102.81	10063	4.06	103.59	10378.8	1.06	301.66	10306.2	1.75	303.96	10332.2	1.50	388.80	10373.2	1.12	503.56	10421	0.65	607.54
60	25	10	7965.6	3599.69	8070.8	13.75	7959.2	0.04	14.84	7959	0.04	14.91	7959.2	0.04	14.93	7959.2	0.04	14.85	7959.2	0.04	14.86	7959.2	0.04	14.91	7989.4	-0.33	54.51
60	50	3	19807.4	3600.61	20811.4	3786.32	19545.2	1.33	102.85	19218	3.03	106.69	19715.2	0.46	301.45	19602	1.05	301.44	19729.4	0.37	449.22	19730.8	0.38	501.29	18989	4.13	715.98
60	50	5	15056	3600.70	15191.6	1454.69	14545.6	3.40	101.56	14513	3.61	103.16	14767	1.93	301.92	14786.6	1.80	302.12	14806.6	1.67	446.98	14895	1.07	498.57	15005.2	0.34	699.85
60	50	10	10395	3601.00	10479.8	28.12	10385.6	0.09	30.54	10386	0.09	30.51	10385.6	0.09	30.58	10385.6	0.09	30.48	10385.6	0.09	30.51	10385.6	0.09	30.50	10392.4	0.03	81.65
60	75	3	26791.8	3600.48	27413.8	3694.65	26488.6	1.14	101.69	26125	2.51	101.50	26572.8	0.82	289.17	26495.4	1.12	302.00	26565.4	0.85	456.39	26614.4	0.67	501.81	26399.6	1.47	846.49
60	75	5	19310.2	3600.80	19464.2	1340.59	18846.8	2.41	81.60	18937	1.97	102.96	18900.4	2.15	172.09	19065.6	1.30	291.08	18947.6	1.89	221.04	19024.4	1.50	407.89	19290.4	0.12	578.29
60	75	10	12346.6	3600.92	12469.2	24.30	12293.6	0.45	26.97	12294	0.45	27.00	12293.6	0.45	27.09	12293.6	0.45	27.02	12293.6	0.45	27.04	12293.6	0.45	26.99	12338.6	0.08	98.36

3.8 Conclusions

3.8.1 Formulations and Relaxations

Over the last 15 years, the quadratic multiple knapsack problem has received increasing attention from the literature, dealing almost exclusively with meta-heuristics. Although in 2019 Bergman [101] and in 2021 Fleszar [119] presented the first specialized exact algorithms, the problem has never been studied from a broader mathematical perspective. We attempted to fill this gap, by focusing on classical reformulations and relaxations and analyzing their properties, in order to gain insight into the strengths and weaknesses of such methods. Currently, exact algorithms can solve instances up to 10 knapsacks and 35 items. Yet, the original benchmark instances considered in the literature (for heuristic solutions) are one order of magnitude larger, involving up to 30 knapsacks and 300 items. We believe our results have implications for the development of future exact algorithms capable of tackling larger instances. Indeed, in an enumerative algorithm, a trade-off must be made between the quality of the upper bound and the time taken to compute it. Our results suggest that, among the different possible approaches, the most promising is the one based on the Lagrangian relaxation of the cardinality constraints of the 0-1 quadratic model, both in terms of bound quality and CPU time. In particular, the convergence of the proximal bundle method to solve the Lagrangian dual problem appears to be very fast. Another interesting observation is that the adoption of non-optimal solutions of the Lagrangian subproblems speeds up the computation of each bundle iteration without deteriorating the bounds significantly. This turns out to be true for both Lagrangian relaxations we have considered. Our experiments also show that the use of specialized methods to solve the subproblems can be crucial to reduce the computing time, and should always be preferred, when possible, to general purpose MIP solvers.

3.8.2 Multi-Start Iterated Local Search

This chapter presented an MS-ILS for the QMKP that successfully solved different groups of instances. We defined four initial solution approaches based on stochastic and deterministic processes. In addition, a perturbation used in the literature and two local searches that consider feasible and infeasible moves were used. Finally, a repair function is applied to the infeasible solution based on its infeasibility with respect to the capacity. The proposed metaheuristic has a good performance regarding the CPU time but obtains solution of lower quality than those obtained by the metaheuristics from the literature.

Different future works can be defined in this line of research. An Adaptive Large Neighborhood Search (ALNS, [66]) can consider additional destruction and repair moves for the problem, use specific defined local search procedures and update the corresponding weights associated with each move. Infeasible moves should be considered for the considered instances, since the literature results suggest to use these moves due to the quality of the obtained solutions [96, 97].

3.8.3 Matheuristic

We presented a matheuristic approach to QMKP based on the Lagrangian relaxation of the quadratic model presented in [59]. We used candidate solutions obtained through the bundle procedure and which are subsequently subjected to a hybrid process using a mathematical model and then heuristic refinements. Two variants of the matheuristic are generated based on a gain vector for the mathematical model. In addition, for each variant, new approaches are presented by imposing time limits on the execution of the Lagrangian relaxation and of the code used to solve the QKP at each iteration of the bundle procedure. The proposed matheuristics prove to be highly efficient in quality of result and much faster than the other approaches presented in the literature for the solution of the considered instances.

Different future works can be defined in this line of research. We can improve the local search methods, changing the VND for some other process, or improve the infeasible local search, specifically, the selection of the relocation move.

Chapter 4

A parallel genetic algorithm for strategic mine planning

4.1 Introduction

Scheduling mineralized material extraction from an open-pit mine is a critical stage in the optimization of mining production. In a deposit, the ore grade is not homogeneous, therefore the order in which the ore and waste must be explored to obtain the maximum economic profit should be planned. A solution to the optimization problem must specify the part of the mineral deposit that must be extracted and processed in each period to obtain the maximum profit. Thus, the solution to the problem must specify spatial and temporal dimensions for the extraction. This complex problem has received attention in the literature, however it is still a computational challenge due to the number of resources required when planning real-world solutions [120]. Extraction schedule must identify the period for each part of the mineral deposit considering the mining capacity, which defines the amount of material that can be excavated in a specific period, as well as the capacity of the plant, which limits the amount of material that can be processed.

The standard model in mine production scheduling consists of a discretization scheme of the ground in cubic extraction blocks [121]. Each block is identified using geostatistical techniques and is denoted by a set of coordinates, a grade and tonnage. Thus, a block model composed of a three-dimensional array of blocks that represents the entire deposit, is the standard tool used to search for the optimal solution. A block model may consist of a few thousand blocks and, for large open-pit operations, this number can reach millions. Then, given the block model, the common way to define the mine production schedule is to find the period in which each block should be mined and the best possible treatment for that block, giving rise to a complex optimization problem. A block schedule must comply with technical and economic considerations. First, the extraction must be compatible with the slope angles that keep the open pit from collapsing. In fact, the slope constraints consider the maximum slope angles to be satisfied at any given period and can be represented as the precedence between the blocks. When only geometrical constraints are considered, an optimal solution for this problem gives the ultimate pit limit contours, and the problem is known as the ultimate pit limit problem (UPIT, [122, 123]).

Second, a set of constraints related to mining capacity involves the extraction equipment capacity. Third, the processing capacities consider the capacity of the available facilities, which also limits some characteristics of the orebody that is sent to each destination at each period. The net present value (NPV) is the measure for evaluating a schedule and the optimal solution is the one with the maximum NPV. NPV was referenced in the objective function of the OPMPSP mathematical model (see section 1.1.3), where for practical study purposes, the value of NPV per block is precalculated before solving any optimization problem associated with the Open Pit Mine. Of course, the NPV depends on factors associated with planning, such as the time in which the block is extracted and a discount ratio α (defined later) associated with the study mine.

A variant of the problem arises when each block must be mined entirely in a single period considering a limit in the number of available resources. This variant is known as the constrained pit limit problem (CPIT) and has been formulated as an integer programming problem [124–126] . To solve the CPIT, the economic value of each block, the minimum and maximum operational capacities per period and a set of precedences per block are required. The optimal solution is obtained by maximizing the NPV for the life of the mine.

In practice, the mining process occurs through extraction phases with access roads and sufficient spaces for the loading of the vehicles that must transport the orebody to some of the predefined destinations. An extraction phase is defined by a subset of blocks that must be extracted during a time interval of the planning period. An extraction phase allows the definition of a set of operating periods by considering the spatial geometry of the mine. To consider this geometry, we propose truncated instead of complete cones. Each truncated cone has: a centroid block of the basal face, a radial basal face and a slope lateral angle. Thus, truncated cones are staggered by cuts or benches with two uncovered faces: a flat top face and a lateral vertical side inner face. Figure 4.1a) shows a cone generated from a block model, while Figure 4.1b) shows benches and ramps generated from such block model. The extraction scheduling requires a design of the benches at each period.

An extraction phase is composed of one or more truncated cones with a common base that contains basal blocks. Each truncated cone is identified by its centroid block in the base. Thus, we define a base by clustering a set of centroids according to the distance. Consequently, each extraction phase is carried out in one or more periods considering the mine capacity. An example of the extraction phases is shown in Figure 4.2. The example is composed of 10 truncated cones with centroid blocks in set $A = \{3, 27, 43, 48, 55, 61, 68, 70, 75, 93\}$. To generate bases that correspond to the extraction phases the set A is clustered with respect to the distance. The clustering process produces three bases: $f1 = \{3, 43, 48\}$, $f2 = \{27, 55, 61, 68\}$ and $f3 = \{70, 75, 93\}$. At the top of the figure are depicted the periods that in turns define the extraction phases. For this example: the first extraction phase requires period 1 and part of period 2; the second extraction phase requires part of period 2, period 3, and part of period 4; and the third extraction phase requires part of period 4, period 5 and part of period 6. The period length is limited by the mine and processing capacities.

The mine production schedule identifies the order in which the extraction phases

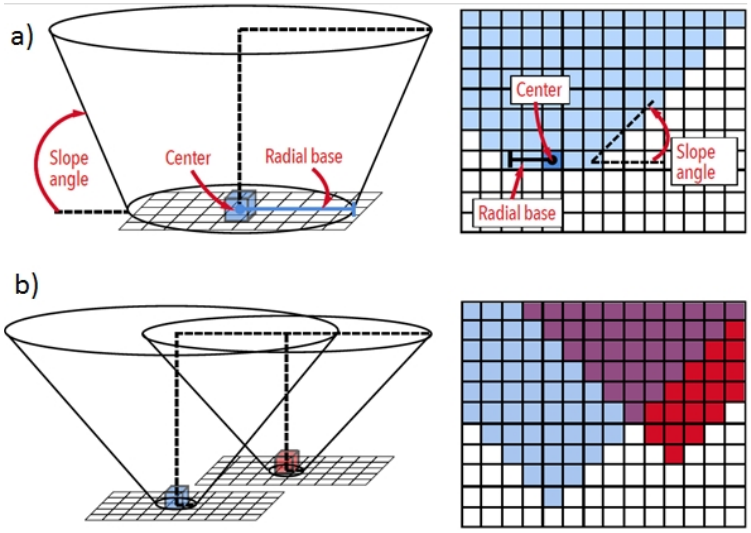


FIGURE 4.1: Truncated cone (Image from [127])

should be mined. Similar to the CPIT, in this variant of the problem that we call CPIT-P, the mine production scheduling grouped by extraction phases must satisfy the operational capacities. Thus, CPIT-P considers three additional aspects related to the CPIT: extraction phases, operational periods in which the extractions phases take place and geometrical conditions to ensure enough space to handle equipment and trucks during the operations.

Formally, let $B = \{b1, b2, \ldots, b_M\}$ be the set of blocks belonging to the initial cube containing the entire mine with M blocks, and R be a set of resources necessary to extract blocks in B. A feasible solution for CPIT-P is identified by a set of extraction phases $F = \{f_1, f2, ..., f_p\}$. In turn, each phase $f_j \in F \forall j \in \{1, 2, ..., p\}$ is characterized by a set of truncated cones and the set of blocks belonging to a truncated cone is individualized from the centroid of its base, the basal radius and the slope angle. Thus, the precedence set of a block corresponds to the set of blocks that are at the next higher level and that satisfy the inclination defined by the slope angle. Furthermore, the processing capacity CP and the mining capacity CM are known parameters. The objective is to maximize the NPV for the life cycle of the mine.

This problem was presented in Navarro, 2015, where a parallel genetic algorithm (PGA) that solves a sub-set of instances from the literature and the first results for the problem are presented. However, not all the instances were considered, and new experiments need to be run. In addition, internal parallelism processes must be improved, to obtain better results.

This chapter presents an approach to CPIT-P based on a parallel genetic algorithm improving vital processes to obtain quality results. This chapter is based on an internship in Santiago de Chile for three months. The proposed model follows a genotype-phenotype scheme. The genotype is represented by a structure that identifies the centroid blocks that compose the phases, whereas the phenotype is constructed by truncated cones, which allow adequate space and properly handle the order precedence constraints. The PGA is based on a master-slave configuration, in which a single computer node is used as the master to coordinate several slaves to

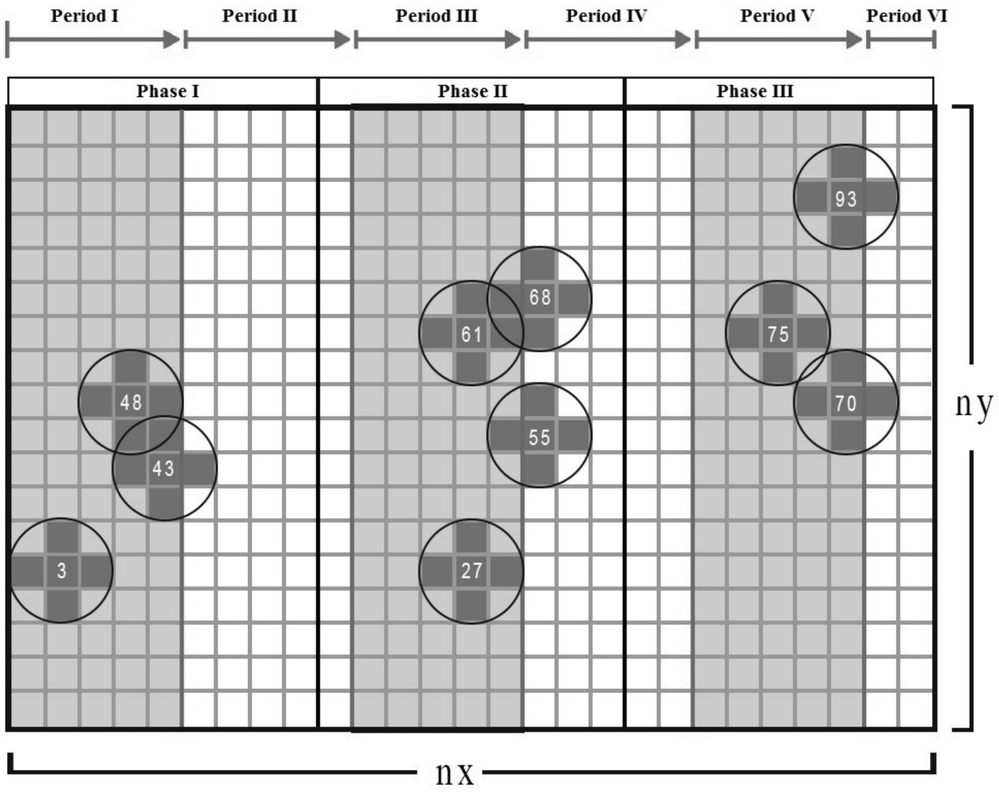


FIGURE 4.2: Examples about the phases

evaluate the fitness of an individual, which is part of the same unique population. A set of instances available in the literature are considered for a numerical experiment aiming to identify the performance of the PGA.

The remainder of this chapter is structured as follows. A general description of the Parallel Genetic Algorithm is presented in the next section. The representation of the problem for the parallel genetic algorithm is presented in Section 4.3 in which, both the representation of a solution and the evaluation function are described. In Section 4.5, a discussion on the results obtained from the computer experiment is given, and the conclusions are described in the last section.

4.2 General description of the Parallel Genetic Algorithm

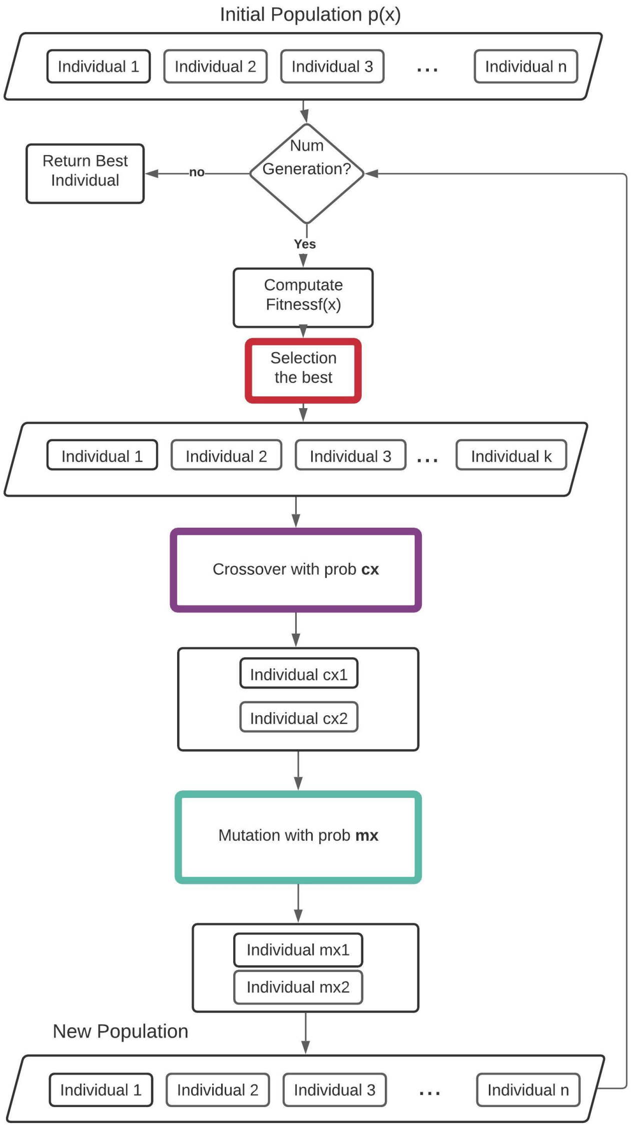


FIGURE 4.3: FlowChart of the Simple Genetic Algorithm

A Genetic Algorithm with its elementary properties was defined to solve the CPIT-P. A genetic algorithm is a population metaheuristic that works with individuals that represent solutions to the combinatorial optimization problem. Each individual is subjected to a set of operators, generating new individuals for the following generations. The logic behind this algorithm is that, as generations advance, the best individuals survive, following the Darwinian logic of human evolution [128]. Figure 4.3 shows the flow diagram of the algorithm:

1. Individuals with population size n are defined randomly.

2. A fitness function should evaluate the current population. This function depends on the problem to be addressed; for this case, a fitness function must be defined for CPIT-P.

3. A selection process of the evaluated individuals must be applied to obtain a candidate population with size k with $k < n$ (i.e., a subpopulation defined as a subset from the main population).

4. A crossover operator is applied with probability *cx* to the subpopulation from the previous selection step. This process is binary, and the content of two individuals must be exchanged with each other. In this way, we have two parent individuals and two child individuals.

5. A mutation operator is applied with probability *1-cx* to the individuals resulting from the previous step (child individuals). This operator is unary, so only the content of an individual is changed.

6. A new population is generated that is subjected again to the fitness function (Step 2). The stopping criterion of the genetic algorithm must be checked; for the presented problem, the number of generations will be used.

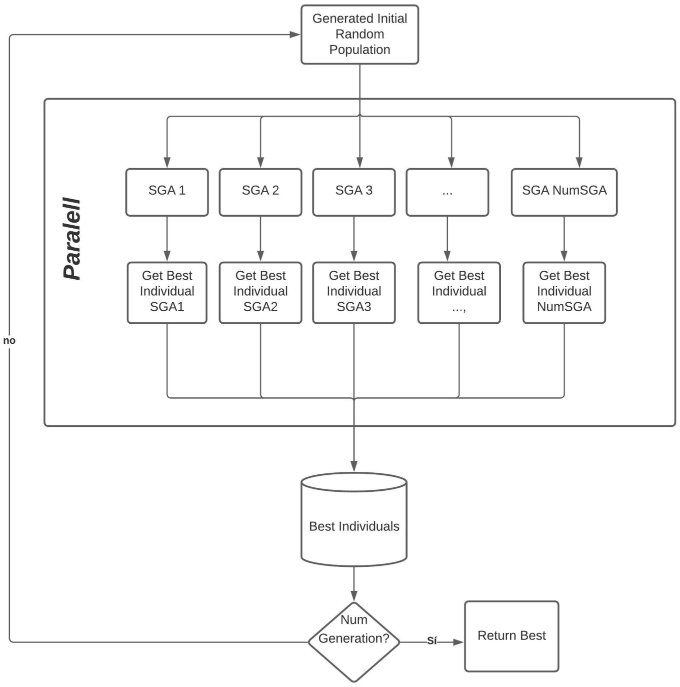


FIGURE 4.4: FlowChart of the Parallel Genetic Algorithm

The genetic algorithm described above can be run in parallel. For this end, a classic form of asynchronous parallelization can be applied to execute the genetic algorithm in different computer cores. Figure 4.4 presents a flow chart for the parallel genetic algorithm. Like the simple genetic algorithm, the parallel algorithm starts with an initial population that is considered for each of the simple genetic algorithms that run in parallel. Subsequently, each algorithm returns the best individual which is stored in a repository of the best individuals. Finally, the termination criterion based on the number of generations of the main scheme is checked and the best solution is returned.

4.3 Representation of CPIT-P for PGA

To search the space of CPIT-P by means of a GA it is necessary to define an individual, the fitness function and the operators. A GA is a search method that allows to find good solutions for optimization problems by imitating the laws of natural evolution [128, 129]. Each individual in the population encodes a point in the search space of a given problem, and the offspring are generated by a random process that emulates natural selection and works with selection, crossover and mutation operators. We propose a genotype-phenotype scheme to represent the CPIT-P solution space [130]. This means that a solution for CPIT-P is only partially coded as a genotype solution and the complete solution is constructed from such partial solution composing the phenotype.

4.3.1 Representing a feasible solution for the CPIT-P

A genotype solution is a set centroids represented by integer numbers. Consequently, the length of the genotype solution is variable and the result is not affected by the order of the elements in the set. The use of a set of integer numbers instead of lists ensures that elements of the individual are not duplicated. To generate the extraction phases from the genotype solution, we generate clusters with those centroids that are close among them. This process is carried out by the clustering technique: corresponding to the K-Means algorithm [131]. An example of the genotype representation is shown in Figure 4.5, which shows a genotype solution composed of four bases of truncated cones with a constant radius r = 2. In such case, the model of blocks, defined by nx = 10 and ny = 8, is framed in a concentric circle. To the left of the figure, the precedences that define the individual structures are shown. A phenotype solution that considers the geotechnical and operational constraints, is constructed from a clustered genotype solution. The phenotype solution specifies the complete mine scheduling, i.e. extraction phases, periods and the blocks to be extracted. Thus, all blocks that belong to an extraction phase are identified by a constructive algorithm that considers the feasibility of a solution. Specifically, given a centroid, the radial base and the slope angle, both blocks belonging to the base and the precedent blocks are, identified.

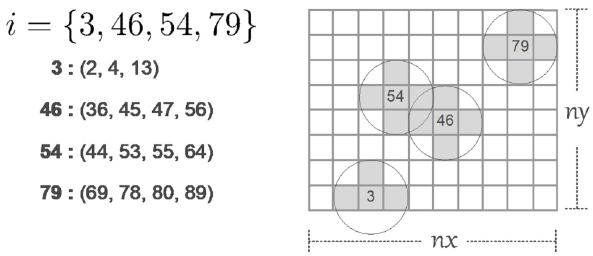
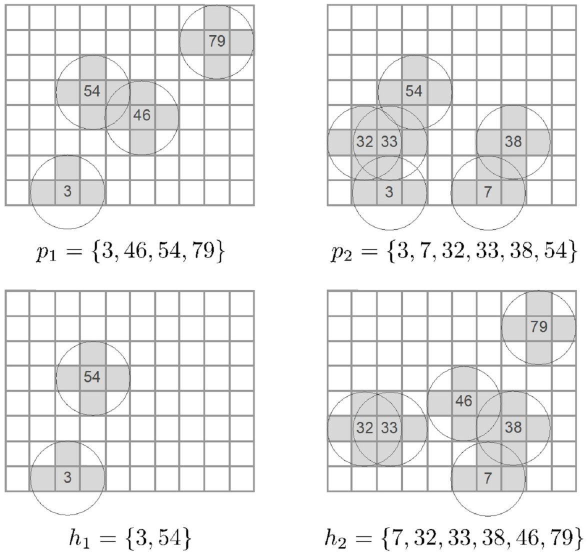


FIGURE 4.5: Representing a feasible solution (Image from [127])



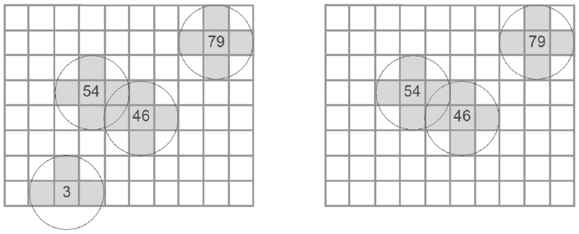
$p_1 = \{3, 46, 54, 79\}$ $p_2 = \{3, 7, 32, 33, 38, 54\}$

$h_1 = \{3, 54\}$ $h_2 = \{7, 32, 33, 38, 46, 79\}$

FIGURE 4.6: Crossover operator (Image from [127])

4.3.2 Definition of the PGA operators

Selection, crossover and mutation operators were defined for this particular algorithm PGA. To select the appropriate genotype solution for the PGA population, a standard roulette operator is used, whereas the variation operators are designed to operate on the set of integer numbers. In fact, weighed roulette selection assigns a proportional part of their fitness to each of the solutions in the population [129]. Besides, the crossover operator combines two sets, $p1$ and $p2$, and two offspring are generated. The first offspring $h1$ is defined by the intersection, which contains common elements of $p1$ and $p2$, whereas the second offspring $h2$ is obtained from the symmetrical difference of the parents. The example in Figure 4.6 considers parents $p1$ and $p2$ of length four and six, respectively, that generate two offspring ($h1$ and $h2$) with lengths of two and five. The mutation operator randomly adds or removes an element according to a determined probability. In Figure 4.7 a single mutation is shown by applying the probability of removing or adding an element by randomly selecting a position in $p1$.



$p_1 = \{3, 46, 54, 79\}$ $h_1 = \{46, 54, 79\}$

FIGURE 4.7: Mutation operator (Image from [127])

4.3.3 Fitness evaluation

Each solution evaluated by the NPV considers the blocks scheduled for each extraction phase. Because each block has an associated economic value, considering the capacities it is possible to identify an extraction period for each block. The value of each period is corrected with the discount rate. The procedure to evaluate a genotype solution is presented in Algorithm 17. The function receives the following input parameters: I the individual from the PGA; G: geometry of the block models; r: basal radius; δ: slope angle; CP: processing capacity; CM: mining capacity; and α: discount rate. From lines 1 to 3, the algorithm variables are initialized, where t is the number of periods and S is the set of blocks extracted. In line 4, the clustering function produces a set of extraction phases F, each of which contains centroids c. Then, in both cycles, each extraction phase is reviewed and a set of blocks to be scheduled is identified. In variable B (line 7) the total of the blocks of the truncated cone is obtained; this solid cone is built based on the center c for the block model G, with radius r and angle δ. Subsequently, a difference of sets is performed (line 8) between the blocks belonging to the truncated cone of phase f with base c and the blocks that have already been extracted in the previous processes. Finally, the *Scheduling* function returns the NPV generated by the extracted blocks and the number of periods used (γ, t). Lines 10 and 11 updates the extracted blocks and the total NPV of the process.

Algorithm 17 Fitness Function

```
Input:
    $I$: the individual from the PGA;
    $G$: geometry of the block models;
    $r$: basal radius;
    $\delta$: slope angle;
    $CP$: processing capacity;
    $CM$: mining capacity;
    $\alpha$: discount rate.
    ——————————————
Output: $NPV$: The net present value.
    ——————————————
$t = 0$
$NPV = 0$
$S = \emptyset$
$F = KMeans(I)$
for $f \in F$ do
    for $c \in f$ do
        $B = Precedence(c, G, r, \delta)$
        $B = B \setminus S$
        $(\gamma, t) = Scheduling(B, G, CP, CM, \alpha, t)$
        $S = S \cup B$
        $NPV = NPV + \gamma$
    end for
end for
return $NPV$
```

4.4 Computational Experiments

4.4.1 Set of instances

To study the performance of the PGA, a set of instances from Minelib (presented in Table 17) were used [57]). Each instance is a block model and specifies the amount of ore contained per block, the total tonnage of the block, and the minimum and maximum limits of operational resources for extraction and processing. A cut-off grade that determines if a material is an ore or waste is also considered. There are costs associated with sending these blocks either to processing or to the mine. Considering the extraction plan, an ad hoc discount rate (α) is used for each instance.

The PGA was implemented in Python 3.4.3 programming language and experiments have been performed on a computer with 32 Intel Xeon Haswell 2.30 GHz and 28.8 GB RAM (by using 32 threads) and a Debian GNU/Linux 9 (stretch), 64-bit operating system.

4.4.2 Tuning of the parameters

The iterated racing for automatic algorithm configuration (Irace) method was used [78]. At each iteration, the samples are updated, and the parameter values with the best performance increase their probabilities of being selected. The instances considered in order to use Irace were: Newman1, Zuck small and KD.

The parameters for PGA are given in the following:

- Clusters: Number of clusters using *k-means*.
- Size Chrom: Size of Chromosome.

- Ngen: Number of generations.
- Npop: Number of populations.
- Cx: Crossover probability.
- Mx: Mutation probability.

TABLE 4.1: Set of Instances for CPIT

Name	*Blocks*	*Precedences*	*CM*	*CP1*	*CP2*	α
*Newman*1	1,060	3,922	2,000,000	1,100,000	No	0.08
Zuck small	9,400	145,640	60,000,000	20,000,000	No	0.1
KD	14,153	219,778	∞	10,000,000	No	0.15
Zuck Medium	29,277	1,271,207	18,000,000	8,000,000	No	0.1
P4HD	40,947	738,609	52,500,000	12,500,000	No	0.15
Marvin	53,271	650,631	60,000,000	20,000,000	No	0.1
W23	74,260	764,786	68,000,000	3,610,000[a]	1,000,000[a]	0.1
Zuck Large	96,821	1,053,105	3,000,000	1,200,000	No	0.1
McLaughlin Limit	112,687	3,035,483	3,300,000	No	No	0.15

[a] : Multidimensional processing Capacity.

TABLE 4.2: Set of Parameters

N°	Clusters	*Size Chrom*	*Ngen*	*Npop*	*Cx*	*Mx*
1	50	80	60	70	0,9784	0.0216
2	20	80	80	70	0,9670	0.0330
3	20	80	80	50	0,9670	0.0330
4	20	80	60	70	0,9670	0.0330

4.5 Results

The considered problem does not have algorithms in the literature that can be consistently compared. However, we can consider the *UPIT* and *CPIT* problems to establish value comparisons and conclude methodological differences. Remember that *UPIT* only considers the flow restriction (precedence), while *CPIT* considers the period and resource restrictions (mine capacity and processing capacity). Table 4.3 shows the different values for the considered instances: the first column has the best known solution value (BKS) for the *UPIT* problem; this value is a Valid Upper Bound when the period and resource constraints are removed. The following three columns contain information about the *CPIT* extracted from [57]. The upper bound is obtained with LP relaxation of the decision variable. Column *Per* in Table 4.3 contains the maximum number of periods (or time horizon) that the CPIT problem can use. Finally, the PGA results are presented: the average number of increments, the average number of periods, the best value of the different executions, the average value, and the average time per run.

The PGA has reasonable solutions regarding the *CPIT* values for small instances. This is due to two reasons: first, the number of periods is not delimited in our algorithm and neither in the *CPIT-P*, so that, as there are more periods to extract more blocks, the higher the NPV value. Second, processing of the predecessor blocks is performed to obtain nested truncated cones. In this way, we can get different values concerning the original *CPIT* problem. Quantitative examples of this are the instances *newman1*, *Zuck small*, *KD*, and *Zuck medium*, where the best value of the PGA is always much better than the *BKS* of the *CPIT*.

The PGA does not get reasonable solutions for large instances. Even the performance of the algorithm is worst than that of the CPIT using more periods. Analyzing the geometric shape of the instances, the number of truncated cones does not always respond to all the scenarios in the same way. Thus, a calibration must be performed with all instances and not just with small instances.

The number of increments for the different instances seems to converge to the same number. This is because K-Means is a parametric algorithm and all the instances generate a similar number of clusters (increments) with respect to the generated chromosome.

The PGA is fast for some specific instances and very slow for two particular instances. As we concluded in the previous paragraphs, the construction of the truncated cones depends on the instance and its geometric shape. For two different instances like *Zuck Medium* and *McLaughlin Limit*, the average PGA evolution time exceeds 4 hours. However, for the instance *Zuck Medium*, good results are found, while for the instance *McLaughlin Limit* it seems is not highly effective algorithm.

4.6 Conclusion

An approach supported by a parallel genetic algorithm is studied for the mining scheduling problem, considering both geotechnical and operational constraints for real-size instances. This new CPIT-P variant was addressed considering a set of integer numbers as the representation. Such a representation, together with an ad hoc constructive function to evaluate each feasible solution, facilitates the data management and the geometric and mathematical operations involved. The solutions provided by PGA respect the criteria of the operational and geotechnical conditions so that robust phases are produced in consideration of the location of both ore and waste. The initial evaluation used provided a good starting point for the convergence of the algorithm, although using a different form of phase extraction could lead to better initial information for the genetic algorithm. Numerical tests demonstrated the flexibility of the presented solution, because the proposed genetic algorithm allows for solving small and large size instances, as well as providing feasible solutions to real large-scale problems

TABLE 4.3: Computacional Results for the CPIT-P with *Set*4

Instance	UPIT BKS	CPIT UB	CPIT BKS	Per	Avg.Increm.	AvgPer.	Max. NPV	Avg.NPV	Avg. Time (s)
*newman*1	26,086,899.00	24,486,184.00	23,483,671.00	6	10.20	10.50	23,895,222.12	23,651,832.89	86.86
Zuck small	1,422,726,898.00	854,182,396.00	788,652,600.00	20	18.70	27.70	1,174,577,648.38	1,154,912,694.66	2,072.51
kd	652,195,037.00	409,498,555.00	396,858,193.00	12	19.90	29.00	410,163,019.89	399,877,622.51	2,931.32
Zuck medium	1,075,124,490.00	710,641,410.00	615,411,415.00	15	17.30	26.10	895,228,926.41	883,325,871.92	27,580.19
*p*4*hd*	293,373,256.00	247,415,730.00	246,138,696.00	10	20.00	20.50	196,607,217.39	192,471,910.60	1,645.52
marvin	1,415,655,436.00	863,916,131.00	820,726,048.00	20	17.70	30.20	649,760,548.65	373,508,100.93	138.41
*w*23	510,973,998.00	400,653,199.00	392,226,063.00	12	19.20	25.90	287,892,579.19	271,295,604.61	1,945.47
Zuck large	122,220,280.00	1,648,051,083.00	1,645,242,774.00	30	20.00	28.30	71,974,729.07	67,669,018.76	5,490.10
McLaughlin Limit	1,495,726,474.00	1,078,979,501.00	1,073,327,197.00	15	20.00	21.20	969,113,623.48	935,218,413.07	16,411.67
Average					18.11	24.38	519,912,612.73	477,992,341.11	6,478.01

4.7 Current and future works

The problem does not have an integer programming mathematical formulation. It is vital to obtain good upper and lower bounds for the problem to be able to perform a better comparison of the proposed PGA.

Local search algorithms are being tested. To this end, three-dimensional vectors will be used to find better bases for the truncated cones. In this way, random crossovers and mutations can be complemented with intelligent operators either within crossover or mutation processes or local searches via Memetic Algorithms [25].

Finally, a machine learning algorithm can be used to improve the offline performance of our PGA. First, images can be generated that allow to identify of winning bases, and classification via Convolutional Neural Network [132] can catalog whether worth while to change the base for the generation of a truncated cone.

Chapter 5

Conclusion

In this thesis, different combinatorial optimization problems have been addressed:

- The TSPPD-H was solved by means of different metaheuristics during the first year of the PhD. (2019), and was compared with the algorithms proposed in [63] and [65]. The first metaheuristic approach, called Iterated Local Search with Frequency (ILS-F), is a fast method but does not get good quality solution. The second approach, called the Granular Iterated Local Search, was executed with four different configurations, and was shown to be competitive with the most effective algorithm of the literature.

- Polynomial-size formulations and relaxations were presented for the QMKP during the second year (2020). The different formulations are competitive with the algorithms of the literature. Other approaches presented in the thesis are the metaheuristic MS-ILS and matheuristics developed during the year 2021.

- A new open-pit mining problem called CPIT-P, is presented, and successfully solved with a Parallel Genetic Algorithm (PGA) in the year 2021. This metaheuristic can solve large-scale instances in reasonable computational times.

For each problem, different future works that are being developed as of the date of the presentation of this document have been proposed.

Although the general situation of the development of the Ph.D. activity was not the best due to the pandemic, the objectives of each work were successfully met. Each work was always supported by the supervisor of my Ph.D activity, independently of the situation, and they all complied with the planning indicated in the process (3 years). The knowledge acquired at the University of Bologna will represent a contribution that will be transferred to the country of the PhD student.

Appendix A

GILS - Details for the Granular parameters

A.1 Results

This section shows the detailed results of the GILS for each group of granular parameters. The results are reported in Tables A.1 to A.8 in order to corroborate the non-dominance between the parameters. In each table, the first 4 columns show: n, number of customers; l ,instance index; BKS, the best value obtained by metaheuristics proposed by [63] and [65]; and UBKS, corresponding to the best value found by all metahueristics (including GILS). For each group of parameters, the following values are reported:

- Min: minimum value for the experiment.
- %Gap : %Gap between Min and BKS with formula $((Min - BKS)/BKS) * 100$.
- %Gap Hns: %Gap between Min and BKS with formula $((Min - BKS)/Min) * 100$.
- Avg. Value : average value for the experiment.
- Avg. Time(s): : Average time for the experiment.

The tables A.1,A.2, A.3 and A.4 contain the values associated with the use of the improved dynamic programming algorithm.

The tables A.5,A.6, A.7 and A.8 contain the values using the heuristics of [63] and dynamic programming.

A.2 Wilcoxon Test

Finally, a statistical test based on the best values of each instance is applied. For this, the *Best* column values were obtained for each metaheuristic (4 columns in total). For this test, two groups of instances were generated: G1, corresponding to instances $n = \{20, 40, 60, 120\}$; and G2, corresponding to the instances $n = \{140, 160, 180, 200\}$. Note that instances with $n = \{80, 100\}$ were discarded as ALNS results are not available. Subsequently. The Wilcoxon signed-rand test is applied using two hypotheses:

- H0: AverageCost (GILS) = AverageCost (X) (null hypothesis).
- H1: AverageCost (GILS) > AverageCost (X) (alternative hypothesis).

With $X = \{$ Erdogan Series, ALNS and ILS-F $\}$. In this way, the $p - values$ must be contrasted for each test between GILS and X. $\alpha = 0.004166666$ is considered and the test is rejected when *p-value* > α. The following table summarizes the result of the statistical test applied for each metaheuristic and each group. The accepted values are black, and the final test result is in the last row.

TABLE A.9: p-values for the Wilcoxon Test for G1

	Erdogan	ILS-F	ALNS
H0	**0.000**	**0.002**	0.005
H1	**0.000**	**0.001**	**0.003**
	Better	Better	Similar

TABLE A.10: p-values for the Wilcoxon Test for G2

	Erdogan	ILS-F	ALNS
H0	**0.000**	**0.000**	0.013
H1	**0.000**	**0.000**	0.994
	Better	Better	Worse

The statistical analysis about the best values is different for G1 and G2. Indeed, for G1, GILS proves to be better than the metaheuristics of [63] and the ILS-F and has a similar performance to ALNS. However, for the G2 group, GILS is better than the [63] series and ILS-F but performs worse than ALNS. Finally, boxplots are shown in Figures A.1 and A.2.

TABLE A.1: Computational Results for the TSPPD-H with the instances from [63]

				$\alpha = 0.5$ $\beta = 0.5$ $\gamma = 0.5$					$\alpha = 0.5$ $\beta = 0.5$ $\gamma = 0.7$					$\alpha = 0.25$ $\beta = 0.75$ $\gamma = 0.5$					$\alpha = 0.75$ $\beta = 0.25$ $\gamma = 0.5$				
n	l	BKS	UBKS	Min	%Gap	% Gap Hns	Avg. Value.	Avg. Time(s)	Min	%Gap	% Gap Hns	Avg. Value.	Avg. Time(s)	Min	%Gap	% Gap Hns	Avg. Value.	Avg. Time(s)	Min	%Gap	% Gap Hns	Avg. Value.	Avg. Time(s)
20	1	633	633	633	0.00	0.00	633.00	1.13	633	0.00	0.00	633.00	1.31	633	0.00	0.00	633.00	1.32	633	0.00	0.00	633.00	1.17
20	2	584	584	584	0.00	0.00	584.00	1.14	584	0.00	0.00	584.00	1.29	584	0.00	0.00	584.00	1.29	584	0.00	0.00	584.00	1.16
20	3	573	573	573	0.00	0.00	573.00	1.16	573	0.00	0.00	573.00	1.32	573	0.00	0.00	573.00	1.37	573	0.00	0.00	573.00	1.15
20	4	706	706	706	0.00	0.00	706.00	1.15	706	0.00	0.00	706.00	1.30	706	0.00	0.00	706.00	1.36	706	0.00	0.00	706.00	1.25
20	5	501	501	501	0.00	0.00	501.00	1.15	501	0.00	0.00	501.00	1.32	501	0.00	0.00	501.00	1.31	501	0.00	0.00	501.00	1.17
20	6	578	578	578	0.00	0.00	578.00	1.14	578	0.00	0.00	578.00	1.27	578	0.00	0.00	578.00	1.26	578	0.00	0.00	578.00	1.14
20	7	612	612	612	0.00	0.00	612.00	1.12	612	0.00	0.00	612.00	1.25	612	0.00	0.00	612.00	1.29	612	0.00	0.00	612.00	1.13
20	8	567	567	567	0.00	0.00	567.00	1.09	567	0.00	0.00	567.00	1.31	567	0.00	0.00	567.00	1.23	567	0.00	0.00	567.00	1.08
20	9	604	604	604	0.00	0.00	604.00	1.39	604	0.00	0.00	604.00	1.44	604	0.00	0.00	604.00	1.53	604	0.00	0.00	604.00	1.37
20	10	565	565	565	0.00	0.00	565.00	2.16	565	0.00	0.00	565.00	1.95	565	0.00	0.00	566.20	1.81	565	0.00	0.00	565.00	1.98
Average		592.3	592.3	592.3	0.00	0.00	592.30	1.26	592.3	0.00	0.00	592.30	1.37	592.3	0.00	0.00	592.42	1.38	592.3	0.00	0.00	592.30	1.26
40	1	909.5	909.5	909.5	0.00	0.00	909.50	17.01	909.5	0.00	0.00	909.50	19.32	909.5	0.00	0.00	909.50	20.55	909.5	0.00	0.00	909.50	17.45
40	2	883	883	883	0.00	0.00	883.00	12.51	883	0.00	0.00	883.00	13.57	883	0.00	0.00	883.00	16.82	883	0.00	0.00	883.00	13.49
40	3	815.5	815.5	815.5	0.00	0.00	815.75	15.35	815.5	0.00	0.00	815.50	19.18	815.5	0.00	0.00	815.75	18.76	815.5	0.00	0.00	815.50	16.15
40	4	898	898	898	0.00	0.00	898.00	10.73	898	0.00	0.00	898.00	13.50	898	0.00	0.00	898.00	13.65	898	0.00	0.00	898.00	13.14
40	5	743.5	743.5	743.5	0.00	0.00	743.80	14.11	743.5	0.00	0.00	743.65	16.39	743.5	0.00	0.00	743.80	16.67	743.5	0.00	0.00	743.65	16.72
40	6	883.5	883.5	883.5	0.00	0.00	884.65	18.13	883.5	0.00	0.00	884.25	17.78	883.5	0.00	0.00	884.25	19.52	883.5	0.00	0.00	883.50	15.78
40	7	798.5	798.5	798.5	0.00	0.00	800.15	12.89	798.5	0.00	0.00	799.05	15.99	798.5	0.00	0.00	798.50	20.01	798.5	0.00	0.00	798.50	16.17
40	8	795	795	795	0.00	0.00	795.40	11.11	795	0.00	0.00	796.20	13.33	795	0.00	0.00	796.30	17.68	795	0.00	0.00	795.60	15.97
40	9	876.5	876.5	876.5	0.00	0.00	876.50	12.71	876.5	0.00	0.00	877.10	14.51	876.5	0.00	0.00	877.10	18.11	876.5	0.00	0.00	877.70	14.59
40	10	862.5	862.5	862.5	0.00	0.00	862.50	13.51	862.5	0.00	0.00	862.50	15.16	862.5	0.00	0.00	862.50	17.87	862.5	0.00	0.00	862.50	16.06
Average		846.55	846.55	846.55	0.00	0.00	846.93	13.81	846.55	0.00	0.00	846.88	15.87	846.55	0.00	0.00	846.87	17.96	846.55	0.00	0.00	846.75	15.55
60	1	1051	1051	1051	0.00	0.00	1060.33	39.42	1051	0.00	0.00	1060.53	45.00	1051	0.00	0.00	1061.66	45.49	1051	0.00	0.00	1061.66	37.62
60	2	1044.3	1044.3	1048.67	0.42	0.42	1051.74	45.95	1047.33	0.29	0.29	1052.30	62.86	1044.33	0.00	0.00	1049.00	60.11	1046.33	0.19	0.19	1048.97	57.21
60	3	990.36	990.36	993.667	0.33	0.33	993.77	44.38	993.667	0.33	0.33	993.67	55.83	993.667	0.33	0.33	993.77	58.52	993.667	0.33	0.33	995.33	42.82
60	4	1061.45	1061.45	1066	0.43	0.43	1066.00	47.23	1066	0.43	0.43	1066.00	53.58	1066	0.43	0.43	1066.00	57.99	1066	0.43	0.43	1066.00	41.70
60	5	986.93	986.93	989.667	0.28	0.28	989.67	40.47	989.667	0.28	0.28	989.67	47.13	989.667	0.28	0.28	989.67	54.10	989.667	0.28	0.28	989.67	37.93
60	6	1067.7	1067.67	1068.33	0.06	0.06	1079.63	72.51	1067.67	0.00	0.00	1079.40	85.82	1073.33	0.53	0.52	1082.57	81.26	1067.67	0.00	0.00	1074.87	91.19
60	7	1005.43	1005.43	1007.33	0.19	0.19	1007.33	49.40	1007.33	0.19	0.19	1007.63	55.59	1007.33	0.19	0.19	1008.00	79.98	1007.33	0.19	0.19	1007.93	55.86
60	8	1027.19	1027.19	1031	0.37	0.37	1032.57	53.14	1031	0.37	0.37	1033.63	70.25	1031	0.37	0.37	1034.20	80.87	1031	0.37	0.37	1036.33	67.42
60	9	1001.43	1001.43	1004.33	0.29	0.29	1004.33	52.58	1004.33	0.29	0.29	1004.33	46.57	1004.33	0.29	0.29	1004.33	65.31	1004.33	0.29	0.29	1005.13	54.56
60	10	1048.7	1048.67	1048.67	0.00	0.00	1049.80	63.32	1048.67	0.00	0.00	1050.77	61.95	1051.67	0.28	0.28	1054.24	63.27	1048.67	0.00	0.00	1052.27	66.82
Average		1028.449	1028.443	1030.8664	0.24	0.24	1033.52	50.84	1030.6664	0.22	0.22	1033.79	58.46	1031.2324	0.27	0.27	1034.34	64.69	1030.5664	0.21	0.21	1033.82	55.31

TABLE A.2: Computational Results for the TSPPD-H with the instances from [63]

				$\alpha = 0.5$ $\beta = 0.5$ $\gamma = 0.5$					$\alpha = 0.5$ $\beta = 0.5$ $\gamma = 0.7$					$\alpha = 0.25$ $\beta = 0.75$ $\gamma = 0.5$					$\alpha = 0.75$ $\beta = 0.25$ $\gamma = 0.5$				
n	l	BKS	UBKS	Min	%Gap	% Gap Hns	Avg. Value.	Avg. Time(s)	Min	%Gap	% Gap Hns	Avg. Value.	Avg. Time(s)	Min	%Gap	% Gap Hns	Avg. Value.	Avg. Time(s)	Min	%Gap	% Gap Hns	Avg. Value.	Avg. Time(s)
80	1	1207.25	1192.25	1192.75	-1.20		1201.13	239.98	1193	-1.18		1205.35	210.74	1192.75	-1.20		1200.10	213.45	1192.75	-1.20		1205.20	212.97
80	2	1191.5	1186	1186	-0.46		1189.48	236.16	1186	-0.46		1196.55	168.81	1187.5	-0.34		1193.10	249.75	1189	-0.21		1194.08	211.72
80	3	1170.75	1170.75	1170.75	0.00		1171.45	136.09	1170.75	0.00		1173.00	127.97	1170.75	0.00		1172.10	166.35	1170.75	0.00		1170.75	185.52
80	4	1279.5	1274	1279.5	0.00		1281.45	163.65	1274	-0.43		1282.63	178.70	1279.5	0.00		1282.73	188.73	1278.25	-0.10		1283.98	171.63
80	5	1228.75	1211.5	1213.25	-1.26		1222.83	265.24	1214	-1.20		1231.78	215.19	1212.25	-1.34		1222.13	325.46	1220	-0.71		1225.38	204.03
80	6	1278	1262.25	1262.25	-1.23		1282.30	281.78	1267.75	-0.80		1285.28	235.68	1264.25	-1.08		1285.43	267.33	1276	-0.16		1289.68	251.73
80	7	1170.75	1168.75	1170.75	0.00		1173.95	160.94	1170	-0.06		1173.43	183.83	1170.75	0.00		1173.30	170.65	1170.75	0.00		1174.55	159.11
80	8	1220	1220	1220.5	0.04		1231.03	172.79	1220	0.00		1229.90	166.69	1220	0.00		1227.73	232.85	1220	0.00		1228.75	210.45
80	9	1205	1194.25	1195.5	-0.79		1199.48	210.14	1194.75	-0.85		1199.63	165.75	1195.75	-0.77		1200.15	243.02	1194.25	-0.89		1199.80	153.64
80	10	1197.25	1197.25	1197.25	0.00		1205.53	179.94	1197.75	0.04		1208.08	150.45	1197.25	0.00		1206.75	195.68	1197.75	0.04		1201.80	216.52
Average		1214.875	1207.7	1208.85	-0.49		1215.86	204.67	1208.8	-0.49		1218.56	180.38	1209.075	-0.47		1216.35	225.33	1210.95	-0.32		1217.40	197.73
100	1	1316.4	1310	1311.4	-0.38		1322.92	514.50	1310.6	-0.44		1324.32	459.59	1316.8	0.03		1325.06	362.88	1320.6	0.32		1331.40	530.49
100	2	1354.4	1329.8	1329.8	-1.82		1342.78	416.86	1339	-1.14		1348.98	382.73	1339	-1.14		1346.22	391.95	1332.4	-1.62		1346.82	498.00
100	3	1370.4	1326	1326	-3.24		1332.40	392.28	1326	-3.24		1333.76	412.42	1326	-3.24		1342.78	397.54	1326	-3.24		1333.58	472.50
100	4	1439	1410	1419.4	-1.36		1432.50	572.83	1413.6	-1.77		1429.26	453.19	1424.8	-0.99		1433.72	433.78	1410	-2.02		1424.16	536.72
100	5	1312.6	1312.6	1312.6	0.00		1321.06	495.69	1312.6	0.00		1324.96	404.52	1315	0.18		1326.86	384.08	1317.8	0.40		1328.64	437.68
100	6	1425.6	1405.6	1405.6	-1.40		1418.26	580.21	1410.4	-1.07		1421.86	488.87	1408	-1.23		1416.58	562.06	1410.6	-1.05		1421.82	580.06
100	7	1358.8	1337.8	1341.8	-1.25		1348.42	413.19	1337.8	-1.55		1347.94	459.18	1341.8	-1.25		1348.38	399.73	1341.2	-1.30		1350.62	448.86
100	8	1408	1392.2	1407.4	-0.04		1415.04	310.70	1393.2	-1.05		1404.40	502.25	1392.2	-1.12		1408.04	390.80	1392.2	-1.12		1402.98	547.02
100	9	1381.8	1372	1377.8	-0.29		1381.46	331.30	1372.4	-0.68		1379.04	408.18	1373.6	-0.59		1380.16	357.91	1372	-0.71		1375.82	406.03
100	10	1414.6	1401.4	1401.4	-0.93		1423.44	393.91	1408.2	-0.45		1416.98	487.32	1401.6	-0.92		1428.24	392.35	1403.2	-0.81		1418.22	565.63
Average		1378.16	1359.74	1363.32	-1.07		1373.83	442.15	1362.38	-1.14		1373.15	445.82	1363.88	-1.03		1375.60	407.31	1362.6	-1.11		1373.41	502.30
120	1	1436.7	1436.7	1439	0.16	0.16	1457.97	603.31	1439.5	0.19	0.19	1457.60	746.54	1439.5	0.19	0.19	1458.88	585.25	1438	0.09	0.09	1446.40	830.38
120	2	1455.5	1451.17	1452.83	-0.18	-0.18	1476.42	536.27	1456.67	0.08	0.08	1468.90	1016.90	1451.17	-0.30	-0.30	1473.28	853.87	1467	0.79	0.78	1481.22	751.90
120	3	1465.5	1463.5	1470	0.31	0.31	1491.25	468.02	1466.17	0.05	0.05	1489.02	824.96	1463.5	-0.14	-0.14	1491.13	550.25	1471.33	0.40	0.40	1489.97	634.84
120	4	1543.7	1525.67	1525.67	-1.17	-1.18	1566.28	625.82	1526.5	-1.11	-1.13	1547.70	1117.07	1530.67	-0.84	-0.85	1554.80	1150.81	1535.17	-0.55	-0.56	1556.60	1361.01
120	5	1430.7	1425.17	1439	0.58	0.58	1445.04	649.06	1434.5	0.27	0.26	1446.87	738.14	1442.17	0.80	0.80	1447.92	524.57	1425.17	-0.39	-0.39	1446.14	616.25
120	6	1528	1525.67	1525.67	-0.15	-0.15	1554.30	873.93	1527.17	-0.05	-0.05	1543.50	1072.67	1542.5	0.95	0.94	1558.43	618.51	1543.33	1.00	0.99	1556.12	877.91
120	7	1517.7	1510.67	1523.17	0.36	0.36	1529.13	827.64	1514.17	-0.23	-0.23	1530.93	856.26	1510.67	-0.46	-0.47	1527.85	861.54	1521.5	0.25	0.25	1534.42	778.53
120	8	1504.8	1504.8	1510.33	0.37	0.37	1525.95	699.92	1513.17	0.56	0.55	1529.69	907.36	1507.33	0.17	0.17	1526.77	852.19	1510.83	0.40	0.40	1527.43	936.75
120	9	1528.9	1528.67	1536.83	0.52	0.52	1545.72	940.51	1538.17	0.61	0.60	1549.17	856.54	1528.67	-0.02	-0.02	1546.10	809.67	1528.67	-0.02	-0.02	1543.60	706.48
120	10	1540.8	1531.83	1531.83	-0.58	-0.59	1540.27	730.49	1534	-0.44	-0.44	1543.87	616.42	1533.17	-0.50	-0.50	1546.23	619.16	1534	-0.44	-0.44	1551.22	669.33
Average		1495.23	1490.385	1495.433	0.02	0.02	1513.23	695.50	1495.002	-0.01	-0.01	1510.72	875.29	1494.935	-0.01	-0.02	1513.14	742.58	1497.5	0.15	0.15	1513.31	816.34

TABLE A.3: Computational Results for the TSPPD-H with the instances from [63]

				$\alpha = 0.5$ $\beta = 0.5$ $\gamma = 0.5$					$\alpha = 0.5$ $\beta = 0.5$ $\gamma = 0.7$					$\alpha = 0.25$ $\beta = 0.75$ $\gamma = 0.5$					$\alpha = 0.75$ $\beta = 0.25$ $\gamma = 0.5$				
n	l	BKS	UBKS	Min	%Gap	% Gap Hns	Avg. Value.	Avg. Time(s)	Min	%Gap	% Gap Hns	Avg. Value.	Avg. Time(s)	Min	%Gap	% Gap Hns	Avg. Value.	Avg. Time(s)	Min	%Gap	% Gap Hns	Avg. Value.	Avg. Time(s)
140	1	1575.2	1575.2	1587	0.75	0.74	1603.15	1153.43	1580.29	0.32	0.32	1601.42	1267.53	1600	1.57	1.55	1605.63	1255.35	1592	1.07	1.06	1600.56	1563.60
140	2	1584.9	1584.9	1608.29	1.48	1.45	1622.26	1645.39	1609.43	1.55	1.52	1625.52	1742.57	1607.57	1.43	1.41	1631.01	1688.03	1599.86	0.94	0.94	1611.52	2086.86
140	3	1571.3	1567.71	1575.43	0.26	0.26	1582.84	2042.91	1577.43	0.39	0.39	1583.79	1860.79	1569	-0.15	-0.15	1583.34	1834.38	1567.71	-0.23	-0.23	1583.73	1472.84
140	4	1682.7	1682.7	1702.29	1.16	1.15	1719.73	1674.79	1688.71	0.36	0.36	1714.46	1897.38	1695.14	0.74	0.73	1714.83	2308.05	1696.14	0.80	0.79	1720.29	1839.94
140	5	1542.9	1541.14	1546.86	0.26	0.26	1562.44	1874.52	1541.14	-0.11	-0.11	1559.94	1685.99	1553.71	0.70	0.70	1565.27	1813.29	1548	0.33	0.33	1563.99	1520.55
140	6	1631.3	1631.3	1652.86	1.32	1.30	1665.16	2365.23	1636.14	0.30	0.30	1672.79	1497.79	1643.71	0.76	0.75	1670.20	2038.13	1645	0.84	0.83	1667.66	1510.18
140	7	1621.4	1621.4	1629.57	0.50	0.50	1653.31	1547.59	1636.29	0.92	0.91	1652.83	1425.19	1632.29	0.67	0.67	1649.54	2089.58	1640.29	1.17	1.15	1659.97	1588.59
140	8	1611	1611	1620	0.56	0.56	1628.54	1606.44	1623.14	0.75	0.75	1630.30	1452.11	1622.14	0.69	0.69	1633.40	1889.33	1623	0.74	0.74	1635.93	1689.56
140	9	1629.1	1629.1	1645	0.98	0.97	1652.36	1464.45	1645.86	1.03	1.02	1652.47	1385.60	1647.14	1.11	1.10	1658.29	1919.23	1649	1.22	1.21	1658.57	1672.67
140	10	1671.9	1665.29	1672.71	0.05	0.05	1685.79	1834.85	1676.71	0.29	0.29	1693.39	1640.42	1665.29	-0.40	-0.40	1690.89	1616.61	1675.43	0.21	0.21	1691.36	1592.50
Average		1612.17	1610.974	1624.001	0.73	0.72	1637.56	1720.96	1621.514	0.58	0.57	1638.69	1585.54	1623.599	0.71	0.71	1640.24	1845.20	1623.643	0.71	0.70	1639.36	1653.73
160	1	1712.6	1712.5	1726.12	0.79	0.78	1741.24	2344.71	1721.25	0.51	0.50	1738.65	2682.82	1714.75	0.13	0.13	1736.74	2996.85	1712.5	-0.01	-0.01	1739.73	2326.97
160	2	1725.7	1725.7	1739.5	0.80	0.79	1757.52	1853.57	1735.88	0.59	0.59	1754.50	2409.17	1738.12	0.72	0.71	1757.63	2438.88	1746.38	1.20	1.18	1757.51	1897.55
160	3	1684.3	1684.3	1704.75	1.21	1.20	1733.98	2057.57	1722.12	2.25	2.20	1744.71	1923.61	1729.25	2.67	2.60	1742.59	2135.61	1697.12	0.76	0.76	1738.51	1851.04
160	4	1837.5	1827.25	1845.75	0.45	0.45	1872.06	3022.94	1827.25	-0.56	-0.56	1845.08	3493.12	1830.25	-0.39	-0.40	1867.96	3635.76	1829.88	-0.41	-0.42	1862.76	3782.43
160	5	1654.1	1637	1650.62	-0.21	-0.21	1667.53	1965.38	1654.12	0.00	0.00	1673.34	1978.03	1642.12	-0.72	-0.73	1670.66	1778.02	1637	-1.03	-1.04	1668.65	2875.41
160	6	1726.7	1726.7	1763.38	2.12	2.08	1790.70	2064.83	1766.38	2.30	2.25	1789.36	2446.96	1770	2.51	2.45	1796.75	2575.89	1774.12	2.75	2.67	1796.00	2265.06
160	7	1742.4	1742.4	1769	1.53	1.50	1785.33	1410.20	1768.88	1.52	1.50	1783.14	1617.07	1769.75	1.57	1.55	1781.65	1954.14	1771.38	1.66	1.64	1777.13	2299.82
160	8	1749.2	1749.2	1763.5	0.82	0.81	1784.64	2516.15	1767	1.02	1.01	1791.94	2904.94	1769.5	1.16	1.15	1787.80	3092.81	1774	1.42	1.40	1793.70	3084.90
160	9	1786.7	1779	1779	-0.43	-0.43	1794.79	2023.76	1782.75	-0.22	-0.22	1796.04	1805.31	1794.12	0.42	0.41	1802.72	2100.39	1784	-0.15	-0.15	1789.90	2226.69
160	10	1756.3	1756.3	1757.75	0.08	0.08	1758.99	1278.25	1757.38	0.06	0.06	1759.22	1470.55	1757.75	0.08	0.08	1759.32	1755.15	1758.62	0.13	0.13	1759.61	1538.89
Average		1737.55	1734.035	1749.937	0.72	0.71	1768.68	2053.73	1750.301	0.75	0.73	1767.60	2273.16	1751.561	0.81	0.79	1770.38	2446.35	1748.5	0.63	0.62	1768.35	2414.87
180	1	1818.9	1815.56	1829	0.56	0.55	1863.36	3578.50	1833.89	0.82	0.82	1860.05	4344.92	1820.56	0.09	0.09	1861.17	5065.75	1815.56	-0.18	-0.18	1861.58	4600.04
180	2	1838.9	1838.9	1858.56	1.07	1.06	1874.02	1682.33	1871	1.75	1.72	1876.11	2889.13	1877	2.07	2.03	1878.00	2467.61	1847.22	0.45	0.45	1870.02	3056.85
180	3	1822.1	1822.1	1837.33	0.84	0.83	1851.62	3271.03	1829.67	0.42	0.41	1858.59	3994.74	1839.33	0.95	0.94	1860.91	4264.20	1832.22	0.56	0.55	1852.28	4165.81
180	4	1942.8	1942.8	1960.33	0.90	0.89	1974.45	4849.43	1962.33	1.01	1.00	1993.75	5039.77	1961.78	0.98	0.97	1994.04	5242.18	1955.22	0.64	0.64	1977.59	5001.16
180	5	1785.2	1785.2	1803.78	1.04	1.03	1819.02	2651.93	1796	0.60	0.60	1819.84	3741.85	1803.56	1.03	1.02	1825.92	3399.93	1801.56	0.92	0.91	1821.23	3457.97
180	6	1817.47	1817.47	1854.22	2.02	1.98	1860.78	1819.13	1846.89	1.62	1.59	1854.13	3308.21	1841.78	1.34	1.32	1854.86	4241.60	1831.11	0.75	0.74	1853.21	3617.20
180	7	1846.8	1831.78	1858.11	0.61	0.61	1878.31	3197.79	1851	0.23	0.23	1883.12	3360.68	1831.78	-0.81	-0.82	1868.12	4014.38	1846.44	-0.02	-0.02	1879.33	3719.02
180	8	1862.2	1862.2	1878.89	0.90	0.89	1906.71	2273.89	1879.89	0.95	0.94	1901.15	3420.40	1899.78	2.02	1.98	1909.91	2958.29	1891.67	1.58	1.56	1904.67	3099.91
180	9	1917.5	1917.5	1937.11	1.02	1.01	1944.48	2526.85	1939.67	1.16	1.14	1944.40	3280.23	1929.44	0.62	0.62	1944.85	3038.53	1941.89	1.27	1.26	1944.34	3424.54
180	10	1845.9	1845.9	1854.44	0.46	0.46	1879.23	1974.27	1854.11	0.44	0.44	1876.95	2924.44	1864.44	1.00	0.99	1880.40	3118.00	1853.89	0.43	0.43	1876.24	2421.05
Average		1849.777	1847.941	1867.177	0.94	0.93	1885.20	2782.51	1866.445	0.90	0.89	1886.81	3630.44	1866.945	0.93	0.91	1887.82	3781.05	1861.678	0.64	0.63	1884.05	3656.36

TABLE A.4: Computational Results for the TSPPD-H with the instances from [63]

				$\alpha = 0.5$ $\beta = 0.5$ $\gamma = 0.5$					$\alpha = 0.5$ $\beta = 0.5$ $\gamma = 0.7$					$\alpha = 0.25$ $\beta = 0.75$ $\gamma = 0.5$					$\alpha = 0.75$ $\beta = 0.25$ $\gamma = 0.5$				
n	*l*	BKS	UBKS	Min	%Gap	% Gap Hns	Avg. Value.	Avg. Time(s)	Min	%Gap	% Gap Hns	Avg. Value.	Avg. Time(s)	Min	%Gap	% Gap Hns	Avg. Value.	Avg. Time(s)	Min	%Gap	% Gap Hns	Avg. Value.	Avg. Time(s)
200	1	1928.6	1928.6	1960.9	1.67	1.65	1973.20	4102.21	1947.9	1.00	0.99	1968.98	4648.11	1945	0.85	0.84	1968.60	5099.32	1940.6	0.62	0.62	1967.89	5365.74
200	2	1964.7	1964.7	1997	1.64	1.62	2020.10	5180.83	1979.5	0.75	0.75	2012.82	5595.20	1984.9	1.03	1.02	2027.68	4963.56	1992.8	1.43	1.41	2020.54	6050.86
200	3	1954.3	1954.3	1979.9	1.31	1.29	1990.56	3834.65	1990.5	1.85	1.82	1993.52	3455.56	1980.2	1.33	1.31	1990.19	4098.39	1979.9	1.31	1.29	1989.75	4976.59
200	4	2041.4	2041.4	2048.7	0.36	0.36	2094.55	6034.28	2091.7	2.46	2.40	2106.22	6755.41	2049.1	0.38	0.38	2097.84	6534.01	2044.2	0.14	0.14	2091.38	6641.43
200	5	1890.4	1888	1888.8	-0.08	-0.08	1905.39	3966.07	1899.3	0.47	0.47	1909.34	3702.09	1904.7	0.76	0.75	1912.24	4656.87	1897.5	0.38	0.37	1907.97	4458.37
200	6	1948.3	1948.3	1979.4	1.60	1.57	1995.77	4037.88	1982.6	1.76	1.73	1995.99	4547.05	1983.2	1.79	1.76	1995.89	4973.84	1979.7	1.61	1.59	1995.73	4642.23
200	7	1930.4	1930.4	1964.1	1.75	1.72	1985.48	4090.13	1948.6	0.94	0.93	1982.57	4999.08	1941.5	0.58	0.57	1982.60	5698.55	1967.4	1.92	1.88	1983.07	5553.71
200	8	1995.1	1995.1	2006.8	0.59	0.58	2022.97	6294.31	2003.5	0.42	0.42	2023.96	5315.90	2007.8	0.64	0.63	2032.24	5888.56	2005.1	0.50	0.50	2025.38	4393.57
200	9	2013.7	2013.7	2058.1	2.20	2.16	2062.93	3117.16	2056.8	2.14	2.10	2062.53	3691.44	2057.1	2.16	2.11	2062.71	4176.62	2047.5	1.68	1.65	2059.22	4364.51
200	10	1927.3	1927.3	1953.3	1.35	1.33	1965.35	2452.35	1944.7	0.90	0.89	1960.41	3242.52	1946.6	1.00	0.99	1962.25	3569.29	1950.4	1.20	1.18	1964.36	2904.72
Average		1959.42	1959.18	1983.7	1.24	1.22	2001.63	4310.99	1984.51	1.27	1.25	2001.63	4595.24	1980.01	1.05	1.04	2003.22	4965.90	1980.51	1.08	1.06	2000.53	4935.17

TABLE A.5: Computational Results for the TSPPD-H with the instances from [63]

				$\alpha = 0.5$ $\beta = 0.5$ $\gamma = 0.5$					$\alpha = 0.5$ $\beta = 0.5$ $\gamma = 0.7$					$\alpha = 0.25$ $\beta = 0.75$ $\gamma = 0.5$					$\alpha = 0.75$ $\beta = 0.25$ $\gamma = 0.5$				
n	l	BKS	UBKS	Min	%Gap	% Gap Hns	Avg. Value.	Avg. Time(s)	Min	%Gap	% Gap Hns	Avg. Value.	Avg. Time(s)	Min	%Gap	% Gap Hns	Avg. Value.	Avg. Time(s)	Min	%Gap	% Gap Hns	Avg. Value.	Avg. Time(s)
20	1	633.00	633.00	633.00	0.00	0.00	633.00	0.52	633.00	0.00	0.00	633.00	0.54	633.00	0.00	0.00	633.00	0.54	633.00	0.00	0.00	633.00	0.51
20	2	584.00	584.00	584.00	0.00	0.00	584.00	0.53	584.00	0.00	0.00	584.00	0.54	584.00	0.00	0.00	584.00	0.54	584.00	0.00	0.00	584.00	0.53
20	3	573.00	573.00	573.00	0.00	0.00	573.00	0.60	573.00	0.00	0.00	573.00	0.62	573.00	0.00	0.00	573.00	0.63	573.00	0.00	0.00	573.00	0.62
20	4	706.00	706.00	706.00	0.00	0.00	706.00	0.54	706.00	0.00	0.00	706.00	0.60	706.00	0.00	0.00	706.00	0.56	706.00	0.00	0.00	706.00	0.57
20	5	501.00	501.00	501.00	0.00	0.00	501.00	0.54	501.00	0.00	0.00	501.00	0.55	501.00	0.00	0.00	501.00	0.55	501.00	0.00	0.00	501.00	0.55
20	6	578.00	578.00	578.00	0.00	0.00	578.00	0.50	578.00	0.00	0.00	578.00	0.52	578.00	0.00	0.00	578.00	0.52	578.00	0.00	0.00	578.00	0.52
20	7	612.00	612.00	612.00	0.00	0.00	612.00	0.50	612.00	0.00	0.00	612.00	0.51	612.00	0.00	0.00	612.00	0.51	612.00	0.00	0.00	612.00	0.50
20	8	567.00	567.00	567.00	0.00	0.00	567.00	0.51	567.00	0.00	0.00	567.00	0.51	567.00	0.00	0.00	567.00	0.51	567.00	0.00	0.00	567.00	0.51
20	9	604.00	604.00	604.00	0.00	0.00	604.00	0.57	604.00	0.00	0.00	604.00	0.60	604.00	0.00	0.00	604.00	0.59	604.00	0.00	0.00	604.00	0.61
20	10	565.00	565.00	565.00	0.00	0.00	565.00	0.80	565.00	0.00	0.00	565.00	0.81	565.00	0.00	0.00	565.00	0.87	565.00	0.00	0.00	565.00	0.77
Average		592.30	592.30	592.30	0.00	0.00	592.30	0.56	592.30	0.00	0.00	592.30	0.58	592.30	0.00	0.00	592.30	0.58	592.30	0.00	0.00	592.30	0.57
40	1	909.50	909.50	909.50	0.00	0.00	909.50	4.16	909.50	0.00	0.00	909.50	4.44	909.50	0.00	0.00	909.50	4.30	909.50	0.00	0.00	909.50	4.47
40	2	883.00	883.00	883.00	0.00	0.00	883.00	4.64	883.00	0.00	0.00	883.00	4.49	883.00	0.00	0.00	883.00	4.25	883.00	0.00	0.00	883.00	4.38
40	3	815.50	815.50	815.50	0.00	0.00	815.50	5.33	815.50	0.00	0.00	815.50	4.97	815.50	0.00	0.00	815.50	5.98	815.50	0.00	0.00	815.50	5.25
40	4	898.00	898.00	898.00	0.00	0.00	898.00	5.35	898.00	0.00	0.00	898.00	5.31	898.00	0.00	0.00	898.00	4.84	898.00	0.00	0.00	898.00	4.46
40	5	743.50	743.50	743.50	0.00	0.00	743.50	4.56	743.50	0.00	0.00	743.50	4.97	743.50	0.00	0.00	743.50	4.91	743.50	0.00	0.00	743.50	4.30
40	6	883.50	883.50	883.50	0.00	0.00	883.50	4.55	883.50	0.00	0.00	883.50	4.77	883.50	0.00	0.00	883.50	4.80	883.50	0.00	0.00	883.50	4.53
40	7	798.50	798.50	798.50	0.00	0.00	798.50	8.14	798.50	0.00	0.00	798.50	6.94	798.50	0.00	0.00	798.50	8.49	798.50	0.00	0.00	798.50	7.56
40	8	795.00	795.00	795.00	0.00	0.00	795.90	6.50	795.00	0.00	0.00	795.90	5.44	795.00	0.00	0.00	795.90	5.01	795.00	0.00	0.00	795.60	5.70
40	9	876.50	876.50	876.50	0.00	0.00	876.50	4.87	876.50	0.00	0.00	876.50	4.43	876.50	0.00	0.00	876.50	4.82	876.50	0.00	0.00	876.50	5.38
40	10	862.50	862.50	862.50	0.00	0.00	862.50	4.66	862.50	0.00	0.00	862.50	5.11	862.50	0.00	0.00	862.50	5.07	862.50	0.00	0.00	862.50	4.86
Average		846.55	846.55	846.55	0.00	0.00	846.64	5.28	846.55	0.00	0.00	846.64	5.09	846.55	0.00	0.00	846.64	5.25	846.55	0.00	0.00	846.61	5.09
60	1	1051.00	1051.00	1051.00	0.00	0.00	1051.00	15.44	1051.00	0.00	0.00	1055.83	15.17	1051.00	0.00	0.00	1055.00	15.71	1051.00	0.00	0.00	1054.80	18.60
60	2	1044.30	1044.30	1046.67	0.23	0.23	1047.26	14.35	1047.33	0.29	0.29	1048.03	13.50	1047.33	0.29	0.29	1047.93	14.93	1047.33	0.29	0.29	1047.33	14.13
60	3	990.36	990.36	993.67	0.33	0.33	993.67	15.57	993.67	0.33	0.33	993.67	17.06	993.67	0.33	0.33	993.67	15.33	993.67	0.33	0.33	993.67	12.83
60	4	1061.45	1061.45	1066.00	0.43	0.43	1066.00	14.73	1066.00	0.43	0.43	1066.00	15.32	1066.00	0.43	0.43	1066.00	14.83	1066.00	0.43	0.43	1066.00	14.23
60	5	986.93	986.93	989.67	0.28	0.28	989.67	10.68	989.67	0.28	0.28	989.67	11.61	989.67	0.28	0.28	989.67	11.79	989.67	0.28	0.28	989.67	11.29
60	6	1067.70	1067.67	1077.67	0.93	0.93	1079.80	23.61	1077.67	0.93	0.93	1080.03	22.33	1074.33	0.62	0.62	1081.23	23.32	1077.67	0.93	0.93	1080.93	24.48
60	7	1005.43	1005.43	1007.33	0.19	0.19	1007.33	15.57	1007.33	0.19	0.19	1007.33	17.52	1007.33	0.19	0.19	1007.33	19.62	1007.33	0.19	0.19	1007.33	16.54
60	8	1027.19	1027.19	1031.00	0.37	0.37	1031.00	15.80	1031.00	0.37	0.37	1031.00	16.80	1031.00	0.37	0.37	1031.00	19.24	1031.00	0.37	0.37	1031.00	15.45
60	9	1001.43	1001.43	1004.33	0.29	0.29	1004.96	18.03	1004.33	0.29	0.29	1004.33	20.99	1004.33	0.29	0.29	1004.33	20.07	1004.33	0.29	0.29	1004.33	18.37
60	10	1048.70	1048.67	1048.67	0.00	0.00	1049.07	17.79	1048.67	0.00	0.00	1049.30	18.14	1048.67	0.00	0.00	1048.67	17.10	1048.67	0.00	0.00	1048.84	21.48
Average		1028.45	1028.44	1031.60	0.30	0.30	1031.98	16.16	1031.67	0.31	0.31	1032.52	16.84	1031.33	0.28	0.28	1032.48	17.19	1031.67	0.31	0.31	1032.39	16.74

TABLE A.6: Computational Results for the TSPPD-H with the instances from [63]

				$\alpha = 0.5$ $\beta = 0.5$ $\gamma = 0.5$					$\alpha = 0.5$ $\beta = 0.5$ $\gamma = 0.7$					$\alpha = 0.25$ $\beta = 0.75$ $\gamma = 0.5$					$\alpha = 0.75$ $\beta = 0.25$ $\gamma = 0.5$				
n	l	BKS	UBKS	Min	%Gap	% Gap Hns	Avg. Value.	Avg. Time(s)	Min	%Gap	% Gap Hns	Avg. Value.	Avg. Time(s)	Min	%Gap	% Gap Hns	Avg. Value.	Avg. Time(s)	Min	%Gap	% Gap Hns	Avg. Value.	Avg. Time(s)
80	1	1207.25	1192.25	1192.25	-1.24		1195.58	56.46	1192.25	-1.24		1195.33	42.81	1193.25	-1.16		1197.05	40.49	1192.25	-1.24		1193.83	40.83
80	2	1191.50	1186.00	1190.50	-0.08		1194.45	34.03	1186.00	-0.46		1194.23	41.59	1190.75	-0.06		1197.68	44.35	1190.00	-0.13		1193.88	43.83
80	3	1170.75	1170.75	1170.75	0.00		1170.75	33.60	1170.75	0.00		1170.75	33.78	1170.75	0.00		1170.75	33.57	1170.75	0.00		1170.75	30.71
80	4	1279.50	1274.00	1278.00	-0.12		1286.03	33.72	1275.50	-0.31		1280.08	38.54	1278.00	-0.12		1286.03	41.22	1278.00	-0.12		1281.63	47.52
80	5	1228.75	1211.50	1213.25	-1.26		1217.88	54.90	1211.50	-1.40		1222.28	64.47	1212.25	-1.34		1221.38	69.94	1214.00	-1.20		1218.40	70.00
80	6	1278.00	1262.25	1266.00	-0.94		1276.05	53.22	1275.50	-0.20		1281.18	49.76	1266.50	-0.90		1278.65	44.87	1264.75	-1.04		1277.05	50.41
80	7	1170.75	1168.75	1170.75	0.00		1173.95	44.25	1170.25	-0.04		1174.35	36.63	1170.00	-0.06		1173.55	42.68	1168.75	-0.17		1171.50	48.73
80	8	1220.00	1220.00	1220.50	0.04		1221.73	43.55	1220.50	0.04		1224.55	49.19	1222.25	0.18		1225.68	42.45	1222.25	0.18		1226.53	46.16
80	9	1205.00	1194.25	1194.25	-0.89		1198.13	45.44	1194.25	-0.89		1196.75	50.89	1197.75	-0.60		1199.73	45.13	1194.25	-0.89		1197.80	45.04
80	10	1197.25	1197.25	1198.75	0.13		1204.08	51.35	1198.75	0.13		1208.25	38.60	1197.75	0.04		1208.68	38.02	1199.25	0.17		1201.63	45.65
Average		1214.88	1207.70	1209.50	-0.44		1213.86	45.05	1209.53	-0.44		1214.77	44.63	1209.93	-0.40		1215.92	44.27	1209.43	-0.44		1213.30	46.89
100	1	1316.40	1310.00	1310.00	-0.49		1315.32	90.09	1310.40	-0.46		1319.76	88.15	1310.60	-0.44		1320.58	83.73	1310.60	-0.44		1316.56	83.14
100	2	1354.40	1329.80	1341.60	-0.95		1347.22	100.15	1335.80	-1.37		1347.06	81.01	1340.00	-1.06		1347.66	95.46	1344.20	-0.75		1346.66	82.19
100	3	1370.40	1326.00	1335.20	-2.57		1341.36	87.24	1335.80	-2.52		1342.78	104.89	1332.80	-2.74		1341.82	81.03	1332.40	-2.77		1344.14	85.55
100	4	1439.00	1410.00	1411.60	-1.90		1420.40	108.58	1416.00	-1.60		1424.26	92.13	1422.00	-1.18		1426.50	99.00	1411.60	-1.90		1420.52	81.13
100	5	1312.60	1312.60	1318.80	0.47		1327.84	109.80	1315.40	0.21		1325.76	98.82	1322.60	0.76		1328.84	106.61	1315.40	0.21		1321.38	104.51
100	6	1425.60	1405.60	1408.40	-1.21		1413.20	111.18	1410.20	-1.08		1415.46	108.66	1408.40	-1.21		1416.60	107.15	1409.80	-1.11		1415.22	106.64
100	7	1358.80	1337.80	1342.40	-1.21		1349.06	85.60	1337.80	-1.55		1350.12	85.53	1341.80	-1.25		1350.14	79.55	1343.80	-1.10		1349.70	81.88
100	8	1408.00	1392.20	1393.20	-1.05		1396.40	82.12	1393.20	-1.05		1398.28	89.84	1397.80	-0.72		1400.60	78.79	1392.20	-1.12		1396.02	88.79
100	9	1381.80	1372.00	1372.00	-0.71		1375.58	82.79	1372.60	-0.67		1376.16	79.12	1374.40	-0.54		1377.58	75.32	1372.00	-0.71		1375.96	70.98
100	10	1414.60	1401.40	1401.80	-0.90		1406.44	125.69	1401.60	-0.92		1406.28	105.58	1402.20	-0.88		1408.74	126.33	1404.60	-0.71		1409.46	98.76
Average		1378.16	1359.74	1363.50	-1.05		1369.28	98.32	1362.88	-1.10		1370.59	93.37	1365.26	-0.93		1371.91	93.30	1363.66	-1.04		1369.56	88.36
120	1	1436.70	1436.70	1437.67	0.07	0.07	1447.23	184.13	1435.67	-0.07	-0.07	1444.98	129.67	1440.33	0.25	0.25	1448.82	150.03	1438.17	0.10	0.10	1445.32	153.23
120	2	1455.50	1451.17	1458.17	0.18	0.18	1464.58	158.85	1453.00	-0.17	-0.17	1461.35	173.23	1462.83	0.50	0.50	1470.27	177.29	1458.17	0.18	0.18	1470.67	209.09
120	3	1465.50	1463.50	1463.50	-0.14	-0.14	1478.25	163.23	1470.33	0.33	0.33	1482.13	131.77	1471.67	0.42	0.42	1476.58	163.74	1475.50	0.68	0.68	1482.55	149.89
120	4	1543.70	1525.67	1534.67	-0.58	-0.59	1540.85	181.04	1531.17	-0.81	-0.82	1544.13	192.04	1533.83	-0.64	-0.64	1545.52	193.47	1537.33	-0.41	-0.41	1543.57	180.60
120	5	1430.70	1425.17	1437.33	0.46	0.46	1442.52	164.79	1445.50	1.03	1.02	1447.72	136.77	1429.83	-0.06	-0.06	1441.52	153.05	1427.50	-0.22	-0.22	1441.12	151.91
120	6	1528.00	1525.67	1528.17	0.01	0.01	1537.28	189.73	1536.00	0.52	0.52	1546.50	186.92	1531.17	0.21	0.21	1550.82	170.60	1529.50	0.10	0.10	1542.45	173.43
120	7	1517.70	1510.67	1517.33	-0.02	-0.02	1524.55	157.58	1521.33	0.24	0.24	1526.78	124.19	1520.50	0.18	0.18	1529.37	149.29	1521.33	0.24	0.24	1524.90	161.53
120	8	1504.80	1504.80	1509.00	0.28	0.28	1518.40	140.48	1511.33	0.43	0.43	1520.12	135.25	1508.83	0.27	0.27	1520.03	129.57	1512.67	0.52	0.52	1519.75	144.76
120	9	1528.90	1528.67	1526.17	-0.18	-0.18	1533.67	142.87	1524.00	-0.32	-0.32	1539.62	152.01	1530.67	0.12	0.12	1538.60	182.41	1521.00	-0.52	-0.52	1534.12	144.40
120	10	1540.80	1531.83	1532.50	-0.54	-0.54	1539.38	122.25	1537.17	-0.24	-0.24	1540.68	130.11	1537.67	-0.20	-0.20	1542.65	122.07	1535.33	-0.36	-0.36	1542.38	135.15
Average		1495.23	1490.39	1494.45	-0.05	-0.05	1502.67	160.50	1496.55	0.09	0.09	1505.40	149.20	1496.73	0.10	0.10	1506.42	159.15	1495.65	0.03	0.03	1504.68	160.40

TABLE A.7: Computational Results for the TSPPD-H with the instances from [63]

				$\alpha = 0.5$ $\beta = 0.5$ $\gamma = 0.5$					$\alpha = 0.5$ $\beta = 0.5$ $\gamma = 0.7$					$\alpha = 0.25$ $\beta = 0.75$ $\gamma = 0.5$					$\alpha = 0.75$ $\beta = 0.25$ $\gamma = 0.5$				
n	*l*	BKS	UBKS	Min	%Gap	% Gap Hns	Avg. Value.	Avg. Time(s)	Min	%Gap	% Gap Hns	Avg. Value.	Avg. Time(s)	Min	%Gap	% Gap Hns	Avg. Value.	Avg. Time(s)	Min	%Gap	% Gap Hns	Avg. Value.	Avg. Time(s)
140	1	1575.20	1575.20	1585.86	0.68	0.67	1596.36	269.40	1585.14	0.63	0.63	1595.34	241.34	1585.86	0.68	0.67	1595.77	267.74	1590.71	0.98	0.98	1597.87	236.83
140	2	1584.90	1584.90	1601.14	1.02	1.01	1611.91	288.37	1598.29	0.84	0.84	1605.13	268.99	1598.71	0.87	0.86	1605.96	227.22	1593.57	0.55	0.54	1606.79	208.66
140	3	1571.30	1567.71	1574.00	0.17	0.17	1581.80	236.51	1572.00	0.04	0.04	1580.74	213.96	1579.43	0.52	0.51	1588.57	233.45	1570.86	-0.03	-0.03	1581.59	230.57
140	4	1682.70	1682.70	1696.43	0.82	0.81	1707.96	311.43	1685.00	0.14	0.14	1708.39	342.81	1696.86	0.84	0.83	1706.86	294.67	1691.00	0.49	0.49	1705.01	354.32
140	5	1542.90	1541.14	1558.14	0.99	0.98	1561.27	260.88	1554.00	0.72	0.71	1560.82	253.79	1547.43	0.29	0.29	1558.79	284.76	1557.29	0.93	0.92	1560.81	253.59
140	6	1631.30	1631.30	1633.86	0.16	0.16	1653.41	324.88	1644.14	0.79	0.78	1660.44	299.38	1636.71	0.33	0.33	1656.14	300.63	1640.71	0.58	0.57	1654.96	264.91
140	7	1621.40	1621.40	1634.00	0.78	0.77	1647.63	264.40	1632.14	0.66	0.66	1646.21	259.20	1623.00	0.10	0.10	1647.96	269.78	1619.43	-0.12	-0.12	1645.44	244.45
140	8	1611.00	1611.00	1620.57	0.59	0.59	1623.39	246.34	1618.14	0.44	0.44	1624.28	224.86	1623.29	0.76	0.76	1624.82	214.73	1621.43	0.65	0.64	1625.86	235.08
140	9	1629.10	1629.10	1642.71	0.84	0.83	1646.33	240.01	1641.00	0.73	0.73	1644.89	230.25	1640.14	0.68	0.67	1646.04	245.39	1643.57	0.89	0.88	1646.17	216.31
140	10	1671.90	1665.29	1671.43	-0.03	-0.03	1680.92	207.24	1677.29	0.32	0.32	1683.59	198.36	1674.71	0.17	0.17	1684.51	226.80	1678.57	0.40	0.40	1684.87	216.75
Average		1612.17	1610.97	1621.81	0.60	0.60	1631.10	264.95	1620.71	0.53	0.53	1630.98	253.29	1620.61	0.52	0.52	1631.54	256.52	1620.71	0.53	0.53	1630.94	246.15
160	1	1712.60	1712.50	1707.00	-0.33	-0.33	1724.29	297.20	1705.88	-0.39	-0.39	1722.89	308.29	1716.12	0.21	0.21	1731.62	228.06	1722.38	0.57	0.57	1728.74	321.10
160	2	1725.70	1725.70	1731.38	0.33	0.33	1743.32	414.45	1736.38	0.62	0.62	1745.49	366.46	1740.12	0.84	0.83	1747.61	396.08	1735.62	0.57	0.57	1748.14	388.13
160	3	1684.30	1684.30	1720.88	2.17	2.13	1740.83	381.37	1717.00	1.94	1.90	1731.46	467.60	1731.38	2.80	2.72	1740.39	377.87	1710.38	1.55	1.52	1730.15	461.66
160	4	1837.50	1827.25	1816.12	-1.16	-1.18	1828.35	446.21	1830.62	-0.37	-0.38	1844.86	535.46	1831.00	-0.35	-0.35	1843.39	508.01	1825.25	-0.67	-0.67	1837.98	476.09
160	5	1654.10	1637.00	1648.88	-0.32	-0.32	1659.23	358.65	1646.88	-0.44	-0.44	1657.21	361.38	1647.00	-0.43	-0.43	1656.80	365.41	1645.00	-0.55	-0.55	1663.60	369.99
160	6	1726.70	1726.70	1753.50	1.55	1.53	1770.04	378.99	1766.12	2.28	2.23	1778.75	375.11	1751.88	1.46	1.44	1766.55	447.90	1764.62	2.20	2.15	1773.76	390.02
160	7	1742.40	1742.40	1758.00	0.90	0.89	1767.36	370.07	1753.62	0.64	0.64	1769.39	329.60	1767.75	1.45	1.43	1772.42	327.19	1762.62	1.16	1.15	1770.24	317.14
160	8	1749.20	1749.20	1763.38	0.81	0.80	1776.22	333.60	1771.12	1.25	1.24	1778.24	323.75	1766.00	0.96	0.95	1778.01	323.08	1764.50	0.87	0.87	1775.36	300.32
160	9	1786.70	1779.00	1782.50	-0.24	-0.24	1789.81	425.12	1791.00	0.24	0.24	1794.15	374.60	1785.38	-0.07	-0.07	1791.71	404.58	1782.75	-0.22	-0.22	1788.33	414.57
160	10	1756.30	1756.30	1757.38	0.06	0.06	1757.58	347.67	1757.75	0.08	0.08	1758.04	318.09	1757.38	0.06	0.06	1757.79	335.29	1757.38	0.06	0.06	1757.99	331.85
Average		1737.55	1734.04	1743.90	0.38	0.37	1755.70	375.33	1747.64	0.59	0.57	1758.05	376.03	1749.40	0.69	0.68	1758.63	371.35	1747.05	0.55	0.54	1757.43	377.09
180	1	1818.90	1815.56	1822.33	0.19	0.19	1838.76	545.11	1818.11	-0.04	-0.04	1835.38	468.88	1830.11	0.62	0.61	1851.81	456.48	1838.22	1.06	1.05	1848.17	480.24
180	2	1838.90	1838.90	1849.89	0.60	0.59	1861.50	560.28	1860.00	1.15	1.13	1868.08	516.50	1861.11	1.21	1.19	1867.52	548.52	1855.11	0.88	0.87	1867.32	522.89
180	3	1822.10	1822.10	1827.67	0.31	0.30	1848.33	573.05	1827.89	0.32	0.32	1851.28	475.92	1824.33	0.12	0.12	1845.38	523.69	1837.33	0.84	0.83	1847.84	514.89
180	4	1942.80	1942.80	1949.22	0.33	0.33	1960.40	741.92	1941.67	-0.06	-0.06	1964.71	549.11	1951.89	0.47	0.47	1968.85	648.98	1943.67	0.04	0.04	1963.33	516.32
180	5	1785.20	1785.20	1793.11	0.44	0.44	1801.65	676.87	1807.56	1.25	1.24	1812.25	591.13	1803.67	1.03	1.02	1807.65	605.01	1794.44	0.52	0.51	1808.94	570.71
180	6	1817.47	1817.47	1829.11	0.64	0.64	1847.75	509.38	1841.00	1.29	1.28	1849.80	409.80	1838.56	1.16	1.15	1849.20	477.26	1842.78	1.39	1.37	1848.13	465.03
180	7	1846.80	1831.78	1848.67	0.10	0.10	1860.05	564.96	1857.44	0.58	0.57	1870.31	529.40	1851.56	0.26	0.26	1866.47	554.03	1847.33	0.03	0.03	1863.42	555.14
180	8	1862.20	1862.20	1875.89	0.74	0.73	1886.43	547.59	1882.67	1.10	1.09	1889.86	500.02	1885.00	1.22	1.21	1890.38	477.33	1879.22	0.91	0.91	1889.42	541.37
180	9	1917.50	1917.50	1930.67	0.69	0.68	1938.58	627.98	1936.33	0.98	0.97	1939.51	532.26	1934.00	0.86	0.85	1938.93	525.42	1931.89	0.75	0.74	1939.53	521.90
180	10	1845.90	1845.90	1857.22	0.61	0.61	1864.19	494.00	1857.11	0.61	0.60	1870.80	467.68	1857.22	0.61	0.61	1870.31	471.19	1861.89	0.87	0.86	1870.08	449.91
Average		1849.78	1847.94	1858.38	0.46	0.46	1870.76	584.11	1862.98	0.72	0.71	1875.20	504.07	1863.75	0.76	0.75	1875.65	528.79	1863.19	0.73	0.72	1874.62	513.84

TABLE A.8: Computational Results for the TSPPD-H with the instances from [63]

				$\alpha = 0.5$ $\beta = 0.5$ $\gamma = 0.5$					$\alpha = 0.5$ $\beta = 0.5$ $\gamma = 0.7$					$\alpha = 0.25$ $\beta = 0.75$ $\gamma = 0.5$					$\alpha = 0.75$ $\beta = 0.25$ $\gamma = 0.5$				
n	*l*	BKS	UBKS	Min	%Gap	% Gap Hns	Avg. Value.	Avg. Time(s)	Min	%Gap	% Gap Hns	Avg. Value.	Avg. Time(s)	Min	%Gap	% Gap Hns	Avg. Value.	Avg. Time(s)	Min	%Gap	% Gap Hns	Avg. Value.	Avg. Time(s)
200	1	1928.60	1928.60	1959.10	1.58	1.56	1967.37	895.12	1958.90	1.57	1.55	1969.22	840.09	1957.10	1.48	1.46	1967.09	844.66	1960.00	1.63	1.60	1967.65	799.88
200	2	1964.70	1964.70	1973.80	0.46	0.46	1990.91	902.99	1986.20	1.09	1.08	2007.01	769.80	1996.50	1.62	1.59	2005.77	912.51	1991.20	1.35	1.33	2003.54	862.49
200	3	1954.30	1954.30	1968.40	0.72	0.72	1975.40	734.50	1972.40	0.93	0.92	1980.17	627.92	1972.40	0.93	0.92	1982.13	655.51	1974.20	1.02	1.01	1980.64	639.57
200	4	2041.40	2041.40	2052.70	0.55	0.55	2070.05	802.85	2077.70	1.78	1.75	2083.08	722.13	2058.00	0.81	0.81	2079.07	700.23	2066.10	1.21	1.20	2077.64	749.64
200	5	1890.40	1888.00	1899.90	0.50	0.50	1905.29	936.95	1902.00	0.61	0.61	1908.96	918.75	1898.90	0.45	0.45	1904.97	899.32	1901.80	0.60	0.60	1907.88	877.52
200	6	1948.30	1948.30	1977.80	1.51	1.49	1982.28	767.20	1973.10	1.27	1.26	1984.09	653.73	1978.60	1.56	1.53	1987.20	650.99	1977.40	1.49	1.47	1987.33	668.49
200	7	1930.40	1930.40	1937.40	0.36	0.36	1963.23	839.47	1948.20	0.92	0.91	1966.85	961.68	1950.80	1.06	1.05	1975.37	920.50	1966.50	1.87	1.84	1971.27	874.32
200	8	1995.10	1995.10	1997.40	0.12	0.12	2009.43	724.29	2010.10	0.75	0.75	2027.68	579.10	2006.90	0.59	0.59	2016.74	610.86	2017.40	1.12	1.11	2023.06	510.58
200	9	2013.70	2013.70	2048.20	1.71	1.68	2053.09	792.22	2044.90	1.55	1.53	2053.55	821.42	2045.80	1.59	1.57	2053.72	745.88	2043.80	1.49	1.47	2051.28	733.24
200	10	1927.30	1927.30	1939.40	0.63	0.62	1951.38	561.05	1946.20	0.98	0.97	1954.78	457.31	1947.10	1.03	1.02	1955.22	451.58	1947.80	1.06	1.05	1958.00	484.63
Average		1959.42	1959.18	1975.41	0.82	0.81	1986.84	795.66	1981.97	1.15	1.13	1993.54	735.19	1981.21	1.11	1.10	1992.73	739.20	1984.62	1.28	1.27	1992.83	720.04

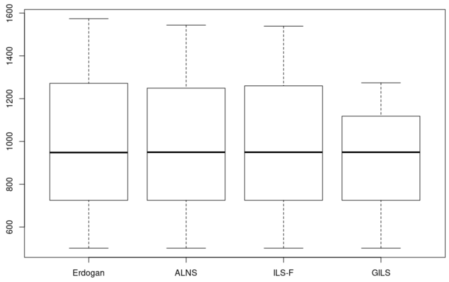


FIGURE A.1: Group G1

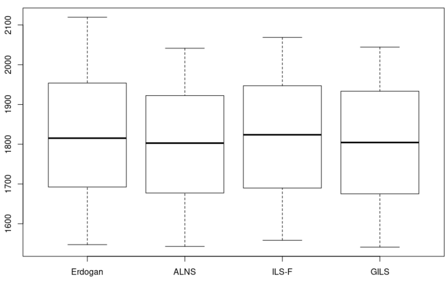


FIGURE A.2: Group G2

Appendix B

MS-ILS

B.1 Repair procedure

Observations about Algorithm Repair:

- The function $Criteria$ returns true if $infeasibleM < infMeasure$. If $infeasibleM = infMeasure$, then the function returns true if $newGain > gain$. In any other case, it returns false.

- The ternary expression $var1 = ((exp1)?(exp2) : (exp3))$ is a simplified statement. If $exp1$ is true, the expression $(exp2)$ is saved in the variable $var1$, otherwise the expression $(exp3)$ is saved in the variable $var1$.

- The function $ApplyMove$ applies the moveType ($moveType$ = 1 is extraction, $moveType$ = 2 is relocation, and $moveType = 3$ is exchange) with the input parameters to the $xInfeasible$ matrix solution.

- The function $Update$ is a simplified way of indicating that the variables $R, Items, totalOverL, xProfit$ are updated.

Algorithm 18 Repair procedure

```
**Input:** $xInfeasible, xFeasible, NiterRep$
**Output:** $xInfeasible2$
$feasible = false$
$Items = R = \emptyset$
$totalOverL = repairCount = 0$
**for** $k \in \{1, 2...K\}$ **do**
    **for** $i \in \{1, 2...N\}$ **do**
        **if** $xInfeasible_{k,i} == 1$ **then**
            $R_k = R_k + w_i$
            $Items_k = Items_k \cup i$
        **end if**
    **end for**
    $R_k = C_k - R_k$
    $totalOverL = totalOverL + ((R_k < 0)?(-R_k) : 0);$
**end for**
$xProfit = f(xInfeasible)$
**while** $feasible == false$ and $repairCount < NiterRep$ **do**
    $moveType = elem1 = elem2 = knap1 = knap2 = gain = -1$
    $infMeasure = \infty$ //*************Extraction*******************
    **for** $k \in \{1, 2...K\}$ **do**
        **for** $i \in Items_k$ **do**
            $newGain \ \ = xProfit - \Delta(i, k)$
            $infeasible1 = ((R_k + w_i) < 0)? - (R_k + w_i) : 0$
            $Rwk \ \ \ = (R_k < 0)? - (R_k) : 0$
            $infeasibleM = totalOverL - (Rwk) + infeasible1$
            **if** $Criteria(infeasibleM, infMeasure, newGain, gain)$ **then**
                $moveType = 1;$
                $elem1 = i;$
                $knap1 = k;$
                $gain = newGain;$
                $infMeasure = infeasibleM$
            **end if**
        **end for**
    **end for**
    //*************re-allocation*************
    **for** $k \in \{1, 2...K\}$ **do**
        **for** $i \in Items_k$ **do**
            **for** $kaux \in \{1, 2...K\}$ **do**
                **if** kaux!=k **then**
                    $newGain \ \ \ = xProfit + \Delta(i, kaux) - \Delta(i, k)$
                    $infeasible1 = ((R_k + w_i) < 0)? - (R_k + w_i) : 0$
                    $infeasible2 = ((R_{kaux} - w_i) < 0)? - (R_{kaux} - w_i) : 0$
                    $Rwk \ \ \ = (R_k < 0)? - (R_k) : 0$
                    $Rwaux \ \ \ = (R_{kaux} < 0)? - (R_{kaux}) : 0$
                    $infeasibleM = totalOverL - (Rwk + Rwaux) + infeasible1 + infeasible2$
                    **if** $Criteria(infeasibleM, infMeasure, newGain, gain)$ **then**
                        $moveType = 2;$
                        $elem1 = i;$
                        $knap1 = k;$
                        $knap2 = kaux;$
                        $gain = newGain;$
                        $infMeasure = infeasibleM$
                    **end if**
                **end if**
            **end for**
        **end for**
    **end for**//*************exchange*************
    **for** $k \in \{1, 2...K\}$ **do**
        **for** $i1 \in Items_k$ **do**
            **for** $kaux \in \{1, 2...K\}$ **do**
                **for** $i2 \in Items_{kaux}$ **do**
                    **if** $kaux$!=k **then**
                        $newGain \ \ \ = xProfit + \Delta(i1, kaux) - \Delta(i1, k) + \Delta(i2, k) - \Delta(i2, kaux) - 2p_{i1,i2}$
                        $infeasible1 = ((R_k + w_{i1} - w_{i2}) < 0)? - (R_k + w_{i1} - w_{i2}) : 0$
                        $infeasible2 = ((R_{kaux} + w_{i2} - w_{i1}) < 0)? - (R_{kaux} + w_{i2} - w_{i1}) : 0$
                        $Rwk \ \ \ = (R_k < 0)? - (R_k) : 0$
                        $Rwaux \ \ \ = (R_{kaux} < 0)? - (R_{kaux}) : 0$
                        $infeasibleM = totalOverL - (Rwk + Rwaux) + infeasible1 + infeasible2$
                        **if** $Criteria(infeasibleM, infMeasure, newGain, gain)$ **then**
                            $moveType = 3;$
                            $elem1 = i1;$
                            $elem2 = i2;$
                            $knap1 = k;$
                            $knap2 = kaux;$
                            $gain = newGain;$
                            $infMeasure = infeasibleM$
                        **end if**
                    **end if**
                **end for**
            **end for**
        **end for**
    **end for**
    $ApplyMove(moveType, xInfeasible, elem1, elem2, knap1, knap2, gain, infMeasure)$
    $Update(R, Items, totalOverL, xProfit)$
    $repairCount = repairCount + 1$
    **if** $isFeasible(xInfeasible2)$ **then**
        $feasible = True$
        break;
    **end if**
**end while**
**if** $feasible$ **then**
    return $xInfeasible2$
**else**
    return $xFeasible$
**end if**
```

B.2 Wilcoxon signed-rand test

TABLE B.1: % Gap between BKS and Average for the instances with $n = 100$

n	d	m	l	MS-ILS	HTS	TIG	SO	IRTS
100	25	3	1	0.95421	0.00000	0.88131	0.28785	0.00000
100	25	3	2	1.41817	0.00000	0.07125	0.00941	0.00000
100	25	3	3	1.04732	0.00000	0.60010	0.01398	0.00000
100	25	3	4	1.16987	0.00000	0.00000	0.04284	0.00000
100	25	3	5	0.60985	0.00000	0.02391	0.25104	0.00000
100	25	5	1	1.03937	0.00797	1.35964	0.78606	0.22284
100	25	5	2	1.37809	0.07418	0.25802	0.37583	0.17048
100	25	5	3	1.11069	0.00000	0.65775	0.40492	0.01436
100	25	5	4	1.17961	0.00000	0.00261	0.07520	0.00045
100	25	5	5	1.67636	0.00000	0.02377	0.47072	0.05806
100	25	10	1	1.27335	0.11837	1.00734	1.38197	0.12619
100	25	10	2	1.43185	0.00000	0.90650	1.61529	0.21879
100	25	10	3	2.09151	0.23447	1.27802	1.86622	0.50245
100	25	10	4	1.31759	0.00000	0.07725	0.60763	0.00000
100	25	10	5	1.69222	0.00000	0.89097	1.50796	0.21532
100	75	3	1	0.34326	0.00000	0.06002	0.06002	0.00000
100	75	3	2	0.42674	0.00000	0.00000	0.00950	0.00633
100	75	3	3	0.12952	0.00000	0.02295	0.02760	0.00000
100	75	3	4	0.43254	0.00000	0.00000	0.00000	0.00000
100	75	3	5	0.31237	0.00000	0.01535	0.05708	0.00000
100	75	5	1	0.50009	0.00000	0.25374	0.36861	0.11133
100	75	5	2	0.53917	0.02834	0.26964	0.35101	0.10000
100	75	5	3	0.28147	0.00000	0.04289	0.27663	0.00000
100	75	5	4	0.53149	0.00000	0.55606	0.24241	0.20798
100	75	5	5	0.74836	0.00000	0.03651	0.07101	0.00800
100	75	10	1	1.66573	0.20993	1.30433	1.31436	0.18418
100	75	10	2	1.04992	0.48162	0.76217	0.93889	0.35633
100	75	10	3	1.38709	0.03912	0.82252	0.90009	0.04430
100	75	10	4	1.44355	0.16088	0.85064	0.88376	0.17474
100	75	10	5	1.64831	0.18389	1.51080	1.48025	0.15898

TABLE B.2: p-values for the Wilcoxon Test for the instances, with $n = 100$.

Hip.	HTS	TIG	SO	IRTS
H0	1.86E-09	1.64E-07	8.01E-08	1.86E-09
H1	1	1	1	1

TABLE B.3: % Gap between BKS and Average for the instances with $n = 200$

n	d	m	l	MS-ILS	HTS	TIG	SO	IRTS
200	25	3	1	1.07873	0.07391	1.24568	0.68492	0.02957
200	25	3	2	0.32753	0.00000	0.13339	0.27325	0.00000
200	25	3	3	0.60585	0.00956	0.13768	0.21417	0.02868
200	25	3	4	1.74528	0.03296	1.29744	0.96419	0.03795
200	25	3	5	0.57814	0.00000	0.42420	0.50728	0.00098
200	25	5	1	1.08525	0.18976	1.66812	1.63069	0.09111
200	25	5	2	1.16621	0.00000	0.71679	0.98534	0.01200
200	25	5	3	1.12521	0.00000	0.41313	0.60503	0.01800
200	25	5	4	1.76470	0.15619	1.55867	1.53089	0.10616
200	25	5	5	1.08746	0.00300	0.92006	1.01827	0.01616
200	25	10	1	2.82026	0.34077	1.90198	2.38860	0.25720
200	25	10	2	1.82665	0.52945	1.37279	1.82163	0.29865
200	25	10	3	2.35012	0.15500	1.65153	2.22534	0.16714
200	25	10	4	2.65009	0.39843	1.98725	2.53242	0.43624
200	25	10	5	1.59042	0.02947	1.42597	2.28623	0.16553
200	75	3	1	0.53803	0.01367	0.00000	0.00776	0.01219
200	75	3	2	0.31335	0.00000	0.07346	0.13876	0.00583
200	75	3	3	0.52037	0.00000	0.00000	0.01518	0.05295
200	75	3	4	0.75235	0.04535	0.12510	0.17733	0.04696
200	75	3	5	0.70358	0.00000	0.00000	0.00000	0.01001
200	75	5	1	1.13907	0.02696	0.38762	0.45932	0.31753
200	75	5	2	0.85034	0.00000	0.11096	0.22364	0.08465
200	75	5	3	0.88819	0.02249	0.14295	0.22594	0.05354
200	75	5	4	1.42849	0.13402	0.39188	0.53607	0.23633
200	75	5	5	0.79577	0.02431	0.15933	0.24520	0.04811
200	75	10	1	1.32262	0.23208	0.87713	0.94067	0.45445
200	75	10	2	1.55494	0.15006	0.84467	0.95509	0.49264
200	75	10	3	1.57043	0.22723	0.80968	0.99512	0.42922
200	75	10	4	2.18701	0.46189	1.09665	1.43295	0.46063
200	75	10	5	1.49234	0.31455	0.88910	1.08943	0.30859

TABLE B.4: p-values for the Wilcoxon Test for the instances with $n = 200$.

Hip.	HTS	TIG	SO	IRTS
H0	1.86E-09	4.71E-07	2.69E-05	1.86E-09
H1	1	0.9999	0.9999	1

A statistical test was performed using the % gap between the BKS and the Average for each instance (n, m, d, l). Tables B.1 and B.3 contain the values for each instance (rows) and for each metaheuristic (columns). These tables are normalized to be able to apply a statistical test and check the performance of the MS-ILS.

The Wilcoxon test is applied to the data in Tables B.2 and B.4. For this test, we consider two hypotheses:

- H0: AverageCost (MSILS) = AverageCost (X) (null hypothesis).
- H1: AverageCost (MSILS) > AverageCost (X) (alternative hypothesis).

With $X = \{$ HTS , TIG , SO , IRTS $\}$ With significance level $\alpha = 0.05$. Looking at the $p - values$ in Tables A.2 and A4, all algorithms are better than MS-ILS (H0 is rejected and H1 is not rejected).

Figures A.1 and A.2 show the average performance against the BKS for each metaheuristic using Tables A1 and A3. HTS and IRTS are the best algorithms to date and show to be much more stable than any other algorithm (HTS for the smallest instances finds the BKS for any seed in 25 cases for the first group and 8 cases for the second group).

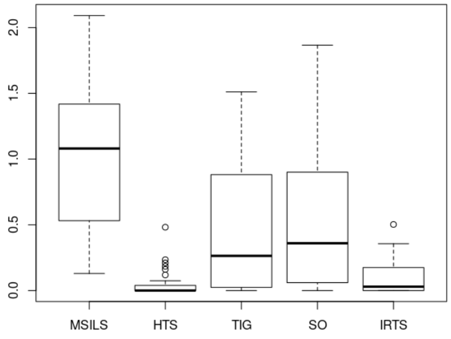


FIGURE B.1: Box plot about the performance of the metaheuristics for the instances with $n = 100$

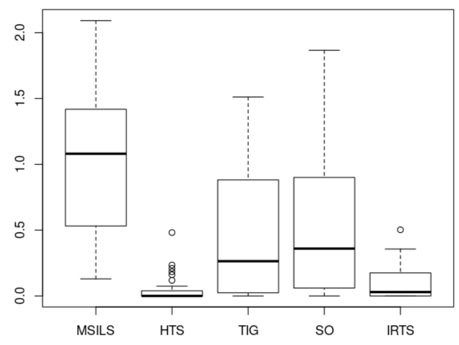


FIGURE B.2: Box plot about the performance of the metaheuristics for the instances with $n = 200$

Bibliography

[1] Joseph F McCloskey. "OR Forum—British Operational Research in World War II". In: *Operations research* 35.3 (1987), pp. 453–470.

[2] Hamdy A Taha. *Operations research an introduction*. Pearson Education Limited 2017, 2017.

[3] Informs. *What is O.R.?* https://www.informs.org/Explore/What-is-O.R.-Analytics/. 2021. (Visited on 04/29/2021).

[4] Alexander Schrijver. *Combinatorial optimization: polyhedra and efficiency*. Springer Science & Business Media, 2003.

[5] Sanjeev Arora and Boaz Barak. *Computational complexity: a modern approach*. Cambridge University Press, 2009.

[6] David L Applegate, Robert E Bixby, Vašek Chvátal, and William J Cook. *The traveling salesman problem*. Princeton university press, 2011.

[7] Martin Grötschel, Michael Jünger, and Gerhard Reinelt. "Optimal control of plotting and drilling machines: a case study". In: *Zeitschrift für Operations Research* 35.1 (1991), pp. 61–84.

[8] Robert D Plante, Timothy J Lowe, and R Chandrasekaran. "The product matrix traveling salesman problem: an application and solution heuristic". In: *Operations Research* 35.5 (1987), pp. 772–783.

[9] Robert G Bland and David F Shallcross. "Large travelling salesman problems arising from experiments in X-ray crystallography: a preliminary report on computation". In: *Operations Research Letters* 8.3 (1989), pp. 125–128.

[10] Marshall Fisher. "Vehicle routing". In: *Handbooks in operations research and management science* 8 (1995), pp. 1–33.

[11] Stephen A Cook. "The complexity of theorem-proving procedures". In: *Proceedings of the third annual ACM symposium on Theory of computing*. 1971, pp. 151–158.

[12] Fereshteh Vaezi, Seyed Jafar Sadjadi, and Ahmad Makui. "A portfolio selection model based on the knapsack problem under uncertainty". In: *PloS one* 14.5 (2019), e0213652.

[13] Hans Kellerer, Ulrich Pferschy, and David Pisinger. "Multidimensional knapsack problems". In: *Knapsack problems*. Springer, 2004, pp. 235–283.

[14] Reuters. *Chile's Cochilco sees 2020 copper production and price rising*. https://www.reuters.com/article/chile-copper-idINL1N2I9124. 2020. (Visited on 04/29/2021).

[15] Jinseok Hong, Minyoung Lee, Taesu Cheong, and Hong Chul Lee. “Routing for an on-demand logistics service”. In: *Transportation Research Part C: Emerging Technologies* 103 (2019), pp. 328–351.

[16] Amazon. *Amazon Hub Locker - General Information*. `https://www.amazon.com/b/?node=6442600011`. 2021. (Visited on 04/29/2021).

[17] Charles Darwin. *On the origin of species*. 1859.

[18] Leonora Bianchi, Marco Dorigo, Luca Maria Gambardella, and Walter J Gutjahr. “A survey on metaheuristics for stochastic combinatorial optimization”. In: *Natural Computing* 8.2 (2009), pp. 239–287.

[19] El-Ghazali Talbi. *Metaheuristics: from design to implementation*. John Wiley & Sons, 2009.

[20] L. J. Fogel. “Toward inductive inference automata”. In: *In Proceedings of the International Federation for Information Processing Congress, Munich* (1962), 395–399.

[21] John H Holland. “Outline for a logical theory of adaptive systems”. In: *Journal of the ACM (JACM)* 9.3 (1962), pp. 297–314.

[22] Vladimír Černỳ. “Thermodynamical approach to the traveling salesman problem: An efficient simulation algorithm”. In: *Journal of optimization theory and applications* 45.1 (1985), pp. 41–51.

[23] Fred Glover. “Future paths for integer programming and links to artificial intelligence”. In: *Computers & operations research* 13.5 (1986), pp. 533–549.

[24] Thomas A Feo and Mauricio GC Resende. “A probabilistic heuristic for a computationally difficult set covering problem”. In: *Operations research letters* 8.2 (1989), pp. 67–71.

[25] Pablo Moscato. “On evolution, search, optimization, genetic algorithms and martial arts: Towards memetic algorithms”. In: *Caltech concurrent computation program, C3P Report* 826 (1989), p. 1989.

[26] Olivier Martin, Steve W Otto, and Edward W Felten. *Large-step Markov chains for the traveling salesman problem*. Citeseer, 1991.

[27] Helena R Lourenço, Olivier C Martin, and Thomas Stützle. “Iterated local search”. In: *Handbook of metaheuristics*. Springer, 2003, pp. 320–353.

[28] Nenad Mladenovic. “A variable neighborhood algorithm-a new metaheuristic for combinatorial optimization”. In: *papers presented at Optimization Days*. Vol. 12. 1995.

[29] M. Dorigo. “Optimization, learning and natural algorithms”. PhD thesis. Politecnico di Milano, Italy, 1992.

[30] James Kennedy and Russell Eberhart. “Particle swarm optimization”. In: *Proceedings of ICNN’95-international conference on neural networks*. Vol. 4. IEEE. 1995, pp. 1942–1948.

[31] John R Koza. *Genetic programming: on the programming of computers by means of natural selection*. MIT press, 1992.

[32] Paolo Toth and Daniele Vigo. “The granular tabu search and its application to the vehicle-routing problem”. In: *Informs Journal on computing* 15.4 (2003), pp. 333–346.

[33] Viet-Phuong Nguyen, Christian Prins, and Caroline Prodhon. “A multi-start iterated local search with tabu list and path relinking for the two-echelon location-routing problem”. In: *Engineering Applications of Artificial Intelligence* 25.1 (2012), pp. 56–71.

[34] Michael Held and Richard M Karp. “The traveling-salesman problem and minimum spanning trees”. In: *Operations Research* 18.6 (1970), pp. 1138–1162.

[35] Gerhard Reinelt. “TSPLIB—A traveling salesman problem library”. In: *ORSA journal on computing* 3.4 (1991), pp. 376–384.

[36] Shen Lin and Brian W Kernighan. “An effective heuristic algorithm for the traveling-salesman problem”. In: *Operations research* 21.2 (1973), pp. 498–516.

[37] Keld Helsgaun. “An effective implementation of the Lin–Kernighan traveling salesman heuristic”. In: *European Journal of Operational Research* 126.1 (2000), pp. 106–130.

[38] Keld Helsgaun. *LKH Version 2.0.9 (July 2018)*. http://webhotel4.ruc.dk/~keld/research/LKH/. 2018. (Visited on 07/01/2018).

[39] Gerardo Berbeglia, Jean-François Cordeau, Irina Gribkovskaia, and Gilbert Laporte. “Static pickup and delivery problems: a classification scheme and survey”. In: *Top* 15.1 (2007), pp. 1–31.

[40] Çağrı Koç, Gilbert Laporte, and İlknur Tükenmez. “A Review on Vehicle Routing with Simultaneous Pickup and Delivery”. In: *Computers & Operations Research* (2020), p. 104987.

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

[42] Maria Battarra, Güneş Erdoğan, Gilbert Laporte, and Daniele Vigo. “The traveling salesman problem with pickups, deliveries, and handling costs”. In: *Transportation Science* 44.3 (2010), pp. 383–399.

[43] David Pisinger and Paolo Toth. “Knapsack problems”. In: *Handbook of combinatorial optimization*. Springer, 1998, pp. 299–428.

[44] PC Gilmore and Ralph E Gomory. “The theory and computation of knapsack functions”. In: *Operations Research* 14.6 (1966), pp. 1045–1074.

[45] Peter J Kolesar. “A branch and bound algorithm for the knapsack problem”. In: *Management science* 13.9 (1967), pp. 723–735.

[46] David S Johnson. “Approximation algorithms for combinatorial problems”. In: *Journal of computer and system sciences* 9.3 (1974), pp. 256–278.

[47] Oscar H Ibarra and Chul E Kim. “Fast approximation algorithms for the knapsack and sum of subset problems”. In: *Journal of the ACM (JACM)* 22.4 (1975), pp. 463–468.

[48] Silvano Martello and Paolo Toth. “Upper bounds and algorithms for hard 0-1 knapsack problems”. In: *Operations Research* 45.5 (1997), pp. 768–778.

[49] Silvano Martello and Paolo Toth. “Knapsack problems: algorithms and computer implementations”. In: *Wiley-Interscience series in discrete mathematics and optimiza tion* (1990).

[50] H. Kellerer, U. Pferschy, and D. Pisinger. *Knapsack Problems*. Springer Nature Book Archives Millennium. Springer, 2004.

[51] David Pisinger. “Where are the hard knapsack problems?” In: *Computers & Operations Research* 32.9 (2005), pp. 2271–2284.

[52] Nicolás Acevedo, Carlos Rey, Carlos Contreras-Bolton, and Victor Parada. “Automatic design of specialized algorithms for the binary knapsack problem”. In: *Expert Systems with Applications* 141 (2020), p. 112908.

[53] Giorgio Gallo, Peter L Hammer, and Bruno Simeone. “Quadratic knapsack problems”. In: *Combinatorial optimization*. Springer, 1980, pp. 132–149.

[54] Christoph Witzgall. “Mathematical methods of site selection for Electronic Message Systems (EMS)”. In: *NASA STI/Recon Technical Report N* 76 (1975), p. 18321.

[55] John MW Rhys. “A selection problem of shared fixed costs and network flows”. In: *Management Science* 17.3 (1970), pp. 200–207.

[56] Amanda Hiley and Bryant A Julstrom. “The quadratic multiple knapsack problem and three heuristic approaches to it”. In: *Proceedings of the 8th annual conference on Genetic and evolutionary computation*. 2006, pp. 547–552.

[57] Daniel Espinoza, Marcos Goycoolea, Eduardo Moreno, and Alexandra Newman. “MineLib: a library of open pit mining problems”. In: *Annals of Operations Research* 206.1 (2013), pp. 93–114.

[58] Saber Elsayed, Ruhul Sarker, Daryl Essam, and Carlos A Coello Coello. “Evolutionary approach for large-Scale mine scheduling”. In: *Information Sciences* 523 (2020), pp. 77–90.

[59] Laura Galli, Silvano Martello, Carlos Rey, and Paolo Toth. “Polynomial-size formulations and relaxations for the quadratic multiple knapsack problem”. In: *European Journal of Operational Research* 291.3 (2021), pp. 871–882.

[60] Jan Dethloff. “Vehicle routing and reverse logistics: the vehicle routing problem with simultaneous delivery and pick-up”. In: *OR-Spektrum* 23.1 (2001), pp. 79–96.

[61] Fermín Alfredo Tang Montané and Roberto Dìéguez Galvão. “Vehicle routing problems with simultaneous pick-up and delivery service”. In: *Opsearch* 39.1 (2002), pp. 19–33.

[62] Jacques F Benders. “Partitioning procedures for solving mixed-variables programming problems”. In: *Numerische mathematik* 4.1 (1962), pp. 238–252.

[63] Güneş Erdogan, Maria Battarra, Gilbert Laporte, and Daniele Vigo. “Metaheuristics for the traveling salesman problem with pickups, deliveries and handling costs”. In: *Computers & Operations Research* 39.5 (2012), pp. 1074–1086.

[64] Fred Glover and Manuel Laguna. “Tabu search”. In: *Handbook of combinatorial optimization*. Springer, 1998, pp. 2093–2229.

[65] Richard P Hornstra, Allyson Silva, Kees Jan Roodbergen, and Leandro C Coelho. “The vehicle routing problem with simultaneous pickup and delivery and handling costs”. In: *Computers & Operations Research* 115 (2020), p. 104858.

[66] Stefan Ropke and David Pisinger. “An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows”. In: *Transportation science* 40.4 (2006), pp. 455–472.

[67] Carlos Rey, Paolo Toth, and Daniele Vigo. “An Iterated Local Search for the Traveling Salesman Problem with Pickup, Delivery and Handling Costs”. In: *2020 39th International Conference of the Chilean Computer Science Society (SCCC)*. IEEE. 2020, pp. 1–8.

[68] Kengo Katayama and Hiroyuki Narihisa. “Iterated local search approach using genetic transformation to the traveling salesman problem”. In: *Proceedings of the 1st Annual Conference on Genetic and Evolutionary Computation-Volume 1*. Morgan Kaufmann Publishers Inc. 1999, pp. 321–328.

[69] Anand Subramanian and Maria Battarra. “An iterated local search algorithm for the travelling salesman problem with pickups and deliveries”. In: *Journal of the Operational Research Society* 64.3 (2013), pp. 402–409.

[70] Valentina Cacchiani, Carlos Contreras-Bolton, John W Escobar, Luis M Escobar-Falcon, Rodrigo Linfati, and Paolo Toth. “An Iterated Local Search Algorithm for the Pollution Traveling Salesman Problem”. In: *New Trends in Emerging Complex Real Life Problems*. Springer, 2018, pp. 83–91.

[71] William Cook. *Concorde TSP solver*. http://www.math.uwaterloo.ca/tsp/concorde.html. Access 05-07-2020. 2015.

[72] Georges A Croes. “A method for solving traveling-salesman problems”. In: *Operations research* 6.6 (1958), pp. 791–812.

[73] Scott Kirkpatrick, C Daniel Gelatt, and Mario P Vecchi. “Optimization by simulated annealing”. In: *science* 220.4598 (1983), pp. 671–680.

[74] Yaghout Nourani and Bjarne Andresen. “A comparison of simulated annealing cooling strategies”. In: *Journal of Physics A: Mathematical and General* 31.41 (1998), p. 8373.

[75] David Abramson, Mohan Krishnamoorthy, and Henry Dang. “Simulated annealing cooling schedules for the school timetabling problem”. In: *Asia Pacific Journal of Operational Research* 16 (1999), pp. 1–22.

[76] Keld Helsgaun. “An extension of the Lin-Kernighan-Helsgaun TSP solver for constrained traveling salesman and vehicle routing problems”. In: *Roskilde: Roskilde University* (2017).

[77] Michel Gendreau, Gilbert Laporte, and Daniele Vigo. “Heuristics for the traveling salesman problem with pickup and delivery”. In: *Computers & Operations Research* 26.7 (1999), pp. 699–714.

[78] Manuel López-Ibáñez, Jérémie Dubois-Lacoste, Leslie Pérez Cáceres, Mauro Birattari, and Thomas Stützle. "The irace package: Iterated racing for automatic algorithm configuration". In: *Operations Research Perspectives* 3 (2016), pp. 43–58.

[79] S. Martello and P. Toth. *Knapsack Problems: Algorithms and Computer Implementations*. New York, NY, USA: John Wiley & Sons, Inc., 1990.

[80] Paul E. Black. "Dictionary of algorithms and data structures". In: (2005).

[81] C. Witzgall. "Mathematical methods of site selection for Electronic Message Systems (EMS)". In: *NASA STI/Recon Technical Report N* 76 (1975).

[82] G. Gallo, P.L. Hammer, and B. Simeone. "Quadratic knapsack problems". In: *Mathematical Programming* (1980), pp. 132–149.

[83] A. Billionnet and E. Soutif. "An exact method based on Lagrangian decomposition for the 0-1 quadratic knapsack problem". In: *European Journal of Operational Research* 157.3 (2004), pp. 565–575.

[84] B.A. Julstrom. "Greedy, genetic, and greedy genetic algorithms for the quadratic knapsack problem". In: *Proceedings of the 7th Annual Conference on Genetic and Evolutionary Computation*. 2005, pp. 607–614.

[85] D. Pisinger. "The quadratic knapsack problem - a survey". In: *Discrete Applied Mathematics* 155.5 (2007), pp. 623–648.

[86] W.D. Pisinger, A.B. Rasmussen, and R. Sandvik. "Solution of large quadratic knapsack problems through aggressive reduction". In: *INFORMS Journal on Computing* 19.2 (2007), pp. 280–290.

[87] S. Pulikanti and A. Singh. "An artificial bee colony algorithm for the quadratic knapsack problem". In: *International Conference on Neural Information Processing*. Springer. 2009, pp. 196–205.

[88] E. Lalla-Ruiz, E. Segredo, and S. Voß. "A cooperative learning approach for the quadratic Knapsack problem". In: *International Conference on Learning and Intelligent Optimization*. 2018, pp. 31–35.

[89] A. Hiley and B.A. Julstrom. "The quadratic multiple knapsack problem and three heuristic approaches to it". In: *Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation*. 2006, pp. 547–552.

[90] A. Singh and A.S. Baghel. "A new grouping genetic algorithm for the quadratic multiple knapsack problem". In: *European Conference on Evolutionary Computation in Combinatorial Optimization*. Springer. 2007, pp. 210–218.

[91] S. Sundar and A. Singh. "A swarm intelligence approach to the quadratic multiple knapsack problem". In: *International Conference on Neural Information Processing*. Springer. 2010, pp. 626–633.

[92] SM. Soak and SW. Lee. "A memetic algorithm for the quadratic multiple container packing problem". In: *Applied Intelligence* 36.1 (2012), pp. 119–135.

[93] C. García-Martínez, F. Glover, F.J. Rodriguez, M. Lozano, and R. Martí. "Strategic oscillation for the quadratic multiple knapsack problem". In: *Computational Optimization and Applications* 58.1 (2014), pp. 161–185.

[94] C. García-Martínez, F.J. Rodriguez, and M. Lozano. "Tabu-enhanced iterated greedy algorithm: a case study in the quadratic multiple knapsack problem". In: *European Journal of Operational Research* 232.3 (2014), pp. 454–463.

[95] Y. Chen and J.K. Hao. "Iterated responsive threshold search for the quadratic multiple knapsack problem". In: *Annals of Operations Research* 226.1 (2015), pp. 101–131.

[96] Y. Chen, J.K. Hao, and F. Glover. "An evolutionary path relinking approach for the quadratic multiple knapsack problem". In: *Knowledge-Based Systems* 92 (2016), pp. 23–34.

[97] J. Qin, X. Xu, Q. Wu, and T.C.E. Cheng. "Hybridization of tabu search with feasible and infeasible local searches for the quadratic multiple knapsack problem". In: *Computers & Operations Research* 66 (2016), pp. 199–214.

[98] T. Tlili, H. Yahyaoui, and S. Krichen. "An iterated variable neighborhood descent hyperheuristic for the quadratic multiple knapsack problem". In: *Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing 2015*. Springer, 2016, pp. 245–251.

[99] Méziane Aïder, Oussama Gacem, and Mhand Hifi. "Branch and solve strategies-based algorithm for the quadratic multiple knapsack problem". In: *Journal of the Operational Research Society* (2020), pp. 1–18.

[100] Mhand Hifi, A Mohamed Youssouf, Toufik Saadi, and L Youssef. "A Co-operative Swarm Optimization-Based Algorithm for the Quadratic Multiple Knapsack Problem". In: *2020 7th International Conference on Control, Decision and Information Technologies (CoDIT)*. Vol. 1. IEEE. 2020, pp. 1168–1173.

[101] D. Bergman. "An Exact Algorithm for the Quadratic Multiknapsack Problem with an Application to Event Seating". In: *INFORMS Journal on Computing* 31.3 (2019), pp. 477–492.

[102] R. Fortet. "L'algèbre de Boole et ses applications en recherche opérationnelle". In: *Cahiers du Centre d'Études de Recherche Opérationnelle* 1.4 (1959), pp. 5–36.

[103] F. Glover and E. Woolsey. "Converting the 0-1 polynomial programming problem to a 0-1 linear program". In: *Operations Research* 22.1 (1974), pp. 180–182.

[104] F. Glover. "Improved linear integer programming formulations of nonlinear integer problems". In: *Management Science* 22.4 (1975), pp. 455–460.

[105] F. Furini and E. Traversi. "Theoretical and computational study of several linearisation techniques for binary quadratic problems". In: *Annals of Operations Research* 279.1-2 (2019), pp. 387–411.

[106] W.P. Adams and H.D. Sherali. "A tight linearization and an algorithm for zero-one quadratic programming problems". In: *Management Science* 32.10 (1986), pp. 1274–1290.

[107] H.D. Sherali and W.P. Adams. "A hierarchy of relaxations between the continuous and convex hull representations for zero-one programming problems". In: *SIAM Journal on Discrete Mathematics* 3.3 (1990), pp. 411–430.

[108] G.L. Nemhauser and L.A. Wolsey. *Integer and combinatorial optimization*. John Wiley & Sons, 1998.

[109] A. Caprara, D. Pisinger, and P. Toth. "Exact solution of the quadratic knapsack problem". In: *INFORMS Journal on Computing* 11.2 (1999), pp. 125–137.

[110] D. Pisinger. "Upper bounds and exact algorithms for p-dispersion problems". In: *Computers & Operations Research* 33.5 (2006), pp. 1380–1398.

[111] A. Caprara. "Constrained 0–1 quadratic programming: Basic approaches and extensions". In: *European Journal of Operational Research* 187.3 (2008), pp. 1494–1503.

[112] M. Guignard. "Strong RLT1 bounds from decomposable Lagrangean relaxation for some quadratic 0–1 optimization problems with linear constraints". In: *Annals of Operations Research* (2020), pp. 173–200.

[113] S. Martello and P. Toth. "A bound and bound algorithm for the zero-one multiple knapsack problem". In: *Discrete Applied Mathematics* 3.4 (1981), pp. 275–288.

[114] A. Frangioni. "Generalized Bundle Methods". In: *SIAM Journal on Optimization* 13.1 (2002), pp. 117–156.

[115] T. Saraç and A. Sipahioglu. "Generalized quadratic multiple knapsack problem and two solution approaches". In: *Computers & Operations Research* 43.1 (2014), pp. 78–89.

[116] Mustafa Avci and Seyda Topaloglu. "An adaptive local search algorithm for vehicle routing problem with simultaneous and mixed pickups and deliveries". In: *Computers & Industrial Engineering* 83 (2015), pp. 15–29.

[117] M. Avci and S. Topaloglu. "A multi-start iterated local search algorithm for the generalized quadratic multiple knapsack problem". In: *Computers & Operations Research* 83 (2017), pp. 54–65.

[118] Martina Fischetti and Matteo Fischetti. "Matheuristics". In: *Handbook of heuristics*. Springer, 2018, pp. 121–153.

[119] Krzysztof Fleszar. "A Branch-and-Bound Algorithm for the Quadratic Multiple Knapsack Problem". In: *European Journal of Operational Research* (2021).

[120] Amina Lamghari. "Mine planning and oil field development: a survey and research potentials". In: *Mathematical Geosciences* 49.3 (2017), pp. 395–437.

[121] W Hustrulid and M Kuchta. "Open pit mine planning and design. Vol 1. Fundamentals; Vol. 2. CSMine software package and orebodey case examples. 2nd." In: (2006).

[122] Helmut Lerchs. "Optimum design of open-pit mines". In: *Trans CIM* 68 (1965), pp. 17–24.

[123] W Brian Lambert, Andrea Brickey, Alexandra M Newman, and Kelly Eurek. "Open-pit block-sequencing formulations: a tutorial". In: *Interfaces* 44.2 (2014), pp. 127–142.

[124] Renaud Chicoisne, Daniel Espinoza, Marcos Goycoolea, Eduardo Moreno, and Enrique Rubio. "A new algorithm for the open-pit mine production scheduling problem". In: *Operations research* 60.3 (2012), pp. 517–528.

[125] Shi Qiang Liu and Erhan Kozan. "New graph-based algorithms to efficiently solve large scale open pit mining optimisation problems". In: *Expert Systems with Applications* 43 (2016), pp. 59–65.

[126] S Ramazan and R Dimitrakopoulos. "Recent applications of operations research and efficient MIP formulations in open pit mining". In: *SME Transactions* 316 (2004).

[127] F Navarro. "Un algoritmo genético paralelo y distribuido para el agendamiento de bloques en minas a cielo abierto". MA thesis. www.usach.cl: Universidad de Santiago de Chile, 2015.

[128] Agoston E Eiben and James Smith. *Introduction to evolutionary computing*. Vol. 53. Springer, 2003.

[129] David E Goldberg and John Henry Holland. "Genetic algorithms and machine learning". In: (1988).

[130] Franz Rothlauf. "Representations for genetic and evolutionary algorithms". In: *Representations for Genetic and Evolutionary Algorithms*. Springer, 2006, pp. 9–32.

[131] James MacQueen. "Some methods for classification and analysis of multivariate observations". In: *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*. Vol. 1. 14. Oakland, CA, USA. 1967, pp. 281–297.

[132] Kunihiko Fukushima and Sei Miyake. "Neocognitron: A self-organizing neural network model for a mechanism of visual pattern recognition". In: *Competition and cooperation in neural nets*. Springer, 1982, pp. 267–285.