Reformulations and Decompositions of Mixed Integer Linear and Nonlinear Programs

Author: Dr. Fabio Furini
Supervisor: Dr. Ridha Mahjoub

A thesis submitted in fulfillment of the requirements for Habilitation à diriger des Recherches of Université Paris-Dauphine

October 14, 2018
“divide et impera”

Philip II of Macedon

“divide ut regnas”

Caesar
My backbone research project is to study and develop algorithms based on Reformulations and Decompositions of Mixed Integer Linear and Nonlinear Programs. The principal idea is to exploit the hidden problem structure in order to take advantage of reformulation and decomposition principles for solving general, large, practical decision problems formulated as Mixed Integer Linear and Nonlinear Programs. My research focuses on identifying and taking advantage of innovative decomposition and reformulation methods for hard Combinatorial Optimisation Problems.

Divide and conquer, from Latin divide et impera, is one of the key techniques for tackling combinatorial optimization problems. It relies on the idea of decomposing complex problems into a sequence of subproblems that are then easier to handle. Decomposition techniques (such as Dantzig-Wolfe, Lagrangian, or Benders decomposition) are extremely effective in a wide range of applications, including cutting & packing, production & scheduling, routing & logistics, telecommunications, transportation and many others. Moreover, decomposition techniques are playing an important role in many different fields of mixed-integer linear and non-linear optimization, multi objective optimization, optimization under uncertainty, bilevel optimization, etc. Despite the tremendous amount of research on these topics, the mathematical optimization community is constantly faced with new challenges coming from theoretical aspects and real world applications that require the development of new advanced tools.

In the first section of this document my principal areas and topics of research are presented. The second section focuses on reformulation of Coloring Problems on Graphs. Several generalizations of the classical Vertex Coloring Problem will be addressed. The third section focuses on reformulations of Knapsack Problems, in this context several effective decompositions techniques will be presented and tailored for challenging variants of the classical knapsack problem. Finally the fourth chapter focuses on an effective reformulation technique for a class of non linear programming problems with semi-continuous variables.
Acknowledgements

First of all I want to thank my family, they have always supported me! They allowed me to find my way, backing me in all my personal and professional projects, without them I would not be who I am. They gave me my principles and they are my roots.

I want to thank my beloved Cecilia for her love and support. She has been beside me during these last years, giving me the necessary strength to achieve my goals. Without her I would not be where I am.

My thanks go also to my PhD supervisors, Alberto and Paolo. They made me discover the world of academic research and they transmitted to me the love for this domain, without them I would not do what I am doing.

A special mention goes to my group of research at Paris-Dauphine, in particular to professor Ridha Mahjoub. It is thank to them that I had my first academic position, they put their trust in me and they supported me. It is also thank to them that I managed to develop my streams of research over the last years.

It is a great pleasure to acknowledge the reviewers of my HDR, i.e., three outstanding researches of the field who agreed to judge my scientific maturity. I want to personally thank professor Martine Labbé, professor Nelson Maculan and professor Eduardo Uchoa for acting as reviewers of this documents. And I want to thank professor Roberto Wolfler Calvo, professor Mauro Dell’Amico and professor Vangelis Paschos for having accepted to be part of the jury of this HDR.

Last but not least, I want to personally thank Ivana and Enrico, two of my main co-authors and friends. We wrote together several papers and I had really good time in doing research with them!
## Contents

Abstract

Acknowledgements

1 Introduction and areas of scientific interest
  1.1 Automatic Dantzig-Wolfe Decomposition ........................................ 1
  1.2 Two-Dimensional Cutting Problems ................................................. 2
  1.3 Generalizations of the Vertex Coloring Problem ............................... 3
  1.4 Generalizations of the Knapsack Problem ......................................... 5
  1.5 Non-Linear Programming Problems .................................................. 6
  1.6 Real-world Applications ................................................................. 8
  1.7 Teaching Statement ........................................................................... 10

2 Reformulation of Coloring Problems on Graphs ..................................... 13
  2.1 The Vertex Coloring and Max-Coloring problems ................................. 14
      2.1.1 Extended Formulations ............................................................. 20
      2.1.2 Computational Experiments ...................................................... 22
  2.2 The Partition Coloring Problem ......................................................... 25
      2.2.1 Extended Formulation ................................................................ 27
      2.2.2 Computational Experiments ...................................................... 32

3 Reformulation of Knapsack Problems .................................................... 35
  3.1 The Knapsack Problem with Setup ...................................................... 35
      3.1.1 Computational Experiments ...................................................... 43
  3.2 The interval min-max regret knapsack problem ..................................... 45
      3.2.1 A Lagrangian-based branch-and-cut algorithm ............................... 49
      3.2.2 Computational Experiments ...................................................... 50

4 Reformulation of a class of convex separable MINLP .............................. 53
  4.1 Improving AP2R using dual information .............................................. 57
  4.2 Computational Experiments ............................................................... 58

5 Conclusion ......................................................................................... 61

Bibliography ......................................................................................... 63
To my beloved family
Chapter 1

Introduction and areas of scientific interest

I have received my Bachelors and Masters degrees in Engineering and Industrial Management, and my Ph.D in Automation and Operations Research at the University of Bologna in 2004, 2007 and 2011 respectively. I have conducted periods of research at prestigious universities in Europe and the United States, including the University of Colorado, the Imperial College of London, the University of Vienna and many others. I am currently holding a position of maître de conférences at Paris-Dauphine University, since September 2013. My research is a combination of the disciplines of Mathematics, Economics, Information Technology, and Operations Research. I have used advanced analytical techniques to arrive at solutions of optimal or near-optimal standard to intricate decision-making problems. The focus of my research has been on achieving operational efficiencies, aiming to develop general software, generalizable insights and applications. I have published articles on these subjects in many leading international journals in the field of optimization such as Mathematical Programming, INFORMS Journal on Computing, European Journal of Operational Research, Computational Optimization and Applications, Operations Research Letters, Networks, Transportation Research Part B, Computer & Operations Research, Information Processing Letters, Discrete Optimization, Journal of Scheduling, Omega and Discrete Applied Mathematics. I have also published several articles in proceedings of international conferences, such as IPCO, ISCO, INOC, CPAIOR and many others.

My principal areas of research concern theory and algorithms for Mixed Integer Linear and Non-Linear Programming problems and the study of challenging applications. In this introduction, I will describe my principal achievements in the field grouped by areas of scientific interest and I will conclude with a teaching statement.

1.1 Automatic Dantzig-Wolfe Decomposition

From the beginning of my Ph.D, I studied the Dantzig-Wolfe decomposition and reformulation. This technique is well-known to provide strong dual bounds for specially structured mixed integer programs (MIPs). However, the method was not implemented in any state-of-the-art MIP solver as it was considered to require structural problem knowledge and tailoring to this structure. As part of a collaboration between different universities, including Bologna and Aachen Universities, I worked on the development of a computational proof-of-concept that the process can be automated. Our results suggested that generic Dantzig-Wolfe reformulation can be effective used to solve generic MIPs outperforming the classical approaches based on Branch-and-Cut algorithms for different classes of problems. Our results
are now implemented in SCIP Optimization Suite which is a state-of-the-art non commercial solver. In particular, now the generic Dantzig-Wolfe decomposition can be automatically exploited using GCG, a generic branch-cut-and-price solver for mixed integer programs. The main results concerning these line of research can be found in my PhD these and in these two manuscripts: Bergner et al., 2015a, Bergner et al., 2011a.

1.2 Two-Dimensional Cutting Problems

Two dimensional cutting problems are about obtaining a set of rectangular items from a set of rectangular stock pieces and are of great relevance in industry, whenever a sheet of wood, metal or other material has to be cut.

Depending on the industry, special features for the cuts may be required; a very common one which applies to glass and wood cutting is to have guillotine cuts.

(i) In guillotine cutting, items are obtained from panels through cuts that are parallel to the sides of the panel and cross the panel from one side to the other;

(ii) Cuts are performed in stages, where each stage consists of a set of parallel guillotine cuts on the shapes obtained in the previous stages;

(iii) Each cut removes a so-called strip from a panel. If during the cut sequence, the width of each cut strip equals the width of the widest item obtained from the strip, then the cut is denoted as restricted.

In Figure 1.1 we report, on the left, a pattern (cutting scheme) that cannot be obtained through guillotine cuts, in the center, a pattern that can be obtained through three-stage restricted guillotine cuts, and in the right a pattern that needs unrestricted guillotine cuts to be obtained.

In Figure 1.1 we report, on the left, a pattern (cutting scheme) that cannot be obtained through guillotine cuts, in the center, a pattern that can be obtained through three-stage restricted guillotine cuts, and in the right a pattern that needs unrestricted guillotine cuts to be obtained.

I have developed several lines of research in this domain and these are the main results:

• In Furini et al., 2012, we considered the Two-Dimensional Cutting Stock Problem where stock of different sizes is available (2DCSP), and a set of rectangular items has to be obtained through two-staged guillotine cuts. We proposed a heuristic algorithm, based on column generation, which requires as subproblem the solution of a Two-Dimensional Knapsack Problem with two-staged guillotines cuts. A further contribution consisted in the definition of a Mixed Integer Linear Programming Model for the solution of this Knapsack Problem,
1.3 Generalizations of the Vertex Coloring Problem

Graph coloring problems are among the most studied ones in both graph theory and combinatorial optimization. Given an undirected graph \( G = (V, E) \) with \( |V| = n \) vertices and \( |E| = m \) edges, the classical Vertex Coloring Problem (VCP) consists of assigning a color to each vertex of the graph in such a way that two adjacent vertices do not share the same color and the total number of colors is minimized. The chromatic number of \( G \), denoted by \( \chi(G) \), is the minimum number of colors in a coloring of \( G \).
I have developed several lines of research in this domain and these are the main results:

- In Furini and Malaguti, 2012a, we considered the Weighted Vertex Coloring Problem (WVCP) (also known in the literature as the Max Coloring Problem), in which a positive weight is associated to each vertex of a graph. In WVCP, one is required to assign a color to each vertex in such a way that colors on adjacent vertices are different, and the objective is to minimize the sum of the costs of the colors used, where the cost of each color is given by the maximum weight of the vertices assigned to that color. This NP-hard problem arises in practical scheduling applications, where it is also known as Scheduling on a Batch Machine with Job Compatibilities. We proposed the first exact algorithm for the problem, which is based on column generation and branch-and-price. Computational results on a large set of instances from the literature were reported, showing excellent performance compared with the best heuristic algorithms from the literature (see Section 2 for further details on this problem).

- In Cornaz, Furini, and Malaguti, 2017, we tackled four different coloring problems formulated as Maximum Weight Stable Set Problems on an associated graph. We exploited the transformation proposed by Cornaz and Jost Cornaz and Jost, 2008, where given a graph $G$, an auxiliary graph $\hat{G}$ is constructed, such that the family of all stable sets of $\hat{G}$ is in one-to-one correspondence with the family of all feasible colorings of $G$. The transformation in Cornaz and Jost, 2008 was originally proposed for the classical Vertex Coloring and the Max-Coloring problems; we extended it to the Equitable Coloring Problem and the Bin Packing Problem with Conflicts. We discussed the relation between the Maximum Weight Stable formulation and a polynomial-size formulation for the VCP, proposed in the literature and called the Representative formulation. We reported extensive computational experiments on benchmark instances of the four problems, and compared the solution method with the state-of-the-art algorithms. By exploiting the proposed method, we managed to largely outperform the state-of-the-art algorithm for the Max-coloring Problem, and we are able to solve, for the first time to proven optimality, 14 Max-coloring and 2 Equitable Coloring instances (see Section 2 for further details on this problem).

- In Furini, Gabrel, and Ternier, 2017 and Furini, Gabrel, and Ternier, 2015, we improved the DSATUR-based Branch-and-Bound (DSATUR-B&B) which is an effective exact algorithm for the VCP. One of the main drawbacks of this exact algorithm is that a lower bound is computed only once at the root node of the branching scheme and it is never updated. We introduced a reduced graph which allows the computation of lower bounds at each node of the search tree. We compared the effectiveness of different classical VCP bounds, plus a new lower bound based on the 1-to-1 mapping between VCPs and Stable Set Problems. Our new DSATUR-B&B outperformed the state-of-the-art DSATUR-B&B for random VCP instances with high density, significantly increasing the size of solvable instances. Similar results can be achieved for a subset of high density DIMACS instances.

- In Furini, Malaguti, and Santini, 2017, we studied the Partition Coloring Problem (PCP), a generalization of the Vertex Coloring Problem where the vertex set is partitioned. The PCP asks to select one vertex for each subset of the partition in such a way that the chromatic number of the induced graph is minimum. We proposed a new Integer Linear Programming formulation with an
1.4 Generalizations of the Knapsack Problem

The classical Knapsack Problem (KP) is one of the most famous problems in combinatorial optimization. Given a knapsack capacity $C$ and a set $N = \{1, \ldots, n\}$ of items, the $j$-th having a profit $p_j$ and a weight $w_j$, KP asks for a maximum profit subset of items whose total weight does not exceed the capacity. KP can be formulated using the following Integer Linear Program (ILP):

$$\max \left\{ \sum_{j \in N} p_j x_j \ : \ \sum_{j \in N} w_j x_j \leq C, x_j \in \{0, 1\}, j \in N \right\}$$  \hfill (1.1)$$

where each binary variable $x_j$ takes value 1 if and only if item $j$ is inserted in the knapsack.

We studied several variants and generalizations of the classical knapsack problem and these are the main results we managed to achieve in this area:

- In Caprara, Furini, and Malaguti, 2013, we studied a natural generalization of the knapsack problem called Temporal Knapsack Problem (TKP), in which each item exists only for a given time interval. In this setting, one has to select a subset of the items, guaranteeing that for each time instant the set of existing selected items has total weight not larger than the knapsack capacity. We focused on the exact solution of the problem, noting that prior to our work the best method was the straightforward application of a general-purpose solver to the natural ILP formulation. Our results indicated that much better results can be obtained by using the same general-purpose solver to tackle a nonstandard Dantzig-Wolfe reformulation in which subproblems are associated with groups of constraints. In Caprara et al., 2016, we proposed to solve the TKP using what we called a Recursive Dantzig-Wolfe Reformulation (DWR) method. The generic idea of Recursive DWR is to solve a Mixed Integer Program (MIP) by recursively applying DWR, i.e., by using DWR not only for solving the original MIP but also for recursively solving the pricing sub-problems. In a binary case (like the TKP), the Recursive DWR method can be performed in such a way that the only two components needed during the optimization are a Linear Programming solver and an algorithm for solving Knapsack Problems. The Recursive DWR allowed us to solve TKP instances through computation of strong dual bounds, which could not be obtained by the previous exact methods based on DWR.

- In Furini et al., 2015a, we considered a generalization of the 0-1 knapsack problem in which the profit of each item can take any value in a range characterized
by a minimum and a maximum possible profit. A set of specific profits is called a scenario. Each feasible solution associated with a scenario has a regret, given by the difference between the optimal solution value for such scenario and the value of the considered solution. The interval min-max regret knapsack problem (MRKP) is then to find a feasible solution such that the maximum regret over all scenarios is minimized. The problem is extremely challenging both from a theoretical and a practical point of view. Its decision version is complete for the complexity class $\Sigma^P_2$ hence it is most probably not in $NP$. In addition, even computing the regret of a solution with respect to a scenario requires the solution of an $NP$-hard problem. We examined the behavior of classical combinatorial optimization approaches when adapted to the solution of the MRKP. We introduced an iterated local search approach and a Lagrangian-based branch-and-cut algorithm, and evaluated their performance through extensive computational experiments (see Section 3 for further details on this problem).

• In Furini, Ljubić, and Sinnl, 2017 and Furini, Ljubić, and Sinnl, 2015, we considered an other challenging variant of the knapsack problem. Given a set of items with profits and weights and a knapsack capacity, we studied the problem of finding a maximal knapsack packing that minimizes the profit of selected items. We proposed an effective dynamic programming (DP) algorithm which has pseudo-polynomial time complexity. We demonstrated the equivalence between this problem and the problem of finding a minimal knapsack cover that maximizes the profit of selected items. In an extensive computational study on a large and diverse set of benchmark instances, we demonstrated that the new DP algorithm outperformed a state-of-the-art commercial mixed-integer programming (MIP) solver applied to the two best performing MIP models proposed in Furini, Ljubić, and Sinnl, 2015.

• Finally in Furini, Monaci, and Traversi, 2018, we consider a generalization of the knapsack problem in which items are partitioned into classes, each characterized by a fixed cost and capacity. We studied three alternative Integer Linear Programming formulations. For each formulation, we designed an efficient algorithm to compute the linear programming relaxation (one of which is based on Column Generation techniques). We theoretically compared the strength of the relaxations and derived specific results for a relevant case arising in benchmark instances from the literature. Finally, we embedded the algorithms above into a unified implicit enumeration scheme which was run in parallel with an improved Dynamic Programming algorithm to effectively solve the problem to proven optimality. An extensive computational analysis showed that our new exact algorithm is capable of efficiently solving all the instances of the literature and turned out to be the best algorithm for instances with a low number of classes (see Section 3 for further details on this problem).

1.5 Non-Linear Programming Problems

My research in the field of Mixed-Integer Non-Linear Optimization has been focused on several important aspects; these are the main results:

• In Frangioni, Furini, and Gentile, 2015, we proposed the Approximated Perspective Relaxations. The Perspective Reformulation (PR) of a Mixed-Integer NonLinear Program with semi-continuous variables is obtained by replacing
each term in the (separable) objective function with its convex envelope. Solving the corresponding continuous relaxation requires appropriate techniques. Under some rather restrictive assumptions, the Projected PR (P^2R) can be defined where the integer variables are eliminated by projecting the solution set onto the space of the continuous variables only. This approach produces a simple piecewise-convex problem with the same structure as the original one; however, this prevents the use of general-purpose solvers, in that some variables are then only implicitly represented in the formulation. In Frangioni, Furini, and Gentile, 2015, we showed how to construct an Approximated Projected PR (AP^2R) whereby the projected formulation is “lifted” back to the original variable space, with each integer variable expressing one piece of the obtained piecewise-convex function. In some cases, this produces a reformulation of the original problem with exactly the same size and structure as the standard continuous relaxation, but providing substantially improved bounds. In the process we also substantially extended the approach beyond the original P^2R development by relaxing the requirement that the objective function be quadratic and the left endpoint of the domain of the variables be non-negative. While the AP^2R bound can be weaker than that of the PR, this approach can be applied in many more cases and allows direct use of off-the-shelf MINLP software; this was shown to be competitive with previously proposed approaches in some applications. In Frangioni, Furini, and Gentile, 2017, we proposed an improvement of the Approximated Projected Perspective Reformulation (AP^2R) for dealing with constraints linking the binary variables. The new approach solves the Perspective Reformulation (PR) once, and then use the corresponding dual information to reformulate the problem prior to applying AP^2R, thereby combining the root bound quality of the PR with the reduced relaxation computing time of AP^2R. Computational results for the cardinality-constrained Mean-Variance portfolio optimization problem showed that the new approach outperformed the previous state-of-the-art approach proposed in Frangioni, Furini, and Gentile, 2015 for a class of problems with semi-continuous variables (see Section 4 for further details on this topic).

- In Furini and Traversi, 2013, we proposed an Hybrid SDP Bounding Procedure. The principal idea of this work was to exploit Semidefinite Programming (SDP) relaxation within the framework provided by Mixed Integer Nonlinear Programming (MINLP) solvers when tackling Binary Quadratic Problems. We included the SDP relaxation in a state-of-the-art MINLP solver as an additional bounding technique and demonstrated that this idea could be computationally useful. The Quadratic Stable Set Problem is adopted as the case study. The tests indicate that the Hybrid SDP Bounding Procedure allows an average 50% cut of the overall computing time and a cut of more than one order of magnitude for the branching nodes.

- In Furini and Traversi, 2017, we performed a theoretical and computational study of the classical Linearisation Techniques (LT) and we propose a new LT for Binary Quadratic Problems (BQPs). We discussed the relations between the Linear Programming (LP) relaxations of the considered LT for generic BQPs. We proved that for a specific class of BQP all the LTs have the same LP relaxation value. We also compared the LT computational performance for four different BQP’s from the literature. We considered the Unconstrained BQP and the Maximum Cut of edge-weighted graphs and, in order to measure the effects of constraints, we also considered the quadratic extension of two classical
combinatorial optimization problems, i.e., the Knapsack and Stable Set problems.

1.6 Real-world Applications

In my research, I have also devoted attention to the real-world applications of Operations Research, and I have had the chance to investigate the problematics associated with Aircraft Routing and Sequencing, Train Timetabling, Production Planning and Liner Shipping. All problems have been studied in collaboration with some of the leading industries of the sector.

- In a scenario characterized by a continuous growth of air transportation demand, the runways of large airports serve hundreds of aircraft every day. Aircraft sequencing is a challenging problem that aims to increase runway capacity in order to reduce delays as well as the workload of air traffic controllers. In many cases, the air traffic controllers solve the problem by using the simple “First-Come-First-Serve” (FCFS) rule. In Furini et al., 2015b and Furini, Persiani, and Toth, 2012, we presented a rolling horizon approach which partitions a sequence of aircraft into chunks and solves the Aircraft Sequencing Problem (ASP) individually for each of these chunks. Some rules for deciding how to partition a given aircraft sequence have been proposed and their effects on solution quality investigated. Moreover, two Mixed Integer Linear Programming (MILP) models for the ASP have been reviewed in order to formalize the problem, and a tabu search heuristic was proposed for finding solutions to the ASP in a short computation time. Finally, we developed an IRHA which, by using different chunking rules, was able to find solutions significantly improving on the FCFS rule for real world air traffic instances from Milano Linate Airport. In Furini et al., 2014, we presented state space reduction techniques for a dynamic programming algorithm applied to the Aircraft Sequencing Problem (ASP) with Constrained Position Shifting (CPS). We considered the classical version of the ASP, which calls for determining the order in which a given set of aircraft should be assigned to a runway at an airport, subject to minimum separations in time between consecutive aircraft, in order to minimize the sum of the weighted deviations from the scheduled arrival/departure times of the aircraft. The focus of the work was on a number of ways of improving the computation times of the dynamic programming algorithm proposed. This was achieved by using heuristic upper bounds and a completion lower bound in order to reduce the state space in the dynamic programming algorithm. We compared our algorithm to an approach based on mixed integer linear programming, which was adapted from the literature for the case of CPS. We showed using real-world air traffic instances from the Milan Linate Airport that the dynamic programming algorithm significantly outperformed the MILP. Furthermore, we showed that the proposed algorithm was capable of solving very large instances in short computation times, and that it is suitable for use in a real-time setting.

- The integration of drones into civil airspace is one of the most challenging problems for the automation of the controlled airspace, and the optimization of the drone route is a key step for this process. In Furini, Persiani, and Toth, 2016, we optimized the route planning of a drone mission that consists of departing from an airport, flying over a set of mission way points and coming
1.6. Real-world Applications

back to the initial airport. We assume that during the mission a set of piloted aircraft flies in the same airspace and thus the cost of the drone route depends on the air traffic and on the avoidance maneuvers used to prevent possible conflicts. Two Air Traffic Management techniques, i.e., routing and holding, are modeled in order to maintain a minimum separation between the drone and the piloted aircraft. The considered problem, called the Time Dependent Traveling Salesman Planning Problem in Controlled Airspace (TDTSPPCA), relates to the drone route planning phase and aims to minimize the total operational cost. Two heuristic algorithms have been proposed for the solution of the problem. A mathematical formulation based on a particular version of the Time Dependent Traveling Salesman Problem, which allows holdings at mission way points, and a Branch and Cut algorithm have been proposed for solving the TDTSPPCA to optimality. An additional formulation, based on a Travelling Salesman Problem variant that uses specific penalties to model the holding times, has been proposed and a Cutting Plane algorithm was designed. Finally, computational experiments on real-world air traffic data from Milano Linate Terminal Maneuvering Area were reported to evaluate the performance of the proposed formulations and of the heuristic algorithms.

- In Cacchiani, Furini, and Kidd, 2016 and Furini and Kidd, 2013, we considered the Train Timetabling Problem (TTP) in a railway node (i.e. a set of stations in an urban area interconnected by tracks), which calls for determining the best schedule for a given set of trains during a given time horizon, while satisfying several track operational constraints. In particular, we considered the context of a highly congested railway node in which different Train Operators wish to run trains according to timetables that they propose, called ideal timetables. The ideal timetables altogether may be (and usually are) conflicting, i.e. they do not respect one or more of the track operational constraints. The goal was to determine conflict-free timetables that differ as little as possible from the ideal ones. The problem was studied for a research project funded by Rete Ferroviaria Italiana (RFI), the main Italian railway Infrastructure Manager, who also provided us with real-world instances. We presented an Integer Linear Programming (ILP) model for the problem, which adapts previous ILP models from the literature to deal with the case of a railway node. The Linear Programming (LP) relaxation of the model was used to derive a dual bound. In addition, we proposed an iterative heuristic algorithm that is able to obtain good solutions to real-world instances with up to 1500 trains in short computing times. The proposed algorithm was also used to evaluate the capacity saturation of the railway nodes.

- In Focacci et al., 2016, a rich lot-sizing problem was studied which comes from a real-world application. Our new lot-sizing problem combines several features, i.e., parallel machines, production time windows, backlogging, lost sale and setup carryover. Three mixed integer programming formulations have been proposed. We theoretically and computationally compared these different formulations, testing them on real-world and randomly generated instances. Our study was the first step for efficiently tackling and solving this challenging real-world lot-sizing problem.

- In Alfandari et al., 2017, the problem of the optimal planning of a liner service for a barge container shipping company has been addressed. Given estimated weekly demands between pairs of ports, our goal was to determine
the subset of ports to be called and the amount of containers to be shipped between each pair of ports, so as to maximize the profit of the shipping company. In order to save possible leasing or storage costs of empty containers at the respective ports, our approach took into account the repositioning of empty containers. The line has to follow the outbound-inbound principle, starting from the port at the river mouth. We proposed a novel integrated approach in which the shipping company can simultaneously optimize the route (along with repositioning of empty containers), the choice of the final port, length of the turnaround time and the size of its fleet. To solve this problem, a new mixed integer programming model was proposed. On the publicly available set of benchmark instances for barge container routing, we demonstrated that this model provides very tight dual bounds and significantly outperformed the existing approaches from the literature for splittable demands. We also showed how to further improve this model by projecting out arc variables for modeling the shipping of empty containers. Our numerical study indicated that the latter model improves the computing times for the challenging case of unsplittable demands. We also studied the impact of the turnaround time optimization on the total profit of the company.

1.7 Teaching Statement

I think that transmitting the knowledge to the students is an important challenge for all professors. I have been involved in teaching since the end of my doctoral studies. I have been teaching courses related to Operations Research, Linear and Integer Programming and Combinatorial optimization in many different universities in Europe. In particular I started teaching together with the research group at Bologna University. Since then I moved to Paris Dauphine University where I have held the position of maître de conférences since 2013. During these years I taught different courses at all levels, starting from bachelor’s and master’s courses up to courses for PhD students. In details, as the head of the Master in Informatics for Finance, I put together the course on Optimization in Finance. This area is one of the most important subjects for Paris Dauphine University master’s students. My course touches important and challenging aspects of Optimization in finance, such as Risk management and Option Pricing. In all the courses I teach, special attention is devoted to the development of the theory and its applications. I strongly believe that the student not only has to understand the theory but also needs to be able to develop concrete applications. A common line of all my courses concerns real world examples and case studies.

I was the co-supervisor of two PhD students, the first working on themes linked to graph coloring and the second one working on real-world applications of production planning. It was a pleasure working with them and our collaborations brought to several publications in different leading journals of the domain.

This is the structure of the following section of this document. Section 2 focuses on reformulation of Coloring Problems on Graphs. Several generalizations of the classical Vertex Coloring Problem will be addressed. Section 3 focuses on reformulations of Knapsack Problems, in this context several effective decompositions techniques will be presented and tailored for challenging variants of the classical
knapsack problem. Finally Section 4 focuses on an effective reformulation technique for a class of non linear programming problems with semi-continuous variables.
Chapter 2

Reformulation of Coloring Problems on Graphs

Graph coloring problems are among the most studied ones in both graph theory and combinatorial optimization. Given an undirected graph $G = (V, E)$ with $|V| = n$ vertices and $|E| = m$ edges, the classical Vertex Coloring Problem (VCP) consists of assigning a color to each vertex of the graph in such a way that two adjacent vertices do not share the same color and the total number of colors is minimized. The chromatic number of $G$, denoted by $\chi(G)$, is the minimum number of colors in a coloring of $G$.

The VCP is an $NP$-hard problem and it has a variety of applications, among which: scheduling, register allocation, seating plan design, timetabling, frequency assignment, sport league design, and many others (we refer the interested readers to Pardalos, Mavridou, and Xue (1998), Marx (2004), and Lewis (2015)). The VCP and its variants are very challenging from a computational viewpoint; state-of-the-art exact methods can fail in optimally solving instances with more than 200 vertices; the best performing exact algorithms are usually based on exponential-size Set Covering formulations, and require Branch-and-Price techniques to be solved (see, e.g., Malaguti, Monaci, and Toth (2011b), Gualandi and Malucelli (2012b), Held, Cook, and Sewell (2012b), and Furini and Malaguti (2012c)). For dense graphs, good results are obtained by advanced Integer Linear Programming (ILP) compact formulations, like the so-called representatives formulation (see Campêlo, Campos, and Corrêa (2008b) and Cornaz, Furini, and Malaguti (2016)), which are able to remove the symmetry affecting classical descriptive compact ILP models. For random graphs, competitive experimental results are obtained by implicit enumeration schemes based on the DSATUR algorithm by Brélaz Brélaz, 1979, see, e.g., San Segundo, 2012. We also mention the Branch-and-Cut approach by Méndez-Díaz and Zabala Méndez-Díaz and Zabala, 2006, which is effective for some special classes of graphs.

Generalizations of the classical VCP:

- The Max-coloring Problem (MCP) is a generalization of the VCP where each vertex has a positive weight, and the cost of a color is given by the maximum weight of the vertices in the corresponding color class. The problem is also known as Weighted VCP and it has been addressed in Malaguti, Monaci, and Toth, 2009. The best performing exact method for MCP is the Branch-and-Price algorithm by Furini and Malaguti Furini and Malaguti, 2012b, which can solve instances with up-to 100 vertices.

- The Partition-coloring Problem (PCP) is generalization of the VCP where the vertex set is partitioned. The PCP asks to select one vertex for each subset of the
partition in such a way that the chromatic number of the induced graph is minimum. Few are the exact approaches for the PCP and only two works study ILP formulations: Frota et al. (2010) and Hoshino, Frota, and De Souza (2011). The first one proposes a branch-and-cut algorithm based on the asymmetric representatives formulation introduced by Campêlo, Campos, and Corrêa (2008b) and Campêlo, Corrêa, and Frota (2004) for the VCP. The second paper proposes instead a branch-and-price algorithm based on the Dantzig-Wolfe reformulation of the representatives formulation.

- All these VCP generalizations are \( \mathcal{NP} \)-hard problems.

**Preliminaries.** Let \( G \) be a graph with vertex set \( V(G) \) and edge set \( E(G) \). A stable set \( S \subseteq V(G) \) is a subset of vertices containing no edge, a clique \( K \subseteq V(G) \) is a subset of vertices inducing a complete subgraph. Stable sets of \( G \) are cliques of the complementary graph \( \overline{G} \) of \( G \).

Given a set of weights \( w \in \mathbb{Z}^{V(G)} \), the Max Weight Stable Set Problem (MWSSP) is to determine a stable set \( S \) of \( G \) maximizing \( \sum_{v \in S} w(v) \). We denote by \( \alpha(G,w) \) the optimum of MWSSP. The MWSSP can be naturally formulated as a Integer Linear Program (ILP)

\[
\alpha(G,w) = \max \sum_{v \in V} w(v) \cdot x_v \\
x_u + x_v \leq 1 \quad u \in E \\
x_v \in \{0,1\} \quad v \in V.
\]

A \( p \)-coloring of \( G \) is a partition \( S = S_1, \ldots, S_p \) of \( V(G) \) into \( p \) stable sets \( S_i \) where each stable set represents a color. The VCP corresponds then to find a \( p \)-coloring of \( G \) with a minimum number of colors \( p \).

### 2.1 The Vertex Coloring and Max-Coloring problems

In this section we first introduce natural ILP formulation for the VCP and the MCP and then we study some reformulations.

A trivial upper bound on the number of colors used in any optimal VCP solution is given by the number \( n = |V| \) of vertices. We can then introduce a set of binary variables \( y \) with the following meaning:

\[
y_c = \begin{cases} 
1 & \text{if color } c \text{ is used} \\
0 & \text{otherwise} 
\end{cases} \quad c = 1, 2, \ldots, n;
\]

and a set of binary variables \( x \) with the following meaning:

\[
x_{vc} = \begin{cases} 
1 & \text{if vertex } v \text{ is colored with color } c \\
0 & \text{otherwise} 
\end{cases} \quad v \in V, c = 1, 2, \ldots, n.
\]
The natural VCP ILP formulation reads as follows:

\[
\chi(G) = \min \sum_{c=1}^{n} y_c \quad (2.4)
\]

\[
\sum_{c=1}^{n} x_{vc} = 1 \quad v \in V \quad (2.5)
\]

\[
x_{vc} + x_{uc} \leq y_c \quad uv \in E, \ c = 1, 2, \ldots, n \quad (2.6)
\]

\[
x_{vc} \in \{0, 1\} \quad v \in V, \ c = 1, 2, \ldots, n \quad (2.7)
\]

\[
y_c \in \{0, 1\} \quad c = 1, 2, \ldots, n \quad (2.8)
\]

where the objective function (2.4) minimizes the number of used colors, constraints (2.10) impose that each vertex is colored, and constraints (2.6) impose that adjacent vertices do not receive the same color. Finally, constraints (2.7) and (2.8) define the variables of the formulation.

Given a graph \( G \) and vertex weights \( w \in \mathbb{Z}^V (G) \), the Max-coloring Problem is the problem of determining a coloring \( S = S_1, \ldots, S_p \) of \( G \) which minimizes \( \psi(S) := \sum_{i=1}^{p} w_i \) where \( w_i = \max_{v \in S_i} w(v) \). We denote by \( \chi_{\text{max}}(G, w) \) the optimum of MCP. When \( w \) is a unit vector, \( \psi(S) = p \) and MCP reduces to the VCP.

Introducing a set of continuous variables \( z_c \ (c = 1, 2, \ldots, n) \) denoting the cost of color \( c \) in the solution; a natural ILP formulation for the MCP reads as follows:

\[
\chi_{\text{max}}(G, w) = \min \sum_{c=1}^{n} z_c \quad (2.9)
\]

\[
\sum_{c=1}^{n} x_{vc} = 1 \quad v \in V \quad (2.10)
\]

\[
x_{vc} + x_{uc} \leq 1 \quad uv \in E, \ c = 1, 2, \ldots, n \quad (2.11)
\]

\[
z_c \geq w(v) \cdot x_{vc} \quad v \in V, \ c = 1, 2, \ldots, n \quad (2.12)
\]

\[
x_{vc} \in \{0, 1\} \quad v \in V, \ c = 1, 2, \ldots, n \quad (2.13)
\]

\[
z_c \geq 0 \quad c = 1, 2, \ldots, n \quad (2.14)
\]

Objective function (2.9) minimizes the sum of the costs of the colors, which are defined by constraints (2.12). Constraints (2.10) require that every vertex is given a color, while constraints (2.11) impose that adjacent vertices cannot receive the same color. Finally, constraints (2.13) require \( x \) variables to be binary.

Descriptive natural models for coloring problems are known to produce weak linear programming relaxations and are affected by symmetry (see Malaguti and Toth (2010) and Cornaz, Furini, and Malaguti (2016)), hence, in general they can be solved to optimality only for small graphs. For these reasons other formulations have been introduced in the literature.

We discuss now how to solve the VCP and the MCP by reformulating them as MWSSPs on an associated graph. We exploit the transformation proposed by Cornaz and Jost Cornaz and Jost, 2008, where given a graph \( G \), an auxiliary graph \( \hat{G} \) is constructed, such that the family of all stable sets of \( \hat{G} \) is in one-to-one correspondence with the family of all feasible colorings of \( G \). The advantage of this approach relies on simplicity: it allows us to solve coloring problems by solving MWSSPs, which can be tackled by problem specific algorithms, or can be formulated as ILPs of polynomial size, and solved by a general purpose ILP solver.
Let us recall the Representative formulation introduced in Campêlo, Campos, and Corrêa, 2008a for the VCP. Observe that any \( p \)-coloring \( S_1, \ldots, S_p \) of any given graph \( G = (V, E) \) is also a partition into \( p \) cliques of its complementary graph \( \overline{G} = (V, \overline{E}) \). Assume that \( \overline{G} = (V, \overline{E}) \) is a orientation of \( \overline{G} \) so that every clique of \( \overline{G} \) is acyclic. For instance, \( \overline{G} \) could be an acyclic orientation but it is not strictly necessary, a necessary and sufficient condition is that \( \overline{G} \) has no 3-dicycle. (Indeed, if a complete graph has a orientation with dicycles, the minimum length of a dicycle is necessarily three).

So, thanks to the way the orientation was chosen, in every (nonempty) clique \( K \subseteq V \) of \( \overline{G} \) there is a unique vertex \( v \in K \) which is not the head of any arc in \( K \). We call \( v \) the representative of \( K \). Moreover, there is a unique star \( \overline{S} \subseteq \overline{E} \), composed of the \( |K| - 1 \) arcs having tail \( v \) and whose head is a vertex in \( K \setminus \{v\} \). So in \( \overline{G} \), any clique \( K \) has both a unique representative \( v \), and a unique star \( \overline{S} \) (which may be empty if \( K \) is a 1-clique). A representative set of \( G \) is a subset of vertices \( v_1, \ldots, v_p \) so that there is a clique partition \( K_1, \ldots, K_p \) in \( \overline{G} \), where \( v_i \) is the representative of \( K_i \).

Thus, any given \( p \)-coloring \( S_1, \ldots, S_p \) of \( G \) is also a clique partition in \( \overline{G} \), which has a unique representative set \( v_1, \ldots, v_p \), and uniquely defined stars \( \overline{S}_1, \ldots, \overline{S}_p \) so that any vertex is either a representative or a leaf of some star. This way, finding the chromatic number of \( G \) amounts to find a representative set with minimum cardinality. This last problem can be formulated in the space \( (x, y) \in \mathbb{R}^{V \times \overline{E}} \) as follows

\[
\chi(G) = \min \sum_{v \in V} y_v \tag{2.15}
\]

\[
y_v + \sum_{e \in \delta^-(v)} x_e = 1 \quad v \in V \tag{2.16}
\]

\[
x_{vu} + x_{vz} \leq y_v \quad v \in V, v_z \in \overline{E} \tag{2.17}
\]

\[
x_{vu} + x_{vz} \leq y_v \quad v \in V, v_z \in \overline{E}, u_z, zu \notin \overline{E} \tag{2.18}
\]

\[
y \in \{0, 1\} \quad v \in V \tag{2.19}
\]

\[
x_{vu} \in \{0, 1\} \quad vu \in \overline{E}, \tag{2.20}
\]

where \( \delta^-(v) \) denotes the set of arcs in \( \overline{E} \) with head \( v \). In the above formulation, \( y \) is the characteristic vector of a representative set \( v_1, \ldots, v_p \), and \( x \) is the characteristic vector of the corresponding stars \( \overline{S}_1, \ldots, \overline{S}_p \) of \( \overline{G} \). Constraints (2.16) and (2.17) ensure that each vertex is either a representative or the leaf of some star, and constraints (2.18) ensure that every star spans a stable set of \( G \) (clique of \( \overline{G} \)). Notice that we can project out the \( x \) variables using Fourier-Motzkin elimination, and we obtain

\[
\chi(G) = |V| - \max \sum_{e \in \overline{E}} x_e \tag{2.21}
\]

\[
x_{vu} + \sum_{e \in \delta^-(v)} x_e \leq 1 \quad v \in V, v_z \in \overline{E} \tag{2.22}
\]

\[
x_{vu} + x_{vz} \leq 1 \quad v, u, z \in V, vu, vz \in \overline{E}, uz, zu \notin \overline{E} \tag{2.23}
\]

\[
x_{vu} \in \{0, 1\} \quad vu \in \overline{E}. \tag{2.24}
\]

One can observe that the feasible vectors \( x \) are in actual fact the characteristic vectors of the stable sets of an auxiliary graph \( \overline{G} \) of \( \overline{G} \) described by Cornaz and Jost Cornaz.
and Jost, 2008. In order to build $\hat{G}$, consider the line-graph $L(\overline{G})$ of the complementary graph $\overline{G}$ of $G$, that is, $L(\overline{G})$ has vertex-set the edge-set of $\overline{G}$ and two vertices are linked in $L(\overline{G})$ if the two corresponding edges are adjacent in $\overline{G}$. The auxiliary graph $\hat{G}$ is obtained from $L(\overline{G})$ by removing each edge of $L(\overline{G})$ corresponding to a simplicial pair of arcs of $\overline{G}$, that is, two arcs with the same tail and with their heads linked by another arc.

In other words, the vectors $x$ satisfying (2.22)-(2.24) are exactly the vector $x$ satisfying (2.26)-(2.29) and the following ILP is a formulation for the VCP:

$$\chi(G) = |V| - \sum_{e \in E} x_e$$

$$x_{vu} + x_{vz} \leq 1 \quad v,u,z \in V : vu,vz \in \overline{E}, uz,zu \notin \overline{E}$$

$$x_e + x_f \leq 1 \quad v \in V, e \in \delta^-(v), f \in \delta^+(v)$$

$$x_e + x_f \leq 1 \quad v \in V, e, f \in \delta^-(v)$$

$$x_{vu} \in \{0,1\} \quad vu \in \overline{E}$$

(2.25) (2.26) (2.27) (2.28) (2.29)

where we denote by $\delta^+(v)$ the set of arcs in $\overline{E}$ with tail $v$. Constraints (2.26),(2.27),(2.28) and (2.29) define $\text{STAB}(\overline{G})$, where $\overline{G}$ is the auxiliary graph defined above. For MCP, weights $\hat{w}_e$ for arcs $e \in \overline{E}$ are introduced and the problem becomes a MWSSP.

In Cornaz and Jost, 2008 it is explain how to transform a weighted graph $(G, w)$ to an auxiliary weighted graph $(\hat{G}, \hat{w})$ so that:

**Theorem 1 (Cornaz and Jost, 2008).** $\alpha(\hat{G}, \hat{w}) + \chi_{\text{max}}(G, w) = 1^T w$.

The first step is to order the vertices $v_1, \ldots, v_n$ of $V$ so that $i < j$ if and only if $w(v_i) \geq w(v_j)$. (The order is uniquely defined if and only if all the weights are different). Now we let $\overrightarrow{G}$ be the corresponding acyclic orientation of $\overline{G}$, that is, each edge $uv$ of $\overline{G}$ is orientated from $u$ to $v$ whenever $u = v_i$ and $v = v_j$ with $i < j$. (Hence, $\overrightarrow{G}$ is not necessarily uniquely defined if some vertices have the same weight). Each edge of $L(\overline{G})$ corresponds to a pair of incident arcs of $\overrightarrow{G}$. The weight $\hat{w}_e$ of a vertex $e$ of $\hat{G}$ (arc of $\overrightarrow{G}$) is $w_v$, where $v$ is the head of $e = uv$ (i.e., the vertex of minimum weight).

A simplicial star of $\overrightarrow{G}$ is a subset of arcs all having the same tail and whose heads are pairwise linked. For any coloring $S_1, \ldots, S_p$ of $G$, each $S_i$ is a clique of $\overrightarrow{G}$, and, since the orientation of $\overrightarrow{G}$ is acyclic, any of these cliques $S_i$ has a unique representative $v_i$, which is the center of a simplicial star $\overrightarrow{S_i}$ of $\overrightarrow{G}$ which spans $S_i$. Remark that

$$|S_i| - |\overrightarrow{S_i}| = 1$$

|2.30|

Moreover,

$$\sum_{v \in S_i} w(v) - \sum_{e \in \overrightarrow{S_i}} \hat{w}(e) = w(v_i) \quad \text{and} \quad w(v_i) = \max_{v \in S_i} w(v)$$

|2.31|

A simplicial stellar forest of $\overrightarrow{G}$ is a subset of arcs of $\overrightarrow{G}$ which can be partitioned into vertex-disjoint simplicial stars. Observe that given a simplicial stellar forest $\overrightarrow{F} = \overrightarrow{S_1} \cup \ldots \cup \overrightarrow{S_p}$ of $\overrightarrow{G}$, this corresponds to a stable set in $\hat{G}$. Since the converse holds as well, it follows that there is a 1-to-1 correspondence between the stable sets of $\hat{G}$ and the clique partition of $\overrightarrow{G}$, and hence with the colorings of $G$. In the following, we let
\(v_1, \ldots, v_p\) be the centers of the simplicial stars of \(\overrightarrow{G}\) (it is a representative set), and we will use the following

\[
\text{If } v \in \{v_1, \ldots, v_p\}, \text{ then } \delta^-(v) \cap \overrightarrow{F} = \emptyset, \text{ otherwise } |\delta^-(v) \cap \overrightarrow{F}| = 1 \quad (2.32)
\]

where \(\delta^-(v)\) is the subset of \(\overrightarrow{E}\) the head of which is \(v\), and an isolated vertex is the center of an empty star. Similarly, \(\delta^+(v)\) is the set of all arcs of \(\overrightarrow{G}\) the tail of which is \(v\). Theorem 1 follows from (2.31) by summing over all centers \(v_i\). It allows to formulate any instance of MCP as an instance of MWSSP.

In Figures 2.1 and 2.2, we depict an example of the transformation. On the left of Figure 2.1 we report a graph \(G\) with 9 vertices and 24 edges. For each vertex \(v\) we report the weight \(w_v\). On the right of the figure we report the oriented complement graph \(\overrightarrow{G}\). The simplicial stellar forest \(\overrightarrow{F}\) is depicted by bold arrows. On the left of Figure 2.2, we report the line graph \(L(\overrightarrow{G})\), where each vertex corresponds to a pair of incident arcs of \(\overrightarrow{G}\) (not depending on the orientation). On the right of the figure, we depict the auxiliary graph \(\hat{G}\), where edges corresponding to simplicial pairs in \(\overrightarrow{G}\) (i.e., \((v_9,v_1)(v_9,v_2)\) and \((v_6,v_9)(v_6,v_5)\)) are removed. For each vertex \(u, v\) we report the corresponding weight \(w_{u,v} = w_{uv}\). Vertices in the stable set corresponding to \(\overrightarrow{F}\) are represented by larger circles. An optimal solution requires 4 colors (see (2.1)), and the optimal solution value is \(\chi_{\text{max}}(G, w) = w(v_1) + w(v_4) + w(v_6) + w(v_8) = 7 + 5 + 9 + 8 = 29\). The value \(1^\top w\) is \(w(v_1) + w(v_2) + w(v_3) + w(v_4) + w(v_5) + w(v_6) + w(v_7) + w(v_8) + w(v_9) = 7 + 3 + 4 + 5 + 7 + 9 + 8 + 9 + 9 = 60\). A maximum weight stable set (see figure 2.2) is composed by the vertices \((v_1, v_2), (v_4, v_3), (v_6, v_5), (v_7, v_8), (v_6, v_9)\). The value \(\chi_{\text{max}}(G, w)\) is equal to \(3 + 4 + 7 + 8 + 9 + 9 = 31\), by subtracting it to \(1^\top w\) we get \(60 - 31 = 29\), i.e., the value \(\chi_{\text{max}}(G, w) = 29\).

\[\text{Figure 2.1: Example: a graph } G \text{ (left) and the orientation of its complement } \overrightarrow{G} \text{ (right).}\]

In the two following propositions, we give ILP formulations for Equitable Coloring Problem (ECP) and the Bin Packing Problem with Conflict (BPPC). Both are obtained from the basic MWSSP formulation in \(\overrightarrow{G}\) by adding constraints.

- Given a graph \(G\), a coloring \(S = S_1, \ldots, S_p\) is equitable if \(|S_i| - |S_j| \leq 1\) for each \(i, j = 1, \ldots, p\). The Equitable Coloring Problem (ECP) is to find an equitable coloring with minimum \(p\). We denote by \(\chi_{\text{eq}}(G)\) the optimum of ECP. The
most recent mathematical programming contributions to the exact solution of the ECP include the Branch-and-Cut algorithm by Bahiense et al., 2014, which exploits an asymmetric formulation; and the ILP formulation by Méndez-Díaz, Nasini and Savarín Méndez-Díaz, Nasini, and Severín, 2014, which is strengthened by valid inequalities derived from a polyhedral study, and can be solved directly by an ILP solver.

- Given a graph $G$, vertex weights $w \in \mathbb{Z}^{|V(G)|}$, and an nonnegative integer $\kappa$, the Bin Packing Problem with Conflict (BPPC) is to determine a coloring $S = S_1, \ldots, S_p$ of $G$ which minimizes $p$ so that the weight of the stable sets $w(S_i) = \sum_{v \in S_i} w(v)$ does not exceed $\kappa$. The optimum is denoted by $\chi_{bp}(G,c,\kappa)$. State-of-the-art algorithms for the BPPC, recently proposed by Fernandes-Muritiba et al., 2010, Elhedhli et al., 2011, Sadykov and Vanderbeck, 2013, exploit a Set Covering formulation and implement a Branch-and-Price framework. Apparently, the presence of capacity constraints on the cardinality of the color classes reduces the practical difficulty of solving the BPPC. The mentioned algorithms can solve to optimality instances with up to 500 vertices.

**Proposition 1.** $\chi_{eq}(G) = \min 1^T x$ over all $x \in STAB(\hat{G})$ satisfying

$$x(\delta^+(u)) - x(\delta^-(v)) \leq 1 + x(\delta^-(v))\left(|\delta^+(u)| - 1\right) \quad \text{for all } u, v \in V \quad (2.33)$$

where $x(A)$ stands for $\sum_{a \in A} x_a$.

**Proof of Proposition 1.** If $u$ is not a center, then $x(\delta^+(v)) = 0$ and the inequality is trivially satisfied. If $v$ is not a center, then by (2.32), we have $x(\delta^-(v)) = 1$ and again the inequality is trivially satisfied. When both $u$ and $v$ are centers, the constraints ensures that $|\overrightarrow{S_u} - |\overrightarrow{S_v}| \leq 1$ where $\overrightarrow{S_u}, \overrightarrow{S_v}$ are the simplicial stars of $u, v$. By (2.30), the coloring is 1-balanced. \hfill $\square$

**Proposition 2.** $\chi_{bp}(G, w, \kappa) = \min 1^T x$ over all $x \in STAB(\hat{G})$ satisfying
Chapter 2. Reformulation of Coloring Problems on Graphs

\[ w_v + \sum_{z : vz \in \delta^+(v)} w(z) \cdot x_{vz} \leq \kappa \quad \text{for all vertex } v \in V \]  
(2.34)

**Proof of Proposition 2.** If \( v \) is not a center, then \( w(v) \leq \kappa \) otherwise the problem is unfeasible. If \( v \) is a center of a simplicial star \( \tilde{S}_v \), then the left-hand-side of the constraint is equal to the weight \( w(S_v) \) of the corresponding stable set. \( \square \)

### 2.1.1 Extended Formulations

In order to improve the strength of the linear programming relaxation, and to remove the symmetry of compact ILP formulations, new formulation has been proposed in the literature. Let us introduce the following exponential-size collection \( \mathcal{S} \) of stable sets of \( G \):

\[ \mathcal{S} = \{ S \subseteq V : uv \notin E, \forall u, v \in S \} . \]  
(2.35)

A valid model for the VCP can be obtained by introducing, for each subset \( S \in \mathcal{S} \), a binary variable \( \xi_S \) with the following meaning:

\[ \xi_S = \begin{cases} 
1 & \text{if vertices in } S \text{ take the same color} \\
0 & \text{at least two vertices in } S \text{ receive different colors} 
\end{cases} \quad S \in \mathcal{S} \]

then the extended ILP formulation reads as follows:

\[ \chi(G) = \min \sum_{S \in \mathcal{S}} \xi_S \]  
(2.36)

\[ \sum_{S \in \mathcal{S} : v \in S} \xi_S = 1 \quad v \in V \]  
(2.37)

\[ \xi_S \in \{0,1\} \quad S \in \mathcal{S} , \]  
(2.38)

where the objective function (2.36) minimizes the number of stable sets (colors), whereas constraints (2.37) ensure each vertex is colored. Finally constraints (2.38) impose all variables be binary.

Exploiting the idea of having each color class initialized by a “special” vertex, the classical set covering formulation for the VCP by Mehrotra and Trick (1996a) can be extended to the WVCP. Let \( W^* \) denote the set of different weights that appear in \( V \), i.e., \( W^* = \{ w^* : \exists v \in V, w(v) = w^* \} \), and \( \mathcal{S}_{w^*} \) the family of all the independent sets of \( G \) having the heaviest vertex of weight \( w^* \):

\[ \mathcal{S}_{w^*} = \{ S \subseteq V : uv \notin E, \forall u, v \in S ; w(v) \leq w^*, \forall v \in S \} . \]  
(2.39)

We introduce, for each independent set \( S \in \mathcal{S}_{w^*} \) and for each \( w^* \in W^* \), a binary variable \( \xi_S \) which can take value 1 when all the vertices of \( S \) receive the same color (as for the VCP). For each \( S \in \mathcal{S}_{w^*} \), the associated cost is \( w^* \), which represents the cost to be paid if independent set \( S \) is assigned a color. We obtain the following ILP
model for MVCP:

\[
\chi_{\text{max}}(G, w) = \min \sum_{w^* \in W^*} \sum_{S \in \mathcal{S}_{w^*}} w^* \cdot \xi_S
\]  
\[
\sum_{w^* \in W^*} \sum_{S \in \mathcal{S}_{w^*}} \xi_S = 1 \quad v \in V; 
\]  
\[
\xi_S \in \{0, 1\} \quad w^* \in W^*, S \in \mathcal{S}_{w^*}. 
\]

Objective function (2.40) minimizes the sum of the costs of the selected independent sets (each of them will correspond to a color class), while constraints (2.41) impose that, for each vertex, at least one independent set containing the vertex is selected. Finally constraints (2.42) impose all variables be binary.

Branch-and-Price algorithms. In this section we describe the main components of the branch-and-price algorithms necessary to solve the previous two exponential-size formulations. The models have exponentially many binary variables corresponding to the independent sets of \( G \). The discussion will be done for the VCP and only the differences for the MVCP will be pointed out. By relaxing the integrality constraint (2.38) to:

\[
\xi_S \geq 0 \quad S \in \mathcal{S} 
\]

we obtain the so-called master problem, whose rounded-up optimal solution value is a lower bound on \( \chi(G) \). To solve the master problem, we iteratively consider a subfamily of the family \( \mathcal{S} \), of all the independent sets of \( G \), i.e., at each iteration, we have a restricted master problem. By solving the restricted master problem to optimality we obtain the values (denoted as profits in the following) \( \pi_v, v \in V \), of the dual variables associated with constraints (2.37). Dual constraints read:

\[
\sum_{v \in S} \pi_v \leq 1 \quad S \in \mathcal{S}. 
\]

The variables (stable sets) to be added to the restricted master problem correspond to violated dual constraints. To detect such violated constraints, we need to solve the pricing problem (PP). This PP corresponds to finding an stable set \( S^* \) for which (2.44) is violated, that is, an stable set of total profit larger than 1. If the optimal solution value of PP if greater than 1, then we have found a stable set with negative reduced cost, we add the corresponding column to the restricted master problem and iterate. Otherwise, the current restricted master problem contains all the columns corresponding to an optimal solution, and hence the master problem is optimally solved.

To detect violated constraints for the MCP, we need to solve one PP for each \( w^* \in W^* \), i.e., finding an independent set of total profit larger than \( w^* \) in \( G_{w^*} = (V_{w^*}, E_{w^*}) \), which is the subgraph of \( G \) induced by the subset of vertices \( V_{w^*} = \{v : w(v) \leq w^*\} \). Thus, for each \( w^* \in W^* \), the slave problem can be encoded by the following ILP, where binary variables \( x_v, (v \in V_{w^*}) \), take value 1 when vertex \( v \) belongs to \( S^* \) and 0 otherwise:

\[
\max \sum_{v \in V_v} \pi_v x_v 
\]
\[
x_v + x_u \leq 1 \quad vu \in E_{w^*} 
\]
\[
x_v \in \{0, 1\} \quad v \in V_{w^*}. 
\]
Model (2.45)–(2.47) defines a Maximum Weighted Stable Set Problem (MWSSP), with profits (weights) $\pi^*$.

Both PP problems can be solved using the Branch-and-Bound for the MWSSP recently proposed by Held, Cook and Sewell Held, Cook, and Sewell, 2011 (publicly available at https://code.google.com/p/exactcolors/) and used as a sub-routine in their branch-and-price algorithm for the VCP. This algorithm can also be stopped as soon as an stable set with profit larger than a specified threshold is found.

A branching scheme is necessary to obtain integer solutions, this can be done following the idea proposed by Zykov, 1949a for VCP. We now discuss only the necessary extensions to the MCP. The idea of this rule is to select two vertices $i$ and $j$ such that $(i, j) \notin E$, and then consider the two subproblems which are obtained 1) by collapsing $i$ and $j$ into a single vertex $k$, with $(k, h) \in E$ for every vertex $h$ for which $(i, h) \in E$ or $(j, h) \in E$, and 2) by adding an edge between $i$ and $j$, i.e., by setting $E = E \cup (i, j)$. The branching scheme is robust in the sense that, after branching, the problem to be solved is still a MVCP; in case 1) the graph has one node less, in case 2) the graph has one additional edge.

### 2.1.2 Computational Experiments

We derived the auxiliary graph $\hat{G} = (\hat{V}, \hat{E})$ for the VCP and the MVCP as described in the previous section, and solved the resulting MSSP formulated as the ILP (2.25)-(2.29). In the following, we denote this solution technique by $MWSS(\hat{G})$. In addition, we tested the Representative formulation (2.15)-(2.19), we denote this solution technique by $REPf(G)$. This ILP formulation is solved by the general purpose commercial ILP solver CPLEX 12.6. Results are compared with state-of-the-art algorithms for the corresponding problems, all based on Branch-and-Price for the VCP and MVCP. An important parameter in these experiments is the density of the graph, denoted by $\delta$, which is the number of edges divided by the maximum possible number of edges (so $\delta(G) + \delta(\hat{G}) = 1$).

MWSSPs can be solved by means of specialized algorithms, or formulated as a ILP and directly tackled by a ILP solver. In our preliminary experiments we considered the following algorithms for solving MWSSPs: i) the Cliquer algorithm by Ostergard, Ostergård, 2002; ii) the Combinatorial Branch and Bound algorithm by Held et al. Held, Cook, and Sewell, 2012a; iii) the MIP formulation of the MWSSP solved by CPLEX 12.6. According to our experiments, the best choice is to tackle directly the ILP formulation of the MWSSP by CPLEX. This result is partially in contrast to what could be expected, since specialized algorithms are known to be faster when a MWSSP has to be solved as a column generation subproblem in a Branch-and-Price algorithm (see, e.g., Malaguti, Monaci, and Toth, 2011a; Held, Cook, and Sewell, 2012a; Furini and Malaguti, 2012b). Note that in the column generation case, the MWSSP to be solved are defined on smaller-size graphs. From our computational experience, for very large graphs, like the ones obtained from the transformation, specialized algorithms tend to be slower. In the following we report results obtained by solving the ILP formulation of the MWSSP by CPLEX 12.6.

**Computational results for the VCP.** Several Branch-and-Price algorithms have been proposed in the literature for the VCP, e.g., Malaguti, Monaci, and Toth, 2011a, Guandalini and Malucelli, 2012a and Held, Cook, and Sewell, 2012a; none of them clearly dominates the others, hence, we compare our results with Malaguti, Monaci, and Toth, 2011a, which reports an extensive set of computational results for benchmark
2.1. The Vertex Coloring and Max-Coloring problems

instances. Among the 115 DIMACS benchmark instances\(^1\) considered in Malaguti, Monaci, and Toth, 2011a, we restricted the set to 58 instances for which the auxiliary graph \(\hat{G}\) has at most 30,000 vertices. In Table 2.1 we report the cardinality of the vertex sets \(|V(G)|\) and \(|V(\hat{G})|\), and of the edge sets \(|E(G)|\) and \(|E(\hat{G})|\), respectively, for the original and transformed graphs. Clearly, the size of \(V(\hat{G})\) remains limited for dense graphs \(G\), which are the instances where the method we study is more promising. To allow a fair comparison on results obtained with different machines, a benchmark program (dfmax) together with a benchmark instance (r500.5) are available. Computing times obtained on different machines can be scaled with respect to the performance obtained on this program. In our experiments, we scale time limits according to the relative speed of the computer we use, but we report original computing times from the respective papers.

In the table we then report the results obtained by the \(\text{MWSS}(\hat{G})\) and \(\text{REPf}(G)\) with a limit of 6.5 hours of computing time, on a single node of a cluster containing machines equipped with an Intel Xeon E3-1220 processor at 3.10 GHz and 8 GB RAM. This machine needs 4.5 seconds to run the benchmark program on the instance. We report, for each considered instance and each method, a lower bound \(LB\) and an upper bound \(UB\) on the value \(\chi(G)\), and the corresponding computing time \((tl\text{ when time limit is reached})\), which, for \(\text{MWSS}(\hat{G})\) includes the (very small) time for applying the transformation of \(G\) to \(\hat{G}\). In the next columns, we report the results of the Branch-and-Price algorithm in Malaguti, Monaci, and Toth, 2011a (columns BP Malaguti, Monaci, and Toth, 2011a), obtained within 10 hours of computing time on a Pentium 4 PC with 2 GB RAM, which needs 7 seconds to solve the benchmark program on the instance.

On the considered 58 instances, the \(\text{REPf}(G)\) method performs very well solving to optimality 41 instances. The \(\text{MWSS}(\hat{G})\) method can solve 39 instances, with comparable computing times with some notable exception, for instance DSJC125.1 and instances of the Insertion and FullIns sets, where the computing times of the \(\text{REPf}(G)\) are shorter for solved problems. Despite one would expect CPLEX ILP solver to be more effective in solving models formulated as (unweighted) stable set problems, our computational tests showed that instead this is not the case. In particular, we observed that the LP solver of CPLEX tends to spend more time when solving the linear programming relaxation of the \(\text{MWSS}(\hat{G})\) than the \(\text{REPf}(G)\). Both methods perform very well when compared with the 39 instances optimally solved by Malaguti, Monaci, and Toth, 2011a. In particular \(\text{MWSS}(\hat{G})\) can solve in less than one hour instance DSJC250.9, which was only recently solved to optimality by Held, Cook, and Sewell, 2012a with almost 3 hours of computing time.

**Computational results for the MVCP.** The best performing algorithm for Max-col, i.e., the Branch-and-Price algorithm Furini and Malaguti, 2012b, can solve instances with up to 100 vertices. This makes the \(\text{MWSS}(\hat{G})\) method an excellent candidate alternative for solving the Max-col, since we expect the weighted auxiliary graph \(\hat{G}\) being not too large, and we do not have to spoil the MWSSP formulation with additional constraints. We considered all the instances in Furini and Malaguti, 2012b. This includes 46 instances from the COLOR02/03/04 benchmark\(^2\) and two sets of instances from matrix-decomposition problems. Time limits are set to 1 hour of computing time for the first to two sets (DIMACS and \(p\)) and to 4 hours for the last set (\(R\)), as in Furini and Malaguti, 2012b. We also tested the \(\text{REPf}(G)\) but, for

\(^1\)ftp://dimacs.rutgers.edu/pub/challenge/graph/

\(^2\)http://mat.gsia.cmu.edu/COLOR02
### Chapter 2. Reformulation of Coloring Problems on Graphs

#### Table 2.1: Results for the VCP, DIMACS benchmark instances.

| instance | $|V(G)|$ | $|E(G)|$ | $G_{MWSS}(G)$ | $\text{REP}_f(G)$ | B&P |
|----------|--------|--------|---------------|-----------------|-----|
| DSJC125.1 | 125    | 736    | 5 5 17314.7   | 5 5 65.5       | 5 5 |
| DSJC125.5 | 125    | 3891   | 14 20 tL      | 14 20 tL       | 17 17 |
| DSJC125.9 | 250    | 6961   | 44 44 0.5     | 44 44 0.5     | 44 44 3916 |
| DSJC250.5 | 250    | 3218   | 5 17 tL       | 5 10 tL        | 6 8  |
| DSJC250.9 | 250    | 15668  | 48 48 18      | 48 40 tL       | 20 28  |
| DSJC500.9 | 500    | 27897  | 72 72 1935.2  | 72 72 2401.5   | 71 72  |
| DSJ500.1c | 121275 | 85 85 1.3 | 85 85 1.3     | 85 85 2885.5 |

| instance | $|V(G)|$ | $|E(G)|$ | $G_{LB}$ | $G_{UB}$ | $G_{time}$ | $B&P_{LB}$ | $B&P_{UB}$ | $B&P_{time}$ |
|----------|--------|--------|---------|---------|----------|----------|----------|----------|
| DSJC125.1 | 125    | 736    | 5 5     | 17314.7 | 65.5     | 5 5 142  |
| DSJC125.5 | 125    | 3891   | 14 20   | tL      | 14 20    | 17 17 18290 |
| DSJC125.9 | 250    | 6961   | 44 44   | 0.5     | 44 44    | 44 44 3916 |
| DSJC250.5 | 250    | 3218   | 5 17    | tL      | 5 10     | 6 8  |
| DSJC250.9 | 250    | 15668  | 48 48   | 18      | 40       | 20 28  |
| DSJC500.9 | 500    | 27897  | 72 72   | 1935.2  | 72 72    | 2401.5 |
| DSJ500.1c | 121275 | 85 85 1.3 | 85 85 1.3 | 85 85 2885.5 |

Note: The table above shows the results for various instances, including their variable and edge counts, lower bounds, upper bounds, and solution times, along with benchmark performance (B&P).
2.2. The Partition Coloring Problem

The Partition Coloring Problem (PCP) is a generalization of the VCP where the vertex set is partitioned and exactly one vertex of each subset of the partition has to be colored. The PCP consists of finding a partial coloring \( \tilde{C} \) such that:

\[
\begin{align*}
& (i) \quad |\tilde{V} \cap P_i| = 1 \text{ for } i = 1, 2, \ldots, k; \\
& (ii) \quad f(v) \neq f(w) \text{ for all } v, w \in \tilde{V}, vw \in E; \\
& (iii) \quad h \text{ is minimum.}
\end{align*}
\]

The minimum number of colors used in any optimal PCP solution is denoted in this manuscript as \( \chi_P(G, \mathcal{P}) \).

Let us introduce an example, called Example 1. In the left part of Figure 2.3, we depict a graph \( G \) of ten vertices and thirteen edges. The graph is partitioned in five subsets \( (k = 5) \), each subset is composed of two vertices; the dotted lines are used to identify the subsets of the partition. In the right part of Figure 2.3, we depict a feasible partial coloring \( \tilde{C} \) using two colors (gray and black). For each subset of the partition exactly one vertex is colored. The colored vertices, i.e., the vertices \( v \in \tilde{V} \), are colored with the corresponding color (red and blue) while the uncolored ones are white.

The PCP models many real-world applications (see Demange et al. (2015)) including: routing and wavelength assignment, dichotomy-based constraint encoding, antenna positioning and frequency assignment, as well as a wide variety of...
Chapter 2. Reformulation of Coloring Problems on Graphs

Figure 2.3: Example: (left) a graph $G$ and a partition of its vertices in 5 subsets ($k = 5$); (right) a feasible partition coloring of $G$ with two colors (red and blue).

scheduling problems (timetabling, quality test, berth allocation) and a variant of the classical Travelling Salesman Problem.

**Literature review.** The PCP was introduced by Li and Simha (2000) to model wavelength routing and assignment problems. Only two works described exact algorithms for the PCP: Frota et al. (2010) and Hoshino, Frota, and De Souza (2011). The first one proposes a branch-and-cut algorithm based on the asymmetric representatives formulation introduced by Campêlo, Campos, and Corrêa (2008b) and Campêlo, Corrêa, and Frota (2004) for the VCP. A number of valid inequalities are introduced and used within a branch-and-cut framework. Computational tests are reported on randomly generated instances (called random), VCP instances from the literature, and instances derived from the routing and wavelength assignment literature (including the nsf instances, and a new set of instances called ring). The second exact algorithm, i.e., the one presented in Hoshino, Frota, and De Souza (2011), is a branch-and-price algorithm based on the Dantzig-Wolfe reformulation of the representatives formulation. Each vertex is the representative of one color, and the color can be used if and only if the associated vertex takes the color. In order to deal with an exponential number of variables, a column generation scheme has been proposed which is based on a set of pricing problems, one for each vertex.

**Compact ILP Formulation.** In this section we first introduce a natural ILP formulation for the PCP and then we derive a new extended formulation based on the Dantzig-Wolfe reformulation of the natural formulation. A trivial upper bound on the number of colors used in any optimal PCP solution is given by the number $k$ of subsets of the partition. We can then introduce a set of binary variables $y$ with the following meaning:

$$y_c = \begin{cases} 
1 & \text{if color } c \text{ is used} \\
0 & \text{otherwise}
\end{cases} \quad c = 1, 2, \ldots, k;$$

and a set of binary variables $x$ with the following meaning:

$$x_{vc} = \begin{cases} 
1 & \text{if vertex } v \text{ is colored with color } c \\
0 & \text{otherwise}
\end{cases} \quad v \in V, \quad c = 1, 2, \ldots, k.$$
The first natural ILP formulation (called ILP\(^N\)) reads as follows:

\[
\chi_P(G, \mathcal{P}) = \min \sum_{c=1}^{k} y_c \quad (2.48)
\]

\[
\sum_{c=1}^{k} \sum_{v \in P_i} x_{vc} = 1 \quad i = 1, 2, \ldots, k \quad (2.49)
\]

\[
x_{vc} + x_{uc} \leq y_c \quad uv \in E, \ c = 1, 2, \ldots, k \quad (2.50)
\]

\[
x_{vc} \in \{0, 1\} \quad v \in V, \ c = 1, 2, \ldots, k \quad (2.51)
\]

\[
y_c \in \{0, 1\} \quad c = 1, 2, \ldots, k, \quad (2.52)
\]

where the objective function (2.48) minimizes the number of used colors, constraints (2.49) impose that one vertex per subset of the partition is colored, and constraints (2.50) impose that adjacent vertices do not receive the same color. Finally, constraints (2.51) and (2.52) define the variables of the formulation.

By replacing constraints (2.51) and (2.52) with

\[
x_{vc} \geq 0 \quad v \in V, \ c = 1, 2, \ldots, k \quad (2.53)
\]

\[
y_c \geq 0 \quad c = 1, 2, \ldots, k, \quad (2.54)
\]

we obtain the Linear Programming relaxation of ILP\(^N\), that will be denoted as LP\(^N\) in what follows.

Descriptive natural models for coloring problems are known to produce weak linear programming relaxations and are affected by symmetry (see Malaguti and Toth (2010) and Cornaz, Furini, and Malaguti (2016)), hence, in general they can be solved to optimality only for small graphs.

### 2.2.1 Extended Formulation

In order to improve the strength of the linear programming relaxation, and to remove the symmetry of model (2.48)–(2.52), we convexify constraints (2.50) through Dantzig-Wolfe reformulation (see Desaulniers, Desrosiers, and Solomon, 2006, Bergner et al., 2011b and Bergner et al., 2015b). Let us introduce the following exponential-size collection \(\mathcal{S}\) of stable sets of \(G\) which intersect each subset of the partition at most once:

\[
\mathcal{S}_P = \{ S \subseteq V : uv \notin E, \ \forall u, v \in S ; \ |S \cap P_i| \leq 1, \ i = 1, \ldots, k \} . \quad (2.55)
\]

A valid model for the PCP can be obtained by introducing, for each subset \(S \in \mathcal{S}_P\), a binary variable \(\xi_S\) with the following meaning:

\[
\xi_S = \begin{cases} 
1 & \text{if vertices in } S \text{ take the same color} \\
0 & \text{otherwise}
\end{cases} \quad S \in \mathcal{S}_P
\]
then the extended ILP formulation reads as follows:

\[
\chi_P(G, \mathcal{P}) = \min \sum_{S \in \mathcal{S}} \xi_S \quad (2.56)
\]

\[
\sum_{S \in \mathcal{S} : \left| S \cap P_i \right| = 1} \xi_S = 1 \quad i = 1, \ldots, k \quad (2.57)
\]

\[
\xi_S \in \{0, 1\} \quad S \in \mathcal{S}, \quad (2.58)
\]

where the objective function (2.56) minimizes the number of stable sets (colors), whereas constraints (2.57) ensure that exactly one vertex of each subset of the partition is colored. Finally constraints (2.58) impose all variables be binary. Constraints (2.57) can be rewritten as follows:

\[
\sum_{S \in \mathcal{S} : \left| S \cap P_i \right| = 1} \xi_S \geq 1 \quad i = 1, \ldots, k, \quad (2.59)
\]

since it is always possible to transform a solution of model (2.56), (2.59) and (2.58) into a solution of model (2.56)–(2.58) of same value. Constraint (2.59) ensures that the associated dual variables take non negative values. The resulting formulation (2.56)-(2.59)-(2.58) is denoted as \( \text{ILP}^E \) in the following sections.

Finally, by relaxing the integrality of constraints (2.58) to

\[
\xi_S \geq 0 \quad S \in \mathcal{S}, \quad (2.60)
\]

we obtain the Linear Programming relaxation of \( \text{ILP}^E \), that is denoted as \( \text{LP}^E \) in what follows.

By observing that \( \text{ILP}^E \) is obtained by applying Dantzig-Wolfe reformulation of constraints (2.50) of \( \text{ILP}^N \) and since constraints (2.50) do not form a totally unimodular matrix, it follows that the quality of the lower bound obtained solving the LP relaxation of \( \text{ILP}^N \) is dominated by its counterpart associated with \( \text{ILP}^E \):

**Proposition 1.** The optimal value of \( \text{LP}^E \) is greater or equal than the optimal value of \( \text{LP}^N \).

**Proof.** Proving the proposition for the specific PCP models give more insight on the structure of the LP relaxation optimal solutions of \( \text{ILP}^N \) and \( \text{ILP}^E \). We first show that any feasible solution of \( \text{LP}^E \) can be converted into a feasible solution of \( \text{LP}^N \) with the same objective function value. Given a function \( p(v) \) which returns the corresponding index \( i \) \( i = 1, 2, \ldots, k \) of the subset of the partition of a vertex \( v \) \( v \in V \), we can uniquely define the color \( c(S) \) of any \( S \in \mathcal{S} \) as \( \min_{v \in S} p(v) \). Let \( \xi^* \) denote a feasible solution of \( \text{LP}^E \) and assume, without loss of generality, that no subset of the partition is covered by more than one selected subset \( S \in \mathcal{S} \). Let us define a solution \( (x^*, y^*) \) as follows: for each color \( c \) set

\[
y^*_c = \sum_{S \in \mathcal{S} : c = c(S)} \xi^*_S \quad \text{and} \quad x^*_{vc} = \sum_{S \in \mathcal{S} : c = c(S), v \in S} \xi^*_S. \quad (2.61)
\]

Thus, inequalities (2.57) ensure that constraints (2.49) are satisfied. Observe that, by construction, for each edge \( vw \in E \) and for each color \( c = 1, 2, \ldots, k \) we have \( x^*_{vc} + x^*_{uc} \leq y^*_c \); thus, \( (x^*, y^*) \) is feasible to \( \text{LP}^N \) and the objective function value remains unchanged.
We then show a case where the optimal value of LP\textsuperscript{E} is strictly larger than the optimal value of LP\textsuperscript{N}. Consider the instance of 2.4, where we depict a graph G of ten vertices and twenty edges. The graph is partitioned into five subsets (k = 5), and each subset is composed of two vertices. As in 2.3, the dotted lines define the subsets of the vertex partition. The figure report also a numbering of the vertices of the graph. Let consider the following solution of LP\textsuperscript{E}:

$$\xi^*_{S_1} = \xi^*_{S_2} = \xi^*_{S_3} = \xi^*_{S_4} = \xi^*_{S_5} = 0.5$$

where the five stable sets are $S_1 = \{1, 8\}$, $S_2 = \{1, 9\}$, $S_3 = \{2, 9\}$, $S_4 = \{2, 10\}$, and $S_5 = \{8, 10\}$. This solution has value 2.5 and it is optimal. Indeed, since in G all stable sets intersecting each partition at most once have at most 2 vertices, then at least 2.5 of them are necessary to cover one vertex per element of the partition. The optimal solution value of LP\textsuperscript{E} is larger than the optimal solution value of LP\textsuperscript{N}, which is 1, and it is obtained by setting $y_c = 0.2$ ($c = 1, \ldots, 5$) and $x_{vc} = 0.1$ ($v \in V, c = 1, \ldots, 5$).

\[\ \]

**Figure 2.4:** Example: a graph G of 10 vertices and a partition of its vertices in 5 subsets (k = 5).

Model ILP\textsuperscript{E} has exponentially many $\xi_S$ variables ($S \in \mathcal{S}_P$), which cannot be explicitly enumerated for large-size instances. Column Generation (CG) techniques are therefore necessary to efficiently solve ILP\textsuperscript{E}. In the following section we present a new Branch-and-Price framework for ILP\textsuperscript{E}, and refer the interested reader to Desaulniers, Desrosiers, and Solomon, 2006 for further details on CG.

**A New Branch-and-Price Algorithm.** There are two main ingredients of a Branch-and-Price algorithm, i.e., a CG algorithm to solve the Linear Programming Relaxation of the exponential-size integer model, and a branching scheme. We discuss separately these two aspects in the next sections.

**Solving the Linear Programming Relaxation of ILP\textsuperscript{E}.** Model (2.56), (2.59) and (2.60), initialized with a subset of variables containing a feasible solution, is called the Restricted Master Problem (RMP). Additional new variables, needed to solve LP\textsuperscript{E} to optimality, can be obtained by separating the following dual constraints:

$$\sum_{i=1,2,\ldots,k : |P \cap S| = 1} \pi_i \leq 1 \quad S \in \mathcal{S}_P,$$

where $\pi_i$ ($i = 1, 2, \ldots, k$) is the dual variable associated with the $i$-th constraint (2.59). Accordingly, the CG performs a number of iterations, where violated dual
Chapter 2. Reformulation of Coloring Problems on Graphs

Constraints are added to the RMP in form of primal variables, and the RMP is re-optimized, until no violated dual constraint exist. At each iteration, the so-called Pricing Problem (PP) is solved. This problem asks to determine (if any) a stable set $S^* \in \mathcal{S}_P$ for which the associated dual constraint (3.17) is violated, i.e., such that

$$\sum_{i=1,2,\ldots,k : |P_i \cap S^*| = 1} \pi^*_i > 1,$$

where $\pi^*$ is the optimal vector of dual variables of RMP at each iteration of the CG procedure.

The pricing problem can be modeled as a Maximum Weight Stable Set Problem (MWSSP) on an auxiliary graph $\hat{G} = (V, \hat{E})$, constructed as follows: the vertex set of $\hat{G}$ coincides with the vertex set of $G$, while the edge set $\hat{E}$ is constructed from the edge set of $G$ and its partition $\mathcal{P} = \{P_1, \ldots, P_k\}$:

$$\hat{E} = E \cup \{uv : u, v \in P_i, i = 1, \ldots, k\}.$$  (2.64)

In other words, each subset of the partition of $G$ is transformed into a clique in $\hat{G}$. Given a weight vector $c \in \mathbb{R}^{|V|}$, where the weight $c_v$ of the vertex $v \in P_i$ is set to the value $\pi^*_i$ associated with the $i$-th subset of the partition, the pricing problem corresponds then to a MWSSP in $\hat{G}$, that is, to determine a stable set $S$ of $\hat{G}$ maximizing $\sum_{v \in S} c_v$.

Notice that since each partition subset has been turned into a clique, such a stable set contains at most one vertex per subset $P_i$ and therefore collects each profit $\pi_i$ at most once. The MWSS can be solved by means of a specialized combinatorial Branch-and-Bound algorithm (see 4.2).

If a stable set $S^*$ has total weight larger than one (that is, the reduced cost is negative), the associated column is added to the RMP and the problem is re-optimized. If, on the other hand, the total weight is not larger than 1, by linear programming optimality conditions no column can improve the objective function of the RMP and therefore LP$^E$ is solved to proven optimality.

**Branching scheme for ILP$^E$.** The design of a branching scheme is crucial for the performance of a branch-and-price algorithm (Vanderbeck, 2011). In the following we describe the branching scheme adopted in our new Branch-and-Price framework. Two are its main properties. Firstly, it is a complete scheme, i.e., it ensures that integrality can be imposed in all cases. Secondly, it does not require modifications neither on the master problem nor the pricing algorithm. The latter means that our branching does not alter the structure of the pricing problem so that the same algorithm can be applied during the entire search.

Consider a fractional solution $\xi^*$ of LP$^E$, at a given node of the branching tree, and let $\mathcal{S} \subseteq \mathcal{J}_P$ be the set of columns in the RMP at the node. We propose a branching scheme FF composed of two rules applied in sequence, i.e., when the branching condition for the first rule fails, the second is applied.

The first branching rule is designed to impose that exactly one vertex is colored for each subset. Constraints (2.59) impose that the sum of the values of the variables associated with stables sets intersecting each subset is at least one, but in a fractional solution these stable sets can include different vertices in the same subset of the
2.2. The Partition Coloring Problem

A given subset $P_i$ has more than one (partially) colored vertex if:

$$|\{v \in P_i : \sum_{S \in \mathcal{F}, v \in S} \xi^*_S > 0\}| > 1.$$ (2.65)

In case more than one of such subsets exists, we select the subset $i$ with the largest number of (partially) colored vertices, breaking ties by size of the subsets (preferring smaller subsets and breaking further ties randomly). We then branch on the vertex $v \in P_i$ with the largest value of $\sum_{S \in \mathcal{F}, v \in S} \xi^*_S$. Two children nodes are created:

- in the first node we impose that $v$ is the colored vertex for subset $P_i$;
- in the second node, we forbid that $v$ is the colored vertex for subset $P_i$.

This branching rule can be enforced without any additional constraint neither for the RMP nor for the pricing problem. To force the coloring of $v$ in the children nodes of the branching scheme, we remove from the graph $G$ all other vertices $u \in P_i$ ($u \neq v$); to forbid the coloring of $v$, we simply remove the vertex from the graph $G$. This first branching rule is not complete since it may happen that a vertex (partially) belongs to more than one stable set in the solution $\xi^*$.

If this happens for a vertex $v$, there must be another vertex $u$ (belonging to a different subset of the partition) such that:

$$\sum_{S \in \mathcal{F}, v,u \in S} \xi^*_S = \gamma, \gamma \text{ is fractional.}$$ (2.66)

We say that $v$ and $u$ are a fractionally colored pair of vertices.

The second branching rule is designed to impose that each pair of (colored) vertices either takes the same color, or the two vertices of the pair take different colors. This rule has been proposed for the VCP by Zykov (1949b) and used to derive several effective Branch-and-Price algorithms for the VCP, starting from the seminal work by Mehrotra and Trick (1996b), see, e.g., Malaguti, Monaci, and Toth, 2011b; Gualandi and Malucelli, 2012b; Held, Cook, and Sewell, 2012b. In case more pairs of fractionally colored vertices exist, we select the pair $v$ and $u$ with the largest $\gamma$ value. Two children nodes are then created:

- in the first node we force vertices $v$ and $u$ to take the same color;
- in the second node we force vertices $v$ and $u$ to take different colors.

The second branching rule can also be enforced without any additional constraint neither for the RMP nor for the pricing problem. To force different colors for a pair of vertices $v$ and $u$ in the children nodes of the branching scheme, we add the edge $vu$ to $E$. On the other hand, to force $v$ and $u$ to take the same color, we remove $v$ and $u$ from the graph $G$ and replace them with a new vertex $z$; then we add edges $zw$ for all $w \in V$ if $uw \in E$ or $vw \in E$ (or both). We then consider a stable set containing the vertex $z$ coloring both $P_{p(v)}$ and $P_{p(u)}$, where the function $p(v)$ ($v \in V$) returns the index of the subset of the partition containing vertex $v$.

In our Branch-and-Price algorithm we first define the vertices for each subset of the partition to be colored, i.e., we apply the first branching rule. Then, in case the solutions are still fractional, we apply the second branching rule in order to obtain integer solutions.

After branching, the variables that are incompatible with the branching decision are removed from the children nodes. The following proposition states that the two
Chapter 2. Reformulation of Coloring Problems on Graphs

proposed branching rules define a complete branching scheme for ILP$^E$:

**Proposition 2.** The two branching rules applied in sequence provide a complete branching scheme for model ILP$^E$.

**Proof.** After the application of the first branching rule, the colored vertex in each subset of the partition is determined. In Barnhart et al., 1998 it is proved that for any 0-1 constraint matrix $A$ (as for the case of LP$^E$), if a basic solution $\xi^*$ to $A\xi = 1$ is fractional, then there exist two rows $i$ and $j$ such that:

\[
0 < \sum_{S \in \mathcal{P} : i,j \in S} \xi^*_S < 1 \tag{2.67}
\]

This result allows us to conclude that if a solution is fractional then we can determine two subsets of the partition such that (2.67) holds. The same holds for the case in which $A\xi^* > 1$: in any optimal fractional solution to LP$^E$, the rows for which covering constraints are satisfied with equality must be covered by at least two columns with associated fractional variables, and the previous result applies. By picking the colored vertex from the first and the colored vertex from the second subset, the two vertices constitute a fractionally colored pair of vertices on which to apply the second branching rule. 

### 2.2.2 Computational Experiments

The experiments have been performed on a single core of a computer equipped with a 3.25 GHz 4-core i5 processor and 8Gb RAM, running a 64-bit Linux operating system. The goal of the experiments is to assess the performance of the new B&P algorithm setting a time limit of 1 hour of computing time. The algorithms were coded in C++ and all the codes were compiled with gcc 6.2 and -O3 optimizations. At each iteration of the Column Generation procedure (see Section 2.2.1), we used Cplex 12.6.1 as a Linear Programming solver (ran single-threaded). The pricing MWSS subproblems were solved using the open-source implementation of the algorithm described in Held, Cook, and Sewell (2012b) and available at https://github.com/heldstephan/exactcolors. The algorithm implementation and the instances used are available at Santini (2017).

**Instances.** In order to compare our results with the ones present in the literature, we tested our approach on the instance classes random, nsfnet and ring presented in 2.2. The entire set of instances can be downloaded at http://www2.ic.uff.br/~celso/grupo/pcp.htm. In their work, Hoshino, Frota, and De Souza (2011) consider a subset of 187 out of a total of 199 instances, removing those instances solved to optimality in less than a second by either their algorithm or that of Frota et al. (2010). We removed 12 instances of the ring class, as we realised that they were identical copies of the same three basic instances. In particular, instances n10_p1.0_1 to n10_p1.0_5 all correspond to the same instance (therefore only 1 out of the 5 instances has been kept), as do the analogous instances of base type n15 and n20. This reduced the total number of instances to 175. We used therefore 56 random, 32 nsfnet, and 87 ring instances.

In this section, we report the results obtained by our new branch-and-price algorithm with a time limit of 1 hour. Since linear relaxation bounds are generally tight, the performance of our algorithm depends on the ability to find good feasible
2.2. The Partition Coloring Problem

solutions (upper bounds) early in the branching tree. Therefore, at the root node we adopt the following approach based on an ALSN+LS (see ), denoted as root node heuristic (RH).

The new B&P algorithm can solve at the root node 20 out of 56 random instances with a computing time smaller than 6 seconds, all the 32 nsfnet instances with a computing time smaller than 22 seconds, and 72 out of 87 ring instances with a computing time smaller than 342 seconds.

Table 2.2 presents the results relative to the instances which cannot be solved at the root node. We report under B&P Hoshino, Frota, and De Souza, 2011 the best results obtained by any of the four implementations of Hoshino, Frota, and De Souza (2011), i.e., the final lower and upper bounds, while under New B&P the results obtained by our algorithm. Columns rLB and rUB are the lower and upper bounds obtained at the end of the exploration of the root node, while column UB is the final upper bound. Column Nodes displays the number of explored Branch-and-Price nodes. Column rCols is the number of columns generated at the root node, solving the MWSSP; column nCols is the average number of columns generated solving the MWSSP, at nodes other than the root node. Column Cols reports the total number of columns generated; this includes columns generated solving the MWSSP, as well as those generated by the heuristic, and by the POPULATE method. Finally, columns rTime and Time list, respectively, the root node and the overall solution time (both including the initial heuristic), in seconds. The values in bold under columns UB denote instances solved to optimality. An asterisk next to the total computation time denotes instances solved in more than 1800s. This is done to facilitate comparison with Hoshino, Frota, and De Souza (2011), as the authors use a time limit of 1800s in their work. Notice that we could not solve the root node of one instance (ring_n20p1.0s1), for which we provide a lagrangean lower bound \( LB = \lceil z_{LP}E / z_{Viol} \rceil \), where \( z_{LP}E \) is the solution of the last linear relaxation of the restricted master problem solved, and \( z_{Viol} \) is the last solution value found by the pricing problem.

In summary, the new B&P algorithm is able to find the optimal solution to 34 out of 36 random instances and 12 out of 15 ring instances, and in 38 cases the solution was found in less than half an hour (1800s).
## Chapter 2. Reformulation of Coloring Problems on Graphs

| Instance | \(|V|\) | \(|E|\) | B&P (lit.) | B&P (new) |
|----------|-------|-------|------------|------------|
| n120p5t2s1 | 120 | 3616 | 8 8 | 4992 135 1.00 5251 4.70 438.52 |
| n120p5t2s2 | 120 | 3563 | 8 8 | 538 150 3.59 2223 4.79 31.40 |
| n120p5t2s3 | 120 | 3638 | 8 8 | 11993 127 0.71 8747 4.65 1148.29 |
| n120p5t2s4 | 120 | 3565 | 8 8 | 490 125 3.24 1833 4.35 22.41 |
| n120p5t2s5 | 120 | 3653 | 8 8 | 9020 145 1.56 3158 4.75 1626.14 |
| n70p5t2s2 | 70 | 1204 | 6 6 | 21 62 8.85 302 1.23 1.52 |
| n70p5t2s3 | 70 | 1218 | 6 6 | 229 66 4.53 1178 1.13 2.92 |
| n70p5t2s4 | 70 | 1217 | 6 6 | 35 63 8.53 431 1.18 1.59 |
| n80p5t2s3 | 80 | 1611 | 6 6 | 2998 75 1.21 3776 1.51 38.38 |
| n80p5t2s4 | 80 | 1595 | 6 6 | 497 76 2.76 1536 1.47 6.02 |
| n80p5t2s5 | 80 | 1634 | 6 6 | 497 68 3.17 1733 1.70 6.18 |
| n90p112s1 | 90 | 445 | 2 3 | 483 234 32.54 15981 7.75 399.67 |
| n90p112s2 | 90 | 442 | 2 3 | 1451 211 27.28 39905 6.93 * 2082.66 |
| n90p112s3 | 90 | 465 | 3 3 | 63 204 36.74 2635 3.93 33.64 |
| n90p112s5 | 90 | 485 | 3 3 | 17 219 49.56 1074 5.84 20.98 |
| n90p212s1 | 90 | 823 | 3 4 | 4881 181 6.91 34938 3.98 * 2496.60 |
| n90p212s3 | 90 | 869 | 3 4 | 281 163 15.16 4512 3.49 73.09 |
| n90p212s4 | 90 | 821 | 3 4 | 5133 176 9.27 47854 3.97 tl |
| n90p212s5 | 90 | 862 | 3 4 | 1019 178 13.28 13815 3.84 341.48 |
| n90p312s1 | 90 | 1215 | 4 5 | 9085 126 3.21 29454 2.43 1652.40 |
| n90p312s2 | 90 | 1234 | 4 5 | 9293 146 3.15 29573 2.32 * 2054.82 |
| n90p312s3 | 90 | 1275 | 5 5 | 197 139 8.62 1921 2.36 18.33 |
| n90p312s4 | 90 | 1211 | 4 5 | 11771 145 3.19 37847 2.77 tl |
| n90p312s5 | 90 | 1268 | 5 5 | 865 141 6.75 6072 2.42 77.84 |
| n90p412s1 | 90 | 1624 | 5 6 | 12455 114 1.46 18361 2.01 1052.99 |
| n90p412s2 | 90 | 1600 | 5 6 | 5398 113 2.42 8681 2.03 159.76 |
| n90p412s3 | 90 | 1650 | 6 6 | 865 117 3.98 3661 1.99 29.44 |
| n90p412s4 | 90 | 1638 | 6 6 | 1777 127 3.36 6182 2.05 72.11 |
| n90p412s5 | 90 | 1671 | 6 6 | 53 113 9.60 712 2.15 4.65 |
| n90p512s1 | 90 | 2039 | 7 7 | 81 99 6.70 716 2.12 3.83 |
| n90p512s2 | 90 | 1988 | 7 7 | 6357 93 1.12 7321 2.01 175.09 |
| n90p512s3 | 90 | 2064 | 7 7 | 23 86 10.00 388 2.22 2.97 |
| n90p612s1 | 90 | 2463 | 8 8 | 494 73 1.88 1103 2.32 5.75 |
| n90p612s2 | 90 | 2478 | 9 9 | 1741 82 0.97 1897 2.51 13.63 |
| n90p812s2 | 90 | 3200 | 12 12 | 12 | 327 4.02 49 38 | 3.11 3.35 |
| n90p812s3 | 90 | 3282 | 12 12 | 13 | 279 2.64 40 | 37 3.49 3.65 |

### Table 2.2: Computational results for instances not solved at the root node.
Chapter 3

Reformulation of Knapsack Problems

The classical Knapsack Problem (KP) is one of the most famous problems in combinatorial optimization. Given a knapsack capacity $C$ and a set $N = \{1, \ldots, n\}$ of items, the $j$-th having a profit $p_j$ and a weight $w_j$, KP asks for a maximum profit subset of items whose total weight does not exceed the capacity. KP can be formulated using the following Integer Linear Program (ILP):

$$\max \left\{ \sum_{j \in N} p_j x_j : \sum_{j \in N} w_j x_j \leq C, x_j \in \{0, 1\}, j \in N \right\}$$

(3.1)

where each variable $x_j$ takes value 1 if and only if item $j$ is inserted in the knapsack.

KP is NP-hard, although in practice fairly large instances can be solved to optimality within low running time. The reader is referred to Martello and Toth, 1990a; Kellerer, Pferschy, and Pisinger, 2004 for comprehensive surveys on applications and variants of this problem.

3.1 The Knapsack Problem with Setup

In this section we consider a generalization of KP arising when items are associated with operations that require some setup time to be performed. In particular, there is a given set $I = \{1, \ldots, m\}$ of classes associated with items, and each item $j$ belongs to a given class $t_j \in I$. A positive setup cost $f_i$ is incurred and a positive setup capacity $s_i$ is consumed in case items of class $i$ are selected in the solution. Without loss of generality, we assume that all input parameters have integer values. The resulting problem is known in the literature as Knapsack Problem with Setup (KPS).

KPS has been first introduced in the literature by Lin, 1998 in a survey of non-standard knapsack problems worthy of investigation. Motivated by an industrial application in a packing industry, KPS was studied by Chebil and Khemakhem, 2015; this article presented a basic dynamic programming scheme and an improved version of the algorithm, with a reduced storage requirement, that proved able to solve instances with up to 10000 items and 30 classes. Recently, KPS has also been addressed in Pferschy and Scatamacchia, (forthcoming) and Della Croce, Salassa, and Scatamacchia, 2017.

Preliminaries. We will denote by $n_i$ the number of items in each class $i \in I$. Without loss of generality, we assume that items are sorted according to their class, i.e., class $i$ includes all items $j \in K_i := [\alpha_i, \beta_i]$, where $\alpha_i = \sum_{k=1}^{i-1} n_k + 1$ and
\( \beta_i = \alpha_i + n_i - 1 \). Moreover, we assume that, within each class, items are sorted according to non-increasing profit over weight ratio, i.e.,

\[
\frac{p_j}{w_j} \geq \frac{p_{j+1}}{w_{j+1}} \quad j = \alpha_i, \ldots, \beta_i - 1; \quad i \in I.
\]

Let us introduce a first numerical example, called Example 1 and reported in Figure 3.1. The optimal solution value of Example 1 is 132 and the corresponding solution takes both items of the second class.

A natural model for KPS is obtained by introducing \( x_j \) variables that have the same meaning as in (3.1), and decision variables \( y_i \) associated with item classes: in particular, each variable \( y_i \) takes value 1 if and only if some item of class \( i \) is included in the solution. The resulting model is as follows

\[
\begin{align*}
\text{max} & \quad \sum_{j \in N} p_j x_j - \sum_{i \in I} f_i y_i \\
\text{s.t.} & \quad \sum_{j \in N} w_j x_j + \sum_{i \in I} s_i y_i \leq C \quad (3.3) \\
& \quad x_j \leq y_j, \quad j \in N \quad (3.4) \\
& \quad x_j \in \{0, 1\}, \quad j \in N \quad (3.5) \\
& \quad y_i \in \{0, 1\}, \quad i \in I. \quad (3.6)
\end{align*}
\]

The objective function (3.2) maximizes the total profit of the selected items minus the setup cost of the used classes, whereas constraint (3.3) takes into account that the sum of the item weights and the class setups must not exceed the capacity. Inequalities (3.4) force a class to be used whenever some item of the class is selected. Finally, (3.5)-(3.6) impose all variables to be binary. It is worth mentioning that constraints (3.4)-(3.5) and the objective function force the \( y \) variables to be binary; thus, in principle, constraints (3.6) are redundant. The resulting model, denoted as M1 in the following, has \( n + m \) variables and \( n + 1 \) constraints, plus variable domain constraints.

By replacing constraints (3.5)-(3.6) with the following ones:

\[
\begin{align*}
x_j & \in [0, 1], \quad j \in N \quad (3.7) \\
y_i & \in [0, 1], \quad i \in I \quad (3.8)
\end{align*}
\]

we obtain the LP relaxation of M1. We will denote by LP1 this relaxation and by \( U_1 \) the associated upper bound value.

- \( m = 2, n = 4, C = 152 \).
- \( n_1 = 2, \alpha_1 = 1, \beta_1 = 2, n_2 = 2, \alpha_2 = 3, \beta_2 = 4 \).
- \( f_1 = 10, s_1 = 10, f_2 = 9, s_2 = 6 \).
- \( p_1 = 84, w_1 = 75, p_2 = 75, w_2 = 72, p_3 = 70, w_3 = 64, p_4 = 71, w_4 = 78 \).

FIGURE 3.1: Example 1
We now present an extended model which contains an exponential number of variables. Let us introduce the following collections \( \mathcal{S}_i \) \( (i \in I) \) of feasible subsets of items \( S \subseteq K_i \) satisfying the knapsack capacity \( C \):

\[
\mathcal{S}_i = \left\{ S \subseteq K_i : \sum_{j \in S} w_j \leq C - s_i \right\}.
\] (3.9)

For each item subset \( S \in \mathcal{S}_i \), we can define its profit and weight taking also into account the setup cost and capacity of the corresponding class \( i(S) \):

\[
P_S = \sum_{j \in S} p_j - f_{i(S)}, \quad W_S = \sum_{j \in S} w_j + s_{i(S)}.
\]

A valid model for KPS can be obtained by introducing, for each subset \( S \in \mathcal{S}_i \) \( (i \in I) \), a binary variable \( \xi_S \) which takes value 1 if and only if subset \( S \) is included in the solution:

\[
\max \sum_{i \in I} \sum_{S \in \mathcal{S}_i} P_S \xi_S \tag{3.10}
\]

\[
\sum_{i \in I} \sum_{S \in \mathcal{S}_i} W_S \xi_S \leq C \tag{3.11}
\]

\[
\sum_{S \in \mathcal{S}_i} \xi_S \leq 1 \quad i \in I \tag{3.12}
\]

\[
\xi_S \in \{0, 1\} \quad i \in I, S \in \mathcal{S}_i. \tag{3.13}
\]

Objective function (3.10) maximizes the total profit of the selected subset of items, whereas constraint (3.11) ensures that the solution satisfies the capacity constraint. Inequalities (3.12) impose that at most one subset is selected for each class, whereas constraints (3.13) impose all variables be binary. The resulting formulation, denoted as M2 in the following, corresponds to the classical formulation of the Multiple-Choice Knapsack Problem (MCKP) with inequality constraints; see Johnson and Padberg, 1981.

By replacing constraints (3.13) with the following ones:

\[
\xi_S \geq 0 \quad i \in I, S \in \mathcal{S}_i, \tag{3.14}
\]

we obtain the linear programming relaxation of M2, that will be denoted as LP2 in what follows. Let \( U^2 \) be the associated upper bound value. Note that constraints (3.12) implicitly provide an upper bound of value 1 on the \( \xi_S \) variables, thus we do not need to impose this bound in (3.14). A similar formulation has also been studied in Chajakis and Guignard, 1994 and Michel, Perrot, and Vanderbeck, 2009.

The quality of the upper bound \( U^2 \) cannot be worse than its counterpart associated with the first model.

**Proposition 3.** Bound \( U^2 \) dominates bound \( U^1 \).

**Proof.** To prove the result, it is enough to consider the model, say \( M1^+ \), obtained by adding to M1 the following valid constraints:

\[
\sum_{j \in K_i} w_j x_j \leq C - s_i y_i, \quad i \in I. \tag{3.15}
\]
We first observe that these inequalities are implied by (3.3), hence their addition does not change the feasible region of the LP relaxation of \( M^1 + \) with respect to that of \( M^1 \). We now apply Dantzig Wolfe Reformulation (DWR) to \( M^1 + \) (see Desrosiers and Lübbecke, 2005) through a convexification of constraints (3.4) and (3.15). This is equivalent to impose the LP relaxation of such reformulation to be equal to the intersection of constraints (3.3) with the convex hull of constraints (3.4), (3.5), (3.6) and (3.15). The resulting model is exactly \( M^2 \). One of the basic properties of DWR is that the LP relaxation of the reformulated model is as strong as the original model. Moreover, since the convexified region does not have the integrality property (i.e., the constraints (3.15) strictly contain their convex hull), this fact implies that there exists at least one objective function for which the reformulation of \( M^1 + \) (i.e., \( M^2 \)) has a strictly better LP relaxation value than \( M^1 \).

For the sake of completeness, we consider again the instance of Example 1. In this case, the optimal solution of LP2 is \( \xi^*_S \) where the two subsets are \( S_1 = \{2\} \) and \( S_2 = \{3, 4\} \), and belong to classes 1 and 2, respectively. For these item sets we have \( P_{S_1} = 74, P_{S_2} = 132, W_{S_1} = 85 \) and \( W_{S_2} = 148 \). Thus, the optimal solution value is \( U^2 = 135.482 \), i.e., it is lower than \( U^1 \), which is 144.254.

**Solving the LP relaxation of \( M^1 \).** In this section we consider the LP relaxation of model \( M^1 \). We observe that Yang and Bulfin, 2009 introduces some properties of an optimal solution to LP1 and derives a combinatorial algorithm for its solution. However, no detailed description nor analysis on the computational complexity of the algorithm is given in Yang and Bulfin, 2009. Thus, for the sake of completeness, we report a pseudo-code of the algorithm in Figure 3.2. In addition, we show that the algorithm can be executed in linear time, thus improving the time complexity over the result claimed in Akinc, 2006. Finally, we propose a strengthened relaxation that can be computed in constant time if an optimal solution to LP1 is known.

**Algorithm LP1:**

initialize: \( \mathcal{N} := \emptyset \);

for each class \( i \in I \) do

    compute the break item \( b_i \);
    define the macro-item \( I_i \);
    \( \mathcal{N} := \mathcal{N} \cup \{I_i\} \cup \{b_i + 1, \ldots, \beta_i\} \);
end do

Solve the LP relaxation of the KP instance defined by item set \( \mathcal{N} \), and let \( \theta \) be the associated solution;

for each class \( i \in I \) do

    set \( y^*_i = \theta_{I_i} \), and \( x^*_j = \theta_{I_i} \) \( \forall j = \alpha_i, \ldots, b_i \);

    set \( x^*_j = \theta_j \) \( \forall j = b_i + 1, \ldots, \beta_i \);
end do

**Figure 3.2:** Algorithm to compute an optimal solution to LP1.

In this algorithm, one has to compute, for each class \( i \), the break item \( b_i \) and the associated macro-item \( I_i \). The former is the first item of the class for which the ratio between the cumulative profit and the cumulative weight is larger than the profit over weight ratio of all subsequent items (if no such item exists, the break item is the last item of the class), while the latter is an artificial item that simulates the profit and
weight incurred when taking all items \( j \in [\alpha_i, b_i] \) at the same value. For this reason, its profit (resp. weight) is equal to the sum of the profits (resp. weights) of all items \( j \in [\alpha_i, b_i] \), plus the cost and capacity required for using the class. Given these figures, the algorithm adopts the well-known strategy for computing an optimal solution to the LP relaxation of KP, i.e., it takes one (either original or macro) item at a time until the entire capacity is used. The last selected item (or macro item) can then be taken at a fractional value. Finally, an optimal solution to LP1 is derived, observing that, for each class \( i \), taking the associated macro-item at a given value, say \( \theta I_i \), corresponds to taking at the same value both \( y_i \) and all \( x_j \) variables associated with items \( j \in [\alpha_i, b_i] \).

In an optimal LP1 solution, though many items may be taken at a fractional value, at most one \( y \) variable may be fractional. A similarity of LP1 with the LP relaxation of KP is given by the following observation:

**Observation 1.** An optimal solution to LP1 can be computed in \( O(n) \) time.

To reach the claimed complexity, it is enough to observe that, for each class \( i \in I \), the break item \( b_i \) can be computed in \( O(n_i) \) time. This can be done with a split argument similar to that proposed by Balas and Zemel, 1980 for finding the critical item in a KP, yielding an overall \( O(n) \) time complexity. As the resulting number of items to pack in the second step cannot be larger than \( n \), this gives a linear time algorithm.

We conclude this section showing a strengthened relaxation that exploits the fact that the optimal LP1 solution has at most one fractional \( y \) variable and that, in any integer solution, this variable must take either value 0 or 1. Let \( p(t) \) and \( w(t) \) denote the profit and the weight, respectively, of the (either original or cumulative) item that is selected at each iteration \( t \) of the Dantzig’s algorithm used to compute the LP1; in addition, let \( \bar{t} \) be the number of iterations executed by the algorithm and \( \tau \) denote the residual capacity before inserting the last item. Similar to the MT bound proposed for KP by Martello and Toth, 1977, we can derive the following upper bound for KPS

\[
UB = \max\{UB_0, UB_1\} \tag{3.16}
\]

where

\[
UB_0 = \sum_{t=1}^{\bar{t}-1} p(t) + \tau \frac{p(\bar{t}+1)}{w(\bar{t}+1)} \quad \text{and} \quad UB_1 = \sum_{t=1}^{\bar{t}-1} p(t) + \left( p(\bar{t}) - (w(\bar{t}) - \tau) \frac{p(\bar{t}-1)}{w(\bar{t}-1)} \right)
\]

represent an upper bound on the optimal solution value when the fractional item is fixed to 0 and 1, respectively. In case \( \bar{t} = 1 \), this improved bound cannot be computed. Otherwise, it can be easily seen that \( UB \) dominates the bound produced by LP1 and that the computational effort for computing this bound is negligible if an optimal solution to LP1 has been computed.

**Solving the LP relaxation of M2.** Model M2 has exponentially many \( \xi_S \) variables \((i \in I, S \in \mathcal{S})\), which cannot be explicitly enumerated for large-size instances. *Column Generation* (CG) techniques are then necessary to efficiently solve its linear programming relaxation. In the following we discuss the CG framework for M2, and refer the interested reader to Desrosiers and Lübbecke, 2005 for further details on CG.
Chapter 3. Reformulation of Knapsack Problems

Model (3.10)–(3.12) and (3.14), initialized with a subset of variables containing a feasible solution, is called Restricted Master Problem (RMP). Additional new variables, needed to solve LP2 to optimality, can be obtained by separating the following dual constraints:

\[ W_S \lambda + \pi_i \geq P_S \quad i \in I, S \in \mathcal{I}_i, \]

(3.17)

where \( \pi_i \ (i \in I) \) is the dual variable associated with the \( i \)-th constraint (3.12) and \( \lambda \) is the dual variable associated with constraint (3.11). Accordingly, CG performs a number of iterations, until no violated dual constraint exist. At each iteration, the so-called Pricing Problem (PP) associated with each class \( i \in I \) is solved. This problem asks to determine (if any) a subset \( S^* \subseteq \mathcal{S}_i \) for which the associated dual constraint (3.17) is violated, i.e., such that

\[ \sum_{j \in S^*} (p_j - \lambda^* w_j) > \pi_i^* + \lambda^* s_i + f_i, \]

(3.18)

where \( \pi_i^* \ (i \in I) \) and \( \lambda^* \) are the dual variables values associated to the current solution of the RMP.

The pricing problem for class \( i \) asks for determining a subset of items \( S^* \subseteq \mathcal{S}_i \) that maximizes the left-hand-side of (3.18), and checking if this is larger than \( \pi_i^* + \lambda^* s_i + f_i \). As such, finding the maximally violated dual constraint can be modeled as a KP, where each item \( j \in K_i \) has profit \( p_j - \lambda^* w_j \) and weight \( w_j \). Using binary variables \( \theta_j \ (j \in K_i) \), the problem reads as follows:

\[
\tau^* = \max \left\{ \sum_{j \in K_i} (p_j - \lambda^* w_j) \theta_j : \sum_{j \in K_i} w_j \theta_j \leq C - s_i, \theta_j \in \{0,1\}, \forall j \in K_i \right\},
\]

(3.19)

where \( \theta_j = 1 \) if and only if item \( j \) belongs to subset \( S^* \). All variables with negative reduced costs that are generated, i.e., such that \( \tau^* > \pi_i^* + \lambda^* s_i + f_i \) (if any), are added to the RMP, which is then re-optimized, according to a classic column generation scheme. If no column with negative reduced cost exists, the RMP is optimally solved and its solution (value) corresponds to the linear programming relaxation (value) of M2.

Since the pricing problems ask for the solution of a KP for each class, solving RMP (and, eventually, LP2) is weakly NP-hard. Computing an optimal solution for RMP at each CG iteration corresponds to solving the LP relaxation of the classical ILP formulation for the MCKP (see also section 2.3) for which a linear-time algorithm is known in the literature (see [13]). We use this algorithm to compute an optimal RMP primal solution and to identify the so-called critical item. Since in LP2, variables are associated to subset of items instead of items, we will denote the critical item as the critical subset. Finally, an optimal RMP dual solution can be characterized thanks to the following proposition:

**Proposition 4.** Let \( \hat{S} \) be the critical subset in an optimal primal solution of RMP. Then, an optimal solution to the associated dual is the following:

\[
\lambda^* = \frac{P_S - P_{S^{-1}}}{W_S - W_{S^{-1}}}, \quad \pi_i^* = \max_{S \in \mathcal{I}_i} \{P_S - W_S \lambda^*\}.
\]
3.1. The Knapsack Problem with Setup

Proof. It is easy to verify that \((\lambda^*, \pi^*)\) satisfies the dual constraints. We hence need to show that the primal and dual solutions have same objective value.

Let us denote by \(i\) the class associated with the critical subset \(\hat{S}\). The value of the optimal primal solution can be written as follows:

\[
\sum_{i \in I} \sum_{S \in \mathcal{A}_i} P_S \xi_S = \sum_{i \in I, i \neq \hat{i}} \sum_{S \in \mathcal{A}_i} P_S \xi_S + P_{\hat{S}-1} + C_r \lambda^*
\]

Since \(C_r = C - \left( \sum_{i \in I, i \neq \hat{i}} \sum_{S \in \mathcal{A}_i} W_S \xi_S + W_{\hat{S}-1} \right)\) we have

\[
\sum_{i \in I} \sum_{S \in \mathcal{A}_i} P_S \xi_S = \sum_{i \in I, i \neq \hat{i}} \sum_{S \in \mathcal{A}_i} (P_S - \lambda^* W_S) \xi_S + C \lambda^* + (P_{\hat{S}-1} - \lambda^* W_{\hat{S}-1}) \leq \sum_{i \in I} \pi_i^* + C \lambda^*
\]

where the latter inequality derives from the definition of \(\pi_i^*\) variables and from the fact that, for each class \(i\), exactly one subset is selected in the primal solution. As the objective function of the dual is \(C \lambda + \sum_{i \in I} \pi_i\), the weak duality theorem ensures that \((\pi_i^*, \lambda^*)\) is an optimal dual solution. \(\square\)

A branch-and-bound(-and price) for KPS. Our enumerative algorithm is based on the observation that KPS reduces to KP in case the set of item classes to be selected is given. This suggests a branching rule in which first-stage decisions are associated with the classes, whereas variables associated with items are treated as second-stage variables. For the sake of simplicity, in this section we will refer to the first formulation of KPS, i.e., we will make use of \(x\) and \(y\) variables to refer to selection of items and classes, respectively.

Figure 3.3 reports the pseudocode of the algorithm that is executed at each node of the tree. We first solve the LP relaxation at the current node and check whether the node can be fathomed, comparing the local upper bound with the incumbent solution, say \(z^*\). In case enumeration must continue, we use a branching scheme similar to the one proposed by Horowitz and Sahni, 1974 for KP: at the root node we sort the classes according to non-increasing profit over weight ratio of the associated macro-item. At each node, we take the first \(y\) variable that is not fixed by branching and define two descendant nodes by fixing this variable to 1 and 0, respectively. Subsequent nodes, if any, are explored in the order they are generated, according to a depth-first strategy. Finally, if all the \(y\) variables are fixed by branching, a backtracking is executed.

Solution of LP1 As branching conditions involve \(y\) variables only, the algorithm described in the previous section for solving LP1 has to be modified as follows. At the root node we store all the original and macro-items, sorted according to profit over weight ratio. At the current node, the local upper bound can be computed simply scanning the list of \(n+m\) items: cumulative items can be used only for classes that are not fixed by branching. Original items can be used only for items that have been selected by branching (i.e., such that \(y_i = 1\)), while items that belong to a class that is forbidden by branching should not be used in the solution. It is easy to check that the computation of the local LP solution takes \(O(n)\) time as at the root node.

Solution of LP2 The same branching scheme can be used with LP2 as well. Since the \(y\) variables are not explicitly considered, the branching decision for a specific
Algorithm \textit{Solve\_node}: 

// LP solution and possible fathoming
solve the LP relaxation at the current node;
let \((x^*, y^*)\) denote an optimal LP solution, and \(U\) the associated value;
if \(U \leq z^*\) then fathom the node and return;
else
  // possible heuristic solution
  if all \(y^*\) variables are integer then
    solve a KP instance defined by items in the selected classes;
    let \(z(KP)\) be the associated profit (including setup costs);
    if \(z(KP) > z^*\) then
      update \(z^* := z(KP)\);
      if \(U \leq z^*\) then fathom the node and return;
    endif
  endif
  // possible branching
  if all \(y\) variables are fixed by branching then fathom the node and return;
  else
    let \(i\) be the first class that is not fixed by branching;
    define two subproblems branching on variable \(y_i\);
  endif
return

\textbf{FIGURE 3.3:} Exploration of a branch-and-bound node.

class \(i\) can be imposed changing the right-hand-side of the associated constraint (3.12) in M2. To impose the condition \(y_i = 1\), the constraint becomes:

\[
\sum_{S \in S_i} \xi_S = 1. \tag{3.20}
\]

On the other side, to impose the condition \(y_i = 0\), the constraint becomes:

\[
\sum_{S \in S_i} \xi_S = 0. \tag{3.21}
\]

These modifications do not change the nature of the formulation nor the associated pricing problems PP. The effect of constraint (3.20) is just to remove the non-negativity constraint on the corresponding dual variable. From a practical viewpoint, imposing \(y_i = 0\) corresponds to disregard item class \(i\) and all the associated items; this makes LP2 easier to solve, since a smaller number of pricing problems has to be solved at each iteration of the column generation process. Finally, since new variables may be generated within the branching nodes, the branch-and-bound algorithm becomes in this case a \textit{branch-and-price} algorithm.

\textbf{A relevant special case.} In this section we introduce a special relevant case that may be encountered when solving KPS. This happens if, for each class \(i \in I\), the following condition is satisfied

\[
s_i + \sum_{j \in K_i} w_j \leq C \tag{3.22}
\]
3.1. The Knapsack Problem with Setup

This means that, for each class $i$, all items of the class can be allocated into the knapsack.

We observe that KPS remains NP-hard also in case assumption (3.22) is valid. Moreover, this special case is relevant from a theoretical viewpoint: while KPS does not admit a polynomial time approximation algorithm with a bounded approximation ratio, there exists an FPTAS that can be derived when assumption (3.22) is satisfied (see, Pferschy and Scatamacchia, (forthcoming)). Finally, this situation is always satisfied for the instances in our testbed that are taken from the literature. Therefore, for the rest of this section we will assume that (3.22) is valid.

**Proposition 5.** Under assumption (3.22), LP2 can be computed in $O(n)$ time.

*Proof.* Consider a given class $i$. If all items in $K_i$ can be inserted in the knapsack, the pricing problem (3.19) for a given $\lambda^*$ has the following optimal solution:

$$
\theta_j = \begin{cases} 
1 & \text{if } p_j - \lambda^* w_j > 0 \\
0 & \text{otherwise} 
\end{cases} \quad (j \in K_i)
$$

Since items are sorted according to non-increasing profit over weight ratio, this means that all items $j \in [\alpha_i, \gamma_i(\lambda^*)]$ will be selected, where $\gamma_i(\lambda^*) = \min \{ j \in K_i : p_j / w_j \leq \lambda^* \}$. Thus, at most $n_i$ variables associated with class $i$ have to be considered into the model—namely, for each item $j \in K_i$, one variable corresponding to item set $[\alpha_i, j]$. Overall, model M2 is thus an MCKP with $n$ variables, whose LP relaxation can be solved in $O(n)$ time using the algorithm presented by Dyer, 1984 and Zemel, 1984.

**Corollary 1.** Under assumption (3.22), $U^1 = U^2$.

*Proof.* This follows from Proposition 3 and by observing that, in case (3.22) is satisfied, constraints (3.15) are equivalent to their convex hull; thus, the convexified region has the integrality property. This implies that no bound improvement can be obtained by the convexification of constraints (3.4) and (3.15).

3.1.1 Computational Experiments

In this section we report an extensive computational analysis on the performances of our algorithms. All codes were implemented using C and were run on an Intel Xeon E3-1220 V2 running at 3.10 GHz.

**Instances from the literature.** Hard KPS instances have been proposed in Chebil and Khemakhem, 2015. These problems, publicly available on internet, were generated to simulate realistic instances from an industrial application. In particular, these instances have been randomly generated with a number of items $n \in \{500, 1000, 2500, 5000, 10000\}$ and a number of classes $m \in \{5,10,20,30\}$; ten instances have been generated for each pair $(n, m)$, thus producing a testbed of 200 problems. Item profits and weights have been generated so as to have strongly correlated instances, and the setup cost (resp. capacity) of each class is a random number correlated to sum of the profits (resp. weights) of the items in the class. Observe that this benchmark has also been used by other recent works on KPS (e.g., Della Croce, Salassa, and Scatamacchia, 2017 and Pferschy and Scatamacchia, (forthcoming)), and that all instances in this set satisfy condition (3.22). Recently, four additional classes of instances have been introduced in Della Croce, Salassa, and Scatamacchia, 2017, we remind the interested reader to FMT17 for detailed results on these instances.
In this section we evaluate the computational performances of exact approaches for KPS. In a set of preliminary computational experiments, we used IBM-ILOG Cplex 12.6 on model M1 and obtained results that are coherent with those reported in Chebil and Khemakhem, 2015; Della Croce, Salassa, and Scatamacchia, 2017: the direct application of an ILP solver is typically unable to solve instances with more than 2500 items in a systematic way, even using state-of-the-art codes. For this reason, we do not report detailed results for this approach in what follows. As to the combinatorial algorithms, we consider B&B_{LP1} and B&P_{LP2}, that denote the Branch-and-Bound and Branch-and-Price algorithms based on the LP relaxations of models M1 and M2, respectively, and an improved dynamic programming algorithm (an improved version of the DP proposed in Chebil and Khemakhem, 2015).

For this analysis, we used the instances of Class 4, and compared our algorithms with the exact algorithms given in Della Croce, Salassa, and Scatamacchia, 2017 and in Pferschy and Scatamacchia, (forthcoming). In the following, these algorithms will be denoted as DSS and PS, respectively. Both algorithms showed to outperform the dynamic programming in Chebil and Khemakhem, 2015 on our benchmark; for this reason, the latter is excluded from comparison. In a personal communication, the authors of Della Croce, Salassa, and Scatamacchia, 2017 provided us with the implementation of their algorithm, which allows a better performance evaluation with our new exact algorithms. Since all our algorithms are sequential, we ran DSS on our machine using 1 thread only and denote by DSS_{ITH} the resulting code (see the next section for a multi-thread performance analysis). We observe that the results for algorithm PS are taken from Pferschy and Scatamacchia, (forthcoming) and were obtained on an Intel i5 CPU running at 3.2 GHz with 16 GB of RAM.

<table>
<thead>
<tr>
<th>Instances</th>
<th>B&amp;B_{LP1}</th>
<th>B&amp;P_{LP2}</th>
<th>DP</th>
<th>DSS_{ITH}</th>
<th>PS</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Time (sec.)</td>
<td>Time (sec.)</td>
<td>Time (sec.)</td>
<td>Time (sec.)</td>
<td>Time (sec.)</td>
</tr>
<tr>
<td>m</td>
<td>n</td>
<td>avg</td>
<td>max</td>
<td>avg</td>
<td>max</td>
</tr>
<tr>
<td>5</td>
<td>500</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.01</td>
</tr>
<tr>
<td>1000</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.01</td>
<td>0.05</td>
</tr>
<tr>
<td>2500</td>
<td>0.00</td>
<td>0.01</td>
<td>0.01</td>
<td>0.01</td>
<td>0.25</td>
</tr>
<tr>
<td>5000</td>
<td>0.00</td>
<td>0.01</td>
<td>0.01</td>
<td>0.01</td>
<td>0.90</td>
</tr>
<tr>
<td>10000</td>
<td>0.01</td>
<td>0.01</td>
<td>0.01</td>
<td>0.02</td>
<td>3.66</td>
</tr>
<tr>
<td>10</td>
<td>500</td>
<td>0.00</td>
<td>0.01</td>
<td>0.00</td>
<td>0.01</td>
</tr>
<tr>
<td>1000</td>
<td>0.00</td>
<td>0.01</td>
<td>0.01</td>
<td>0.01</td>
<td>0.06</td>
</tr>
<tr>
<td>2500</td>
<td>0.01</td>
<td>0.02</td>
<td>0.01</td>
<td>0.02</td>
<td>0.25</td>
</tr>
<tr>
<td>5000</td>
<td>0.01</td>
<td>0.02</td>
<td>0.01</td>
<td>0.02</td>
<td>0.93</td>
</tr>
<tr>
<td>10000</td>
<td>0.02</td>
<td>0.03</td>
<td>0.02</td>
<td>0.03</td>
<td>3.70</td>
</tr>
<tr>
<td>20</td>
<td>500</td>
<td>0.00</td>
<td>0.02</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>1000</td>
<td>0.20</td>
<td>0.52</td>
<td>0.15</td>
<td>0.37</td>
<td>0.06</td>
</tr>
<tr>
<td>2500</td>
<td>3.19</td>
<td>5.98</td>
<td>1.42</td>
<td>2.74</td>
<td>0.25</td>
</tr>
<tr>
<td>5000</td>
<td>3.05</td>
<td>8.46</td>
<td>1.23</td>
<td>3.24</td>
<td>0.94</td>
</tr>
<tr>
<td>10000</td>
<td>7.07</td>
<td>14.63</td>
<td>2.61</td>
<td>5.18</td>
<td>3.76</td>
</tr>
<tr>
<td>30</td>
<td>500</td>
<td>0.00</td>
<td>0.02</td>
<td>0.01</td>
<td>0.03</td>
</tr>
<tr>
<td>1000</td>
<td>3.02</td>
<td>6.45</td>
<td>2.71</td>
<td>5.60</td>
<td>0.06</td>
</tr>
<tr>
<td>2500</td>
<td>18.95</td>
<td>70.47</td>
<td>10.00</td>
<td>37.77</td>
<td>0.25</td>
</tr>
<tr>
<td>5000</td>
<td>1.38</td>
<td>7.86</td>
<td>0.66</td>
<td>3.65</td>
<td>0.94</td>
</tr>
<tr>
<td>10000</td>
<td>11.90</td>
<td>78.98</td>
<td>4.87</td>
<td>30.83</td>
<td>3.76</td>
</tr>
</tbody>
</table>

Table 3.1: Performance comparison of exact solution methods for KPS.
Table 3.1 reports the performance comparison of the sequential algorithms. Each line of the table refers to 10 instances with the same number of classes and items. For each algorithm we report the average and maximum computing time (in seconds) for solving the associated instances.

Results in Table 3.1 show that our enumerative algorithms are competitive with both DSS1TH and PS on this benchmark. There are some specific sets of problems in which they are considerably faster than the previous approaches from the literature, in particular when the number of item classes is “small”, regardless of the number of items. We also observe that M2 is typically faster than M1. Indeed, as condition (3.22) is always verified on these instances, these two algorithms explore a similar number of nodes.

### 3.2 The interval min-max regret knapsack problem

Consider an investor who wants to select the best way to invest a certain capital $c$, among a number of financial products, each requiring a given amount of money $w_j$, and ensuring a fixed return $p_j$. This problem is easily formalized by the classical 0-1 knapsack problem. Given $n$ items, each having an associated profit $p_j$ and weight $w_j$ ($j = 1, 2, \ldots, n$), and a capacity $c$, the 0-1 knapsack problem (01KP) is to select a subset of items which has maximum total profit, and a total weight not exceeding $C$.

While the above model concerns, e.g., financial products such as repurchase agreements (repos), in a more general situation the return of product $j$ can be a priori unknown, and may be expected to vary within a given range $[p^-_j, p^+_j]$. By assuming that we are dealing with a prudent investor (which can quite frequently be the case in times of financial crisis) s/he could aim at minimizing the a posteriori regret of the selected choice with respect to any possible profit scenario. This problem can be formalized by the following “robust” generalization of the 01KP. The profit of each item $j$ can take any value in a range characterized by two values $p^-_j, p^+_j$ for $j = 1, 2, \ldots, n$. A set $s$ of $n$ profits $p^+_j$ satisfying $p^+_j \in [p^-_j, p^+_j]$ for $j = 1, 2, \ldots, n$ is called a scenario.

Let $X$ be the set of all feasible solutions, i.e.,

$$X = \left\{ x = (x_1, x_2, \ldots, x_n) : \sum_{j=1}^{n} w_j x_j \leq C, \ x_j \in \{0, 1\} (j = 1, 2, \ldots, n) \right\}.$$  \hspace{1cm} (3.23)

We denote by $z^s(x)$ the solution value given, for scenario $s$, by a solution vector $x \in X$, i.e.,

$$z^s(x) = \sum_{j=1}^{n} p^+_j x_j.$$  \hspace{1cm} (3.24)

Let $z^*_s$ be the optimal solution value under scenario $s$. The regret associated with a solution $x$, for a scenario $s$, is then

$$r^s(x) = z^*_s - z^s(x).$$  \hspace{1cm} (3.25)

Let $S_0$ denote the set of all possible scenarios $s$, i.e., $S_0 = \{ s : p^+_j \in [p^-_j, p^+_j] (j = 1, 2, \ldots, n) \}$. The maximum regret $r(x)$ of a solution $x$ is then the maximum $r^s(x)$ value over all scenarios, i.e., $r(x) = \max_{s \in S_0} r^s(x)$. The interval min-max regret knapsack problem (MRKP) is to find a feasible solution vector $x$ such that the maximum
regret is minimized. Formally,

\[
\begin{align*}
\text{(MRKP)} \quad & \min_{s \in S_0} \max_s \sum_{j=1}^n w_j x_j \\ 
\text{s.t.} \quad & \sum_{j=1}^n w_j x_j \leq C \quad (j = 1, 2, \ldots, n).
\end{align*}
\]

For the 01KP it is usual to assume that all profits are non-negative, as cases with negative values can be easily handled through a preprocessing (see Martello and Toth 1990b, Section 2.1). For the MRKP, while items \( j \) (financial products) with both \( p_j^- \) and \( p_j^+ \) less than 0 can obviously be removed from the instance, items with \( p_j^- \leq 0 \) and \( p_j^+ \geq 0 \) need a specific handling, which will be discussed in this section. Also, from now on, we will assume that capacity and weights are positive integers.

The 01KP is the special case of the MRKP that arises if \( p_j^- = p_j^+ \) \((j = 1, 2, \ldots, n)\), which implies that the MRKP is \( \mathcal{NP} \)-hard in the ordinary sense. It is an open question whether it is \( \mathcal{NP} \)-hard in the strong sense. Deineko and Woeginger, 2010 recently proved that the decision version of the problem is complete for the complexity class \( \Sigma_2^p \) (see Garey and Johnson 1979, Chapter 7, or Papadimitriou 1994, Chapter 17) and hence is most probably not in \( \mathcal{NP} \). To our knowledge, the only paper devoted to the MRKP is the aforementioned study of its complexity by Deineko and Woeginger, 2010. As recently observed by Deineko and Woeginger, 2010, the solution of even moderately sized instances of the interval min-max regret knapsack is challenging and seems to require innovative approaches. Also note that even computing the regret of a single solution with respect to a single scenario requires the solution of an \( \mathcal{NP} \)-hard problem (a 01KP).

The following basic result (whose roots are in Yaman, Karaşan, and Pinar 2001) has been explicitly proved for the MRKP (in the more general context of binary interval min-max regret problems) by Aissi, Bazgan, and Vanderpooten, 2009:

**Lemma 1.** For any solution \( x \in X \), its worst case scenario is \( \sigma(x) \), defined by

\[
p^\sigma_j = \begin{cases} 
p_j^- & \text{if } x_j = 1; \\
p_j^+ & \text{otherwise.}
\end{cases}
\]

(Intuitively, the scenario inducing the maximum regret has the worst profits for the selected items, and the best profits for the non-selected items.) Hence, from now on, we will restrict our attention, without loss of generality to the subset of scenarios \( S \subseteq S_0 \) induced by Lemma 1, i.e., \( S = \{ \sigma(x) : x \in X \} \). In addition, by observing that there is a unique scenario \( \sigma(x) \in S \) for each solution \( x \in X \), when no confusion arises we will use \( \sigma \) instead of \( \sigma(x) \).

From Lemma 1 and equations (3.23), (3.25) and (3.26), by considering the worst-case scenario of \( x \), the MRKP can be re-written as

\[
\min_{x \in X} \left( \max_{y \in X} \sum_{j=1}^n p_j^\sigma y_j - \sum_{j=1}^n p_j^\sigma x_j \right) = \min_{x \in X} \left( \max_{y \in X} \sum_{j=1}^n p_j^\sigma y_j - \sum_{j=1}^n p_j^\sigma x_j \right)
\]

By further observing that the last summation of (3.30) can be developed without including the \( p_j^+ \) terms (which disappear when \( x_j = 0 \)), we get a simpler formulation
of the MRKP:

\[
\min_{x \in X} (z^\sigma_\star - \sum_{j=1}^{n} p_j^- x_j),
\]

(3.31)

where (recall that we use \( \sigma \) for \( \sigma(x) \))

\[
z^\sigma_\star = \max_{y \in X} \sum_{j=1}^{n} p_j^\sigma y_j = \sum_{j=1}^{n} p_j^\sigma \bar{y}_j^\sigma
\]

(3.32)

and \( \bar{y}_j^\sigma \) \( (j = 1, 2, \ldots, n) \) denotes an optimal solution vector for the 01KP under scenario \( \sigma \).

By observing that, from Lemma 1, for \( x \in X \) we can write \( p_j^\sigma \) as

\[
p_j^\sigma = p_j^+ + (p_j^- - p_j^+) x_j,
\]

(3.33)

and introducing a new (non integer) variable \( \vartheta \), along with a constraint that forces \( \vartheta \) to satisfy \( \vartheta \geq z^\sigma_\star \) for all \( \sigma \in S \), the MRKP is expressed by the mixed integer linear model

\[
M(S) \min \vartheta - \sum_{j=1}^{n} p_j^- x_j
\]

(3.34)

s.t. \( \vartheta \geq \sum_{j=1}^{n} p_j^+ \bar{y}_j^\sigma + \sum_{j=1}^{n} (p_j^- - p_j^+) \bar{y}_j^\sigma x_j \) \( \forall \sigma \in S \) (3.35)

\[
\sum_{j=1}^{n} w_j x_j \leq C
\]

(3.36)

\[
x_j \in \{0, 1\} \quad (j = 1, 2, \ldots, n),
\]

(3.37)

because an optimal solution \((x^\star, \vartheta^\star)\) satisfies \( \vartheta^\star = z^{\sigma(x^\star)}_\star \) for all \( s \in S \).

We will denote by \( M(S) \) the continuous relaxation of \( M(S) \), obtained by relaxing \( (3.37) \) to

\[
0 \leq x_j \leq 1 \quad (j = 1, 2, \ldots, n),
\]

(3.38)

Note that the model has \( n+1 \) variables, but an exponential number of constraints \( (3.35) \) (one per scenario), each of which would require the solution of a 01KP. In the next section we discuss how to deal with such inconvenience.

Lemma 1 can also be used to see how to handle cases where \( p_j^- \leq 0 \) holds for some item \( j \).

**Property 1.** Let \( j \) be an item for which \( p_j^- \leq 0 \). If \( p_j^+ \leq |p_j^-| \) then item \( j \) can be removed from the instance.

**Proof** If \( p_j^- = 0 \), then \( p_j^+ = 0 \) and the claim follows. Hence assume \( p_j^- < 0 \). We show that the regret of a solution \( x \) with \( x_j = 1 \) is never smaller than that of a solution \( x' \) with \( x'_j = 0 \) and \( x'_i = x_i \) for all \( i \neq j \). From Lemma 1, we have \( p_j^{\sigma(x)} = p_j^- \), \( p_j^{\sigma(x')} = p_j^+ \) and \( p_i^{\sigma(x)} = p_i^{\sigma(x')} \) for all \( i \neq j \). Hence

\[
z^{\sigma(x)}(x) = z^{\sigma(x')}(x') - |p_j^-|.
\]

(3.39)
The optimal solutions under scenarios \( \sigma(x) \) and \( \sigma(x') \) satisfy
\[
 z^\sigma_*(x) \geq z^\sigma_*(x') - \max\{0, p_j^+\} \tag{3.40}
\]
because by removing item \( j \) from the latter optimal solution (if present) we obtain a solution that is feasible for the former. From (3.39) and (3.40), the regrets for the two cases satisfy
\[
 r^\sigma(x)(x) = z^\sigma_*(x) - z^\sigma_*(x) \geq z^\sigma_*(x') - \max\{0, p_j^+\} - |p_j^-| = r^\sigma_*(x') - \max\{0, p_j^+\} + |p_j^-| \geq r^\sigma_*(x')(x') \tag{3.41}.
\]

Additionally observe that an item with \( p_j^- \leq 0 \) but \( p_j^+ > |p_j^-| \) cannot be removed from the instance, since it could be included in an optimal solution. Consider indeed the family of instances with \( n = 2, w_1 = w_2 = C, p_1^- < 0, p_1^+ > |p_1^-|, \) and \( p_2^- = p_2^+ = \varepsilon \) (where \( \varepsilon \) is a small positive value). The regrets of the three feasible solutions \( x^{(1)} = (0, 0), x^{(2)} = (1, 0) \) and \( x^{(3)} = (0, 1) \) are, respectively, \( p_1^+ - 0, \varepsilon + |p_1^-| \) and \( p_2^+ - \varepsilon \). For a sufficiently small \( \varepsilon \), the regret of \( x^{(2)} \) is smaller than the other two, i.e., the unique optimal solution includes item 1.

As already observed, evaluating the regret \( r(x) \) of a solution \( x \in X \) through Lemma 1 requires the solution to an \( \mathcal{NP} \)-hard problem. The exact and approximation algorithms introduced in the next sections will try to limit the number of such solutions through a standard approach that assumes that a feasible solution providing an upper bound \( U^r \) on the optimal regret is known. Let 01KP(\( \sigma \)) denote the instance of 01KP in which \( p_j = p_j^\sigma \) for \( j = 1, 2, \ldots, n \). Recall that \( z^\sigma(x) \) is the solution value given, for scenario \( s \), by solution \( x \) (see (3.24)).

**Standard exact algorithms.** In this section we discuss two solution approaches (a Benders-like decomposition and a branch-and-cut algorithm) that have been frequently used for the exact solution of problems of this kind. The former approach is not the classical decomposition, as the slave problem is not a linear program. However, the term is widely used in the literature, and specifically in the min-max regret literature. We thus decided to adopt it for better clarity with respect to previous similar approaches. Also note that the results presented so far in the literature on robust versions of combinatorial optimization problems do not indicate a clear winner among such approaches.

We will use model M(S) by iteratively solving instances M(R) in which only a subset, \( R \subseteq S \), of scenarios (i.e., of constraints (3.35)) is considered, and progressively adding scenarios to \( R \) until an optimal solution is found. A fundamental tool for such approaches is a method to separate violated constraints from non-violated ones.

**Separation.** Given a solution \( (\bar{x}, \tilde{\vartheta}) \) satisfying (3.36)-(3.37), we want to check if it also satisfies constraints (3.35). In order to determine if there exists a scenario \( \sigma \in S \) for which
\[
\sum_{j=1}^{n} \left( p_j^+ + (p_j^- - p_j^+) \bar{x}_j \right) \tilde{y}_j^\sigma > \tilde{\vartheta} \tag{3.41}
\]
holds, we can define, for \( j = 1, 2, \ldots, n \),
\[ p'_j = p^+_j + (p^-_j - p^+_j)\tilde{x}_j, \]
and solve the 01KP
\[
\max \sum_{j=1}^{n} p'_j y_j : \sum_{j=1}^{n} w_j y_j \leq C, y_j \in \{0, 1\} \ (j = 1, 2, \ldots, n).
\]

If the optimal solution value is greater than \( \tilde{\vartheta} \), then a violated constraint has been found.

It is interesting to observe that the above procedure does not exploit the integrality of \( \tilde{x} \), and hence the separation also holds for fractional solutions \((\tilde{x}, \tilde{\vartheta})\) satisfying (3.36) and (3.38). However, for a fractional \( \tilde{x} \), \( p'_j \) can take any value in the continuous interval \([p^-_j, p^+_j]\) so the set of scenarios extends from \( S \) to \( S_0 \).

The above separation method can be used for two classical algorithmic approaches which iteratively solve problem instances by only considering a subset of scenarios.

**Benders-like decomposition.** We tested a first approach, based on the Benders’ decomposition idea, which solves, at each iteration, a master problem \( M(R) \) defined by a relaxation of (3.34)-(3.37) in which \( S \) is replaced a by a subset \( R \subset S \) of constraints. Let \((\tilde{x}, \tilde{\vartheta})\) be the optimal solution for the current master. The slave problem (3.42) is then used to find a violated constraint, if any: if such a constraint is found, the corresponding scenario is added to \( R \). The process is iterated until a solution is found that violates no constraint, and hence is feasible. Since each solution to the master problem provides a valid lower bound on the optimal solution value, the first feasible solution encountered is optimal.

**Branch-and-cut.** The second approach we tested was a branch-and-cut algorithm. At each node of the branch-decision tree, we solve the continuous relaxation, \( \hat{M}(R) \), of the master problem above, i.e., (3.34)-(3.36) and (3.38) with \( R \subset S_0 \). If its value is not smaller than that of the incumbent solution, then the node is fathomed. Otherwise, the (possibly fractional) current solution is tested, as above, to find violated constraints (cuts) to be added to the current set \( R \). When no constraint is violated, if the current solution is integer, the incumbent is possibly updated. If instead it is fractional, a branching follows. The general framework of branch-and-cut algorithms is very flexible, and various implementations are possible.

The former approach usually requires a smaller number of 01KP solutions, but has the disadvantage of only providing a feasible solution upon completion. Although branch-and-cut tends to solve a (much) larger number of 01KPs, it can be more efficient as the produced cuts can fathom branch-decision nodes at an earlier stage hence accelerating the overall convergence. For a case such as the considered one, in which the separation is provided by a problem which (although \( \mathcal{NP} \)-hard) is relatively easy to solve in practice, the latter approach tends to be faster.

### 3.2.1 A Lagrangian-based branch-and-cut algorithm

In this section we describe the specifically tailored branch-and-cut algorithm we developed, which we denote such algorithm as FIMY. The algorithm starts by only considering a very small subset of the exponentially many constraints (3.35), i.e., problem (3.34)-(3.37) with \( S \) replaced by \( R \subset S \). At each iteration the current set \( R \) is possibly augmented. The branch-decision tree follows the strategy commonly adopted for the 01KP: after having sorted the items, at each level two child nodes are
generated by setting the next variable \(x_j\) to 1 (if the current residual capacity allows insertion of item \(j\)) and to 0. The tree is searched in a depth-first fashion by first exploring the node, if any, generated by condition \(x_j = 1\). On the basis of preliminary computational experiments, the values \((p_j^+/p_j^-)/w_j\) were used for sorting the items.

At each iteration, a first attempt to fathom the current node is performed by solving the continuous relaxation, \(\hat{M}(R)\), of the current problem \(M(R)\). If the resulting lower bound is not strong enough to fathom the node, a second attempt is performed through the Lagrangian relaxation of those constraints (3.35) that have been generated so far. Namely, we relax the current set \(R\), obtaining:

\[
L(R, \lambda) = \min \left( \vartheta - \sum_{j=1}^{n} p_j^- x_j - \sum_{\sigma \in R} \lambda_{\sigma} \left( \vartheta - \sum_{j=1}^{n} p_j^+ y_j^\sigma - \sum_{j=1}^{n} (p_j^- - p_j^+) \tilde{y}_j^\sigma x_j \right) \right)
\]

\[
= \sum_{\sigma \in R} \lambda_{\sigma} \sum_{j=1}^{n} p_j^+ y_j^\sigma + \min \left( \vartheta \left( 1 - \sum_{\sigma \in R} \lambda_{\sigma} \right) - \sum_{j=1}^{n} \tilde{p}_j x_j \right) \tag{3.43}
\]

s.t. \(\sum_{j=1}^{n} w_j x_j \leq C\) \tag{3.44}

\(x_j \in \{0, 1\} \ (j = 1, 2, \ldots, n)\). \tag{3.45}

where \(\tilde{p}_j = p_j^- - \sum_{\sigma \in R} \lambda_{\sigma} (p_j^- - p_j^+) \tilde{y}_j^\sigma\).

Solving the Lagrangian dual problem, i.e., finding the multipliers that provide the highest possible value of \(L(R, \lambda)\), could be computationally intractable. Instead, as the solution to \(\hat{M}(R)\) also provides an optimal solution to its dual, we use the optimal values of the dual variables associated with (3.35) (i.e., optimal multipliers for the continuous relaxation of \(L(R, \lambda)\)) as Lagrangian multipliers \(\lambda_{\sigma}\). Since \(\vartheta\) is not restricted in sign, the dual of \(\hat{M}(R)\) forces

\[
\sum_{\sigma \in R} \lambda_{\sigma} = 1. \tag{3.46}
\]

It follows that, for this specific (not necessarily optimal) choice of \(\lambda\), an optimal solution to \(L(R, \lambda)\) is independent of \(\vartheta\), and hence it can be obtained by solving the 01KP

\[
\max \sum_{j=1}^{n} \tilde{p}_j x_j : \sum_{j=1}^{n} w_j x_j \leq C, \ x_j \in \{0, 1\} \ (j = 1, 2, \ldots, n).
\]

When the node is not fathomed by the Lagrangian bound, we invoke the separation procedure on an optimal solution \((\hat{x}, \hat{\vartheta})\) to \(\hat{M}(R)\), in order to determine a violated constraint (if any), which is then added to \(R\). The two attempts are iterated until either no violated constraint is found, or a pre-specified maximum number of attempts has been performed. Whenever the Lagrangian bound is computed, the actual value of the integer solution it provides (i.e., its regret with respect to all scenarios) is evaluated, and the incumbent solution is possibly updated.

### 3.2.2 Computational Experiments

We performed an extensive computational evaluation of the exact algorithms. All algorithms were coded in C++, and run on a PC with a Pentium 4 at 3.2 GHz and 1 GB RAM memory, under Linux Ubuntu 11. All the approaches require the solution to 01KP sub-instances, which were produced using algorithm combo by Martello, Pisinger, and Toth, 1999 (a C code available at http://www.diku.dk/hjemmesider/ansatte/pisinger/codes.html).
3.2. The interval min-max regret knapsack problem

The ILP solutions needed by our Benders-like decomposition were obtained through Cplex 12.3 single thread. The branch-and-cut approach of Section was implemented using the Cplex callback framework. In algorithm FIMY, Cplex was only used to solve the LP relaxations.

We randomly generated nine classes of MRKP instances by defining profit intervals for nine standard classes of 01KP instances from the literature, obtained through the generator used in Martello, Pisinger, and Toth, 1999 (available at the same URL as combo), see Furini et al., 2015a for mode details. For each class and value of $\bar{R}$, we generated 27 MRKP instances through all combinations of

- number of items $n \in \{50, 60, 70\}$;
- capacity $C \in \{\lfloor 0.45W \rfloor, \lfloor 0.50W \rfloor, \lfloor 0.55W \rfloor \}$, with $W = \sum_{j=1}^{n} w_j$ (and $c$ increased by 1, if even, for classes 7 and 8);
- profit interval $[p^-_j, p^+_j]$, with $p^-_j$ u.r. in $[(1 - \delta)p_j, p_j]$, $p^+_j$ u.r. in $[p_j, (1 + \delta)p_j]$, and $\delta \in \{0.1, 0.2, 0.3\}$,

thus obtaining 486 MRKP instances.

We discuss now the computational experiments performed on the exact algorithms for the MRKP. We compared the two standard approaches, i.e., the Benders-like decomposition, denoted in the following as ‘Benders’, and B&Cut) and the proposed Lagrangian-based algorithm (FIMY) on the nine classes of instances previously described. Each algorithm had a time limit of 1 CPU hour per single instance. An entry ‘t.l.’ in the tables indicates that the algorithm reached the time limit for all six instances. For the other cases, the time limits (if any) were included in the average CPU time computation.

Benders was always dominated by B&Cut, both for what concerns CPU time and number of failures. In turn, FIMY dominated B&Cut, with few irrelevant exception mostly concerning the very easy instances of Classes 1 and 9, which were solved by all algorithms within very short CPU times. The hardest instances were clearly those in Classes 6 and 7: Benders and B&Cut could not solve a single instance over 108, while FIMY solved 42 of them to proven optimality. The remaining five classes (2, 3, 4, 5, and 8) were also very hard for Benders and B&Cut, which solved only a small portion of the instances (65 and 128, respectively, out of 270), whereas FIMY solved the majority of them (241 out of 270), frequently within short CPU times. Overall, FIMY turns out to be, by far, the most effective algorithm for the exact solution of the MRKP.

In Table 3.2 we show how the average CPU time and the total number of failures vary when increasing one of the following four parameters: number of items $n$ (first group of lines), capacity $c$ (second group), profit interval width $\delta$ (third group), and range $R$ (fourth group). For FIMY we additionally give the average number of explored nodes. Each entry refers in this case to all instances generated for the considered parameter value (162 instances for the first three groups, 243 instances for the last group). The overall average and total values (obviously the same for each group) are provided in the final line. As expected, the difficulty sharply increases when $n$ grows, while the value of $C$ has a lesser impact. The relatively most difficult capacity value is 0.5 $W$, which confirms a known property of the 01KP: instances in which about half the items are in the optimal solution tend to be more difficult than instances with solutions containing many or few items. An increase in the value of $\delta$ produces instances that are much more difficult for Benders and B&Cut, while its effect is less remarkable for FIMY. The results in the final group show that instances
with weights in a larger range are considerably easier to solve, while the 01KP usually exhibits an opposite behavior: indeed, the maximum regret value turned out to be, on average, relatively smaller for $\bar{R} = 10000$ ($11181 \approx 1.12 \bar{R}$) than for $\bar{R} = 1000$ ($1550 = 1.55 \bar{R}$), i.e., the worst scenario has, in the former case, a smaller relative deviation between the two involved 01KP solutions.
Chapter 4

Reformulation of a class of convex separable MINLP

We study solution techniques for convex separable Mixed-Integer Non-Linear Programs (MINLP) with \( n \) semi-continuous variables \( x_i \in \mathbb{R} \) for \( i \in N = \{1, \ldots, n\} \) which either assume the value 0 or lie in the interval \( \mathcal{X}_i = [x_i, \bar{x}_i] \) \((-\infty < x_i < \bar{x}_i < \infty\)). This can be expressed, introducing \( y_i \in \{0, 1\} \) for \( i \in N \), as

\[
\begin{align*}
\text{(P)} & \quad \min h(z) + \sum_{i \in N} f_i(x_i) + c_i y_i \\
& \quad Ax + By + Cz = b \\
& \quad (x, z) \in \mathcal{O} \\
& \quad \bar{x}_i y_i \leq x_i \leq \bar{x}_i y_i, \quad y_i \in [0,1]^n, \quad x_i \in \mathbb{R}^n \quad i \in N
\end{align*}
\]

We assume the functions \( f_i \) to be closed convex, one time continuously differentiable and finite in the interval \( (x_i, \bar{x}_i) \); w.l.o.g. we also assume \( f_i(0) = 0 \). In (P) we single out the linking constraints (4.2) that contain all the relationships linking the \( y_i \) variables among them and with the other variables of the problem, except those (4.4) that “define” the semi-continuous nature of the \( x_i \). The reformulation technique developed in Frangioni et al., 2011 require (4.2) to be empty; the extension developed in Frangioni, Furini, and Gentile, 2016 allows to overcome this limitation, but potentially at the cost of a worse bound quality. The aim of this paper is to deal with constraint (4.2) in a cost-effective way. For our approach to work, (4.2) must have a compatible structure with that of (4.1). Because our approach hinges on availability of dual information for the continuous relaxation, we assume that the function \( h(\cdot) \) in the “other variables \( z' \) and the “other constraints (4.3)” are convex, i.e., (P) is a convex MINLP. Actually, in many applications everything but (4.1) is linear. It will be sometimes expedient to refer to (4.3)–(4.4) as “\((x,y,z) \in \mathcal{P}'\)”, and to \( \mathcal{P} \) as the set obtained by \( \mathcal{P} \) relaxing integrality constraints on \( z \) and \( x \), if any.

Often, the most pressing issue in solving (P) is to derive tight lower bounds on its optimal value \( \nu(P) \), which is typically done by solving its (convex) continuous relaxation \( (\mathcal{P}) \) (we denote by \( \nu(X) \) and \( (X) \), respectively, the optimal value and the continuous relaxation of any problem \( X \)). However, often \( \nu(\mathcal{P}) \ll \nu(P) \), making the solution approaches inefficient. The presence of semi-continuous variables has been exploited to propose reformulations \( (\mathcal{P}') \) of (P) such that \( \nu(P) = \nu(\mathcal{P}') \geq \nu(\mathcal{P}) \gg \nu(P) \). This starts from considering (4.1) as \( h(z) + \sum_{i \in N} f_i(x_i, y_i) \), where \( f_i(x_i, y_i) = f_i(x_i) + c_i \) if \( y_i = 1 \) and \( \bar{x}_i \leq x_i \leq \bar{x}_i, f_i(0,0) = 0 \), and \( f_i(x_i, y_i) = \infty \) otherwise. The convex envelope of \( f_i(x_i, y_i) \) is known Frangioni and Gentile, 2006 to be \( \tilde{f}_i(x_i, y_i) = y_if_i(x_i/y_i) + c_i y_i \)—using the perspective function of \( f_i \)—which yields the Perspective Reformulation of (P)
(PR) \[
\min \left\{ h(z) + \sum_{i \in N} \tilde{f}_i(x_i, y_i) : \ (4.2), \ (x, y, z) \in \mathcal{P}, \ (4.5) \right\}.
\]

As \( f_i \) is convex, \( \tilde{f}_i \) is convex for \( y_i \geq 0 \); since \( x_i = 0 \) if \( y_i = 0 \), \( \tilde{f}_i \) can be extended by continuity assuming \( 0 f_i(0/0) = 0 \). Hence, (PR) is a convex MINLP if (P) is. Its continuous relaxation (PR)—the Perspective Relaxation of (P)—usually has \( \nu(\text{PR}) \gg \nu(\text{P}) \), making (PR) a more convenient formulation Frangioni, Gentile, and Lacalandra, 2009; Günlük and Linderoth, 2008. If \( f_i \) is SOCP-representable then so is \( \tilde{f}_i \), hence the PR of a Mixed-Integer Second-Order Cone Program (MI-SOCP) is still a MI-SOCP. Thus, (PR) is not necessarily more complex to solve—and, sometimes, even less so Frangioni, Galli, and Scutellà, 2015—than (P). Alternatively, one can consider a Semi-Infinite MINLP reformulation of (PR) where Perspective Cuts Frangioni and Gentile, 2006—linear outer approximations of the epigraph of \( \tilde{f}_i \)—are dynamically added. This is often the best approach Frangioni and Gentile, 2009, in particular for “general” (P) where no other structure is available. It is appropriate to remark that the (PR) approach also applies if the \( x_i \) are vectors such that \( y_i = 0 \implies x_i = 0 \) and \( y_i = 1 \implies x_i \in \mathcal{X}_i \), with \( \mathcal{X}_i \) a polytope; yet, here, as in Frangioni, Furini, and Gentile, 2016; Frangioni et al., 2011, each \( x_i \) must be a single variable.

While (PR) provides a better bound, it is also usually more time consuming to solve than (P) because \( \tilde{f}_i \) is “more complex” than \( f_i \). This trade-off is nontrivial, in particular if \( f_i \) is “simple”. For instance, if \( f_i \) is quadratic and everything else is linear, (P) is a Mixed-Integer Quadratic Program (MIQP) whereas (PR) is a MI-SOCP, hence, (P)—a QP—can be significantly cheaper to solve than (PR)—a SOCP. The Projected PR (P2R) idea underpinning the approach studied here was indeed proposed in Frangioni et al., 2011 for the quadratic case, and \( \bar{x}_i \geq 0 \). It was then extended in Frangioni, Furini, and Gentile, 2016 to a more general class of functions, and allowing \( \bar{x}_i < 0 \). However, \( \bar{x}_i < 0 < x_i \) renders some of the arguments significantly more complex, hence for the sake of simplicity we will only present here the case where \( \bar{x}_i \geq 0 \); it will be plain to see that the arguments immediately extend to the more general one. The P2R idea is to analyze \( f_i \) as a function of \( x_i \) only, i.e., projecting away \( y_i \): under appropriate assumptions, and if there are no linking constraints (4.2), this turns out to be a piecewise-convex functions with a “small” number of pieces, that can be characterized by just looking at the data of (P) (cf. (4.7)). Hence, (PR) can be reformulated in terms of piecewise-convex objective functions, which makes it easier to solve, especially when \( \mathcal{O} \) has some valuable structure (e.g., flow or knapsack) Frangioni et al., 2011. However, in several applications (4.2) are indeed present Frangioni and Gentile, 2006; Frangioni, Gentile, and Lacalandra, 2009; Frangioni, Furini, and Gentile, 2016; Frangioni, Galli, and Stea, 2015; Zheng, Sun, and Li, 2014. Furthermore, since the binary variables \( y_i \) are removed from the formulation, branching has to be done “indirectly”, which rules out using off-the-shelf solvers. To overcome these two limitations, in Frangioni, Furini, and Gentile, 2016 the Approximated P2R (AP2R) reformulation has been proposed whereby the \( y_i \), after having been eliminated, are re-introduced in the formulation in order to encode the piecewise nature of \( f_i \). This is possible even if (4.2) are present, and it has the advantage that (AP2R) is still a MIQP if (P) is. However, \( \nu(\text{AP2R}) < \nu(\text{PR}) \) may, and does, happen when linking constraints (4.2) are present, whence the “Approximate” moniker. This is still advantageous in some cases, but it may happen that the weaker bounds outweigh the faster solution time, making the approach not competitive with more straightforward implementations of the PR Frangioni, Furini, and Gentile, 2016.

The aim of this section if to show how to improve the AP2R by presenting a simple and effective way to ensure that \( \nu(\text{AP2R}) = \nu(\text{PR}) \) even if (4.2) are present,
A quick overview of $\text{AP}^2\text{R}$. We now quickly summarize the analysis in Frangioni, Furini, and Gentile, 2016, albeit limited to the case $x_i \geq 0$, in order to prepare the ground for the new extension. We focus on the basic problem corresponding to one pair $(x_i, y_i)$

$$
\begin{align*}
(P_i) \quad \min \{ \; f_i(x_i) + c_i y_i : \; x_i y_i \leq x_i \leq \bar{x}_i y_i, \; y_i \in \{0, 1\} \; \}.
\end{align*}
$$

The analysis hinges on considering the $(\text{PR})$ of $(P_i)$ rewritten as

$$
\begin{align*}
(\text{PR}_i) \quad \min \{ \; p_i(x_i) = \min \{ \; \tilde{f}_i(x_i, y_i) : \; x_i y_i \leq x_i \leq \bar{x}_i y_i, \; y_i \in \{0, 1\} \; : \; x_i \in [0, \bar{x}_i] \; \}.
\end{align*}
$$

i.e., first minimizing $\tilde{f}_i(x_i, y_i)$ with respect to $y_i$, and then minimizing the resulting function $p_i(x_i)$ with respect to $x_i$. The function $p_i(x_i)$ is convex, and can be characterized by studying the optimal solution $y_i^*(x_i)$ of the inner problem in $(\text{PR}_i)$. Differentiability of $f_i$ implies that $y_i^*(x_i)$ is strictly related to the solutions (if any) of the first-order optimality conditions

$$
c_i + f_i(x_i/y_i) - f_i'(x_i/y_i)x_i/y_i = 0 \quad (4.6).
$$

The approach of Frangioni, Furini, and Gentile, 2016 requires $(4.6)$ to only have (at most) one solution, whose dependency on $y_i$ is “easy”:

**Assumption 1.** $(4.6)$ has at most one solution for $x_i \geq 0$, which has the form $\bar{y}_i(x_i) = g_i x_i$ where $g_i \geq 0$ is a constant that can be determined by the data of the problem

For instance, for the quadratic case $f_i(x_i) = a_i x_i^2 + b_i x_i$, the first-order optimality condition $(4.6)$ is $c_i - a_i x_i^2 / y_i^3 = 0$, whence $\bar{y}_i(x_i) = |x_i| \sqrt{a_i}/c_i$ if $c_i > 0$, and there is no solution otherwise. Assumption 1 is satisfied by a surprisingly large set of functions, and a more general version can be stated for the case $x_i < 0$ Frangioni, Furini, and Gentile, 2016. If $(4.6)$ has a solution then $y_i^*(x_i)$ can be found by projecting $\bar{y}_i(x_i)$ over the interval $[x_i/\bar{x}_i, \min\{1, x_i/\bar{x}_i\}]$; if no solution exists then $y_i^*(x_i)$ is in one of the two extremes. In all cases one can then write $p_i(x_i) = \tilde{f}_i(x_i, y_i^*(x_i))$. All this gives that there exists some $\bar{x}_i \leq \hat{x}_i \leq \bar{x}_i$ such that

$$
p_i(x_i) = \{ (f_i(\hat{x}_i)/\hat{x}_i + c_i/\hat{x}_i)x_i \; : \; 0 \leq x_i \leq \hat{x}_i, \; f_i(x_i) + c_i \; : \; \hat{x}_i \leq x_i \leq \bar{x}_i \} \quad (4.7).
$$

Thus, $p_i(x_i)$ is piecewise-convex with at most two pieces (although these become four if $x_i < 0$), one of which is linear and the other is the original objective function. The crucial breakpoint $\hat{x}_i$ can be determined a-priori: in particular, $\hat{x}_i = 1/g_i$ if $(4.6)$ has a solution and $1/g_i \in \mathcal{X}_i$, and $\hat{x}_i \in \{x_i, \bar{x}_i\}$ otherwise Frangioni, Furini, and Gentile, 2016. For a numerical illustration, the quadratic case

$$
(P_1) \quad \min \{ 2x_i^2 + 8y_1 : \; y_1 \leq x_1 \leq 10y_1, \; y_1 \in \{0, 1\} \}
$$

has $\bar{y}_1(x_1) = x_1 \sqrt{a_1/c_1} = x_1/2$, thus $g_1 = 1/2$, and therefore $1/g_1 = 2 \in \mathcal{X}_1 = [1, 10]$: hence,

$$
p_1(x_1) = \{ 8x_1 \; : \; 0 \leq x_1 \leq 2, \; 2x_1^2 + 8 \; : \; 2 \leq x_1 \leq 10 \} \quad (4.8).
$$

Writing $(4.7)$ as the objective function is typically done with the “variable splitting” approach Frangioni et al., 2011, whereby two new variables $0 \leq x_i/\hat{x}_i \leq \hat{x}_i$ and $0 \leq \hat{x}_i$.
$x''_i \leq \bar{x}_i - \bar{x}_i$ are introduced such that $x_i = x'_i + x''_i$ (although a different form is sometimes preferable Frangioni, Galli, and Stea, 2015): $x'_i$ gets the linear cost, while $x''_i$ has cost $f_i(x''_i)$. This yields the Projected Perspective Reformulation (P$^2$R), having the same form as (P)—a MIQP if (P) was one—with at most (and, often, less than) twice as many variables. The corresponding (P$^2$R) might be more efficient to solve, especially if (P) has some structure that allows application of specialized approaches Frangioni et al., 2011. However, removing the $y_i$ variables from the formulation prevents from using off-the-shelf software to solve (P$^2$R). This is why in Frangioni, Furini, and Gentile, 2016 it was proposed to “lift back” (4.7) in the original $(x_i, y_i)$ space by defining the problem

$$(LP_i(x_i)) = \min \left\{ y_i p_i(\bar{x}_i) + f_i(x''_i + \bar{x}_i) + c_i - p_i(\bar{x}_i) : (x_i - \bar{x}_i)y_i \leq x''_i \leq (\bar{x}_i - \bar{x}_i)y_i, x_i = \bar{x}_i y_i + x''_i, y_i \in [0, 1] \right\}.$$ 

It can be proven Frangioni, Furini, and Gentile, 2016 that $\nu(LP_i(x_i)) = p_i(x_i)$ for each feasible $x_i$; therefore,

$$(AP^2R_i) = \min \left\{ y_i p_i(\bar{x}_i) + f_i(x''_i + \bar{x}_i) + c_i - p_i(\bar{x}_i) : (x_i - \bar{x}_i)y_i \leq x''_i \leq (\bar{x}_i - \bar{x}_i)y_i, x_i = \bar{x}_i y_i + x''_i, y_i \in (0, 1) \right\}$$

is a reformulation of (P$_i$) and $\nu(AP^2R_i) = \nu(PR_i)$, which implies that, typically, $\nu(AP^2R_i) \gg \nu(P_i)$. For illustration, consider (P$_i$): plugging (4.8) into (AP$^2$R$_i$) yields

$$(AP^2R_1) = \min \left\{ 2(x'_1)^2 + 8x''_1 + 16y_1 : -y_1 \leq x''_1 \leq 8y_1, x_1 = 2y_1 + x''_1, y_1 \in (0, 1) \right\}.$$ 

It can be verified that $\nu(AP^2R_1) \geq \nu(P_1)$ for any fixed $x_1$. For instance, for $x_1 = 2$ the optimal solution to (P$_1$) is $y_1 = 1/5$, yielding $\nu(P_1) = 9 + 3/5$, while the optimal solution to (AP$^2$R$_1$) is $(y_1, x''_1) = (1, 0)$, yielding $\nu(AP^2R_1) = 16$, i.e., the same estimate as (PR$_1$) for $x_1 = 2$. In fact,

$$\nu(PR_1) = \min \left\{ 2x''_1/y_1 + 8y_1 : y_1 \leq x_1 \leq 10y_1, y_1 \in (0, 1) \right\} = 16$$

since $\min\{8y_1 + 8/y_1 : y_1 \in [1/5, 1]\}$ has optimal solution $y_1 = 1$.

We will denote by (AP$^2$R) the reformulation of (P) where (AP$^2$R$_i$) is separately applied to each $x_i$ for $i \in N$. If there are no linking constraints (4.2), then obviously $\nu(AP^2R) = \nu(PR)$. Otherwise the reformulation is still possible, but (PR$_i$) is a relaxation of the true projection problem, which, besides on $x_i$, also depends on all the other variables that $y_i$ is linked with. Hence, $\nu(AP^2R) < \nu(PR)$ can happen, and it does in practice. For illustration consider the problem

$$(P_{12}) = \min \left\{ 2x'_1 + 2x'_2 + 8y_1 + 8y_2 : y_1 \leq x_1 \leq 10y_1, y_2 \leq x_2 \leq 10y_2 \right\}$$

(4.9)

$$y_1 \in \{0, 1\}, \quad y_1 + y_2 = 1, \quad y_2 \in \{0, 1\}, \quad x_1 + x_2 = 8$$

(4.10)

obtained by “duplicating” (P$_i$) and adding the linking constraint $y_1 + y_2 = 1$. The optimal solution of (P$_{12}$) is $x_1 = x_2 = 4, y_1 = y_2 = 1/2$, yielding $\nu(P_{12}) = 72 \ll \nu(P_{12}) = 136$, the latter obtained by setting $x_1 = 8, y_1 = 1, x_2 = y_2 = 0$ (or the symmetric solution). The (PR), obtained by replacing (4.9) with

$$\min \left\{ 2x'_1/y_1 + 2x''_2/y_2 + 8y_1 + 8y_2 : y_1 \leq x_1 \leq 10y_1, y_2 \leq x_2 \leq 10y_2 \right\}$$

has the same optimal solution as (P$_{12}$); however, that same solution yields the much stronger (in fact, exact) bound of 136. Instead, for the (AP$^2$R)
(AP^2R_{12}) \min 2(x''_1)^2 + 2(x''_2)^2 + 8x''_1 + 8x''_2 + 16y_1 + 16y_2 \\
(4.11) \text{, } -y_1 \leq x''_1 \leq 8y_1 \text{, } -y_2 \leq x''_2 \leq 8y_2 \text{, } x_1 = 2y_1 + x''_1 \text{, } x_2 = 2y_2 + x''_2 

(cf. (AP^2R_1)) the optimal solution of (AP^2R_{12}) is \( x_1 = x_2 = 4 \), \( y_1 = y_2 = 1/2 \), \( x''_1 = x''_2 = 3 \), yielding \( \nu(P_{12}) = 72 < \nu(AP^2R_{12}) = 100 < \nu(PR_{12}) = 136 \). In the next section we modify the (AP^2R) to increase its lower bound, avoiding the bound disadvantage with the (PR)—at least at the root node—while retaining the simpler (hence, cheaper) model shape.

4.1 Improving AP^2R using dual information

The idea is to reformulate (P) to include information about the linking constraints (4.2) in the objective function (4.1), so that it can be “processed” by the AP^2R. This hinges on the availability of dual information, and hence mainly concerns the continuous relaxations. The Lagrangian relaxation of (P) w.r.t. (4.2)

\[(P^\lambda) \min \{ h(z) + \sum_{i \in N} f_i(x_i) + c_i y_i + \lambda (Ax + By + Cz - b) : (x, y, z) \in \mathcal{P} \} \]

has an objective function that is still separable in the \( x_i \)

\[h(z) + \lambda Cz + \sum_{i \in N} (f_i(x_i) + \lambda A^i x_i + (c_i + \lambda B^i)y_i) - \lambda b \text{ . (4.12)}\]

Hence one can apply the PR to (P^\lambda), which—since the PR does not change linear functions—yields

\[(PR^\lambda) \phi(\lambda) = \min \{ h(z) + \lambda Cz + \sum_{i \in N} (y_i f_i(x_i/y_i) + \lambda A^i x_i + (c_i + \lambda B^i)y_i) : (x, y, z) \in \mathcal{P} \} - \lambda b \text{ . (4.12)}\]

We will assume that the corresponding Lagrangian dual satisfies \( \max_{\lambda} \{ \phi(\lambda) \} = \nu(PR) \), and in particular that an optimal dual solution \( \lambda^* \) is available that satisfies \( \phi(\lambda^*) = \nu(PR) \). This requires some conditions that are typically satisfied in most applications (e.g., all the constraints are linear or some appropriate Slater-type condition holds), so that we can expect that \( \lambda^* \) is generated by any approach used to solve (PR). Note that any inexact solution of the Lagrangian dual could be used as well, as long as it (significantly) improves the bound. Also, (4.12) clearly satisfies Assumption 1 if \( f_i \) does: (4.6) for \( f_i(x_i) + \lambda A^i x_i \) only differs from (4.6) for \( f_i \) for the constant \( \lambda A^i \). Hence, one can form (AP^2R^\lambda), the AP^2R of (PR^\lambda), for any value of \( \lambda \). Taking \( \lambda = \lambda^* \) yields our main result:

**Theorem 2.** Assume that \( f_i \) satisfies Assumption 1 and that \( \lambda^* \) is available; for the reformulation of (P)

\[(P^+) \min \{ h(z) + \lambda^* Cz + \sum_{i \in N} (f_i(x_i) + \lambda^* A^i x_i + (c_i + \lambda^* B^i)y_i) : (4.2), (x, y, z) \in \mathcal{P} \} - \lambda^* b \]

denote by (PR^+) its PR and by (AP^2R^+) its AP^2R. Then, \( \nu(AP^2R^+) = \nu(PR) = \nu(PR^+) \).

**Proof.** First of all, (P^+) is a valid reformulation of (P) since it contains (4.2), the Lagrangian term \( \lambda^* (Ax + By + Cz - b) \) is always null at feasible points, hence both the feasible regions and the objective functions coincide. Also, as discussed for (AP^2R^\lambda) the objective function of (P^+) satisfies Assumption 1 since the \( f_i \) do, and therefore (AP^2R^+) can be formed and \( \nu(P) = \nu(P^+) = \nu(AP^2R^+) \) holds. However, note that
(4.1) and (4.12) are in general different: in fact, the equivalence between the two optimal values only holds when taking into account the constant $-\lambda^* b$. Now, clearly $\nu(\text{PR}) = \nu(\text{PR}^{\lambda^*}) = \nu(\text{PR}^+)$ due to the choice of $\lambda^*$. Note that $\text{(PR}^{\lambda^*})$ is a relaxation of $(\text{PR}^+)$, which in itself would give only $\nu(\text{PR}^{\lambda^*}) \leq \nu(\text{PR}^+)$; however the relaxed constraints are precisely (4.2), those of which $\lambda^*$ are the optimal multipliers, and therefore equality must hold. Now, since $(\text{PR}^{\lambda^*})$ has no linking constraints (4.2), $\nu(\text{AP}^2R^{\lambda^*}) = \nu(\text{PR}^+)$ for each $\lambda$, and therefore in particular for $\lambda = \lambda^*$. Hence, $\nu(\text{PR}) = \nu(\text{PR}^{\lambda^*}) = \nu(\text{AP}^2R^{\lambda^*}) \leq \nu(\text{AP}^2R^+)$, because, again, $(\text{AP}^2R^{\lambda^*})$ is the relaxation of $(\text{AP}^2R^+)$ w.r.t. (4.2). But on the other hand $\nu(\text{PR}^+) \geq \nu(\text{AP}^2R^+)$, as the bound of $\text{AP}^2R$ is always at most as good as that the PR, and therefore the thesis is proven. \[\square\]

To illustrate Theorem 2, consider again $(P_{12})$. The optimal dual multiplier of the linking constraint $y_1 + y_2 = 1$ in $(\text{PR}_{12})$ can be seen to be $\lambda^* = 120$. Hence, $(P_{12}+) \quad \min \left\{ 2x_1^2 + 2x_2^2 + 128y_1 + 128y_2 : (4.10), (4.11) \right\} = 120 .

In its $\text{AP}^2R$, $c_i = 128$ gives $g_i = \sqrt{a_i/c_i} = \sqrt{2/128} = 1/8$, i.e., $\hat{x}_i = 8$ for $i = 1, 2$. Hence, $p_i(x_i) = y_ip(\hat{x}_i) + f_i(x_i') + c_i - p_i(\hat{x}_i) = 256y_i + 2(x_i')^2 + 32x_i' \implies$

$(\text{AP}^2R_{12}+) \quad \min 2(x_1')^2 + 2(x_2')^2 + 32x_1'' + 32x_2'' + 256y_1 + 256y_2$

$\quad (4.11), -7y_1 \leq x_1'' \leq 2y_1, -7y_2 \leq x_2'' \leq 2y_2$,

$x_1 = 8y_1 + x_1'', x_2 = 8y_2 + x_2''$.

The optimal solution of $(\text{AP}^2R_{12}+) = x_1 = x_2 = 4, y_1 = y_2 = 1/2, x_1'' = x_2'' = 0$, yielding an objective function value of 256: counting the constant $-\lambda^* b = -120$, this finally gives $\nu(\text{AP}^2R_{12}+) = 136$: (much) better than $\nu(\text{AP}^2R_{12}) = 100$, and in fact precisely equal to $\nu(\text{PR}_{12})$ as predicted.

We end this section by remarking that the assumption that (4.2) is an equality constraint is not crucial. Inequality linking constraints $Ax + By + Cz \leq b$ can be transformed into equalities by the addition of slack variables: $Ax + By + Cz + s = b, s \geq 0$. Note that this has to be done in (P), so that after the reformulation they will have cost $\lambda^* s$ in the objective function, i.e., they no longer will be slack variables. In principle the constraints could also be nonlinear in $x$ and $z$ (linearity in $y$ can always be assumed without loss of generality), provided that $A(x) = \sum_{i \in N} A^i(x_i)$ with each $A^i(\cdot)$ convex, Assumption 1 holds for $f_i(x_i) + \lambda^* A^i(x_i)$, and $C(z)$ is convex. However, in order for (4.2) to be convex they necessarily had to be inequalities. This means that the optimal dual multipliers $\lambda^*$ would be non-negative, retaining convexity of $\lambda^* A^i(x_i)$, but the constraints would have to be turned into nonlinear inequalities, that can never be convex.

### 4.2 Computational Experiments

In this section we report results of computational tests of the proposed approach for the Mean-Variance cardinality-constrained portfolio optimization problem on $n$ risky assets

$$\text{(MV)} \quad \min \left\{ x^TQx : \sum_{i \in N} x_i = 1, \sum_{i \in N} \mu_i x_i \geq \rho, \sum_{i \in N} y_i \leq k, (4.4), (4.5) \right\} ,$$

where $\mu$ is the vector of expected unitary returns, $\rho$ is the prescribed total return, $Q$ is the variance-covariance matrix, and $k \leq n$ is the maximum number of purchasable assets. Without the cardinality constraint ($k = n$), (MV) is well suited for $\text{AP}^2R$: the
bound is the same as that of the PR, and the computation time per node is greatly reduced with respect to the Perspective Cut (P/C) technique. While AP^2R is competitive also for \( k \ll n \), it becomes less so as the quality of the bound significantly deteriorates Frangioni, Furini, and Gentile, 2016. Hence, (MV) is a promising application for AP^2R+. Since (MV) is a non-separable MIQP, a diagonal matrix \( D \) has to be determined such that \( Q - D \) is positive semidefinite: the PR technique is applied to \( \sum_{i \in N} D_{ii} x_i^2 \), leaving the remaining part \( x^T (Q - D) x \) untouched. Choosing \( D \) is nontrivial: one can use e.g., a “small” SDP as advocated in Frangioni and Gentile, 2007, or a “large” SDP as proposed in Zheng, Sun, and Li, 2014. We denote these two by \( D_s \) and \( D_l \). Although \( D_l \) provides a better root node bound, it is not necessarily the best choice throughout the enumeration tree: sometimes a convex combination between the two, denoted by \( D_c \), works better Zheng, Sun, and Li, 2014.

For our tests we used the 90 randomly-generated instances, 30 for each value of \( n \in \{200, 300, 400\} \), already employed in Frangioni, Furini, and Gentile, 2016; Frangioni and Gentile, 2006; Frangioni and Gentile, 2007; Frangioni and Gentile, 2009; Zheng, Sun, and Li, 2014 to which the interested reader is referred for details. Here we only remark that “+” instances are strongly diagonally dominant, “0” ones are weakly diagonally dominant, and “−” ones are not diagonally dominant; the less diagonally dominant, the harder an instance is. We have set \( k = 10 \), as in Frangioni, Furini, and Gentile, 2016; Zheng, Sun, and Li, 2014; this is a “tight” value, since the maximum number of assets that the model can choose, due to the lower limits \( x_i > 0 \), without the cardinality constraint is \( \approx 20 \) for all \( n \). The (MV) instances and the diagonals used in the experiments are available at http://www.di.unipi.it/optimize/Data/MV.html.

The experiments have been performed on a computer with a 3.40 Ghz 8-core Intel Core i7-3770 processor and 16Gb RAM, running a 64 bits Linux operating system. All the codes were compiled with g++ (version 4.8.4) using -O3 as optimization option. We have tested AP^2R+ vs. AP^2R using Cplex 12.7, single-threaded, with all default parameters (save for one explicitly described below, and only for the tests with the \( D_l \) diagonal). All the formulations have been given in the natural form, i.e., as a MIQP with linear constraints and convex quadratic nonseparable objective function. We have obtained the (PR) root node bound with the P/C technique, implemented through callbacks, which is typically the best choice when AP^2R is not available Frangioni and Gentile, 2007; hence, for completeness we also report results for the full B&C using P/C. Since we are not interested in comparing the cost/effectiveness of the different diagonal choices, this having been done in Zheng, Sun, and Li, 2014, we don’t report detailed SDP times beyond saying that the “small” SDP requires on average about 0.2, 0.7, and 1.6 seconds while the “large” SDP requires 9, 21 and 47 seconds, respectively for \( n = 200, 300 \) and 400 (with little variance for the same \( n \)). We also don’t report comparisons with either not using PR techniques at all, or using SOCP formulations, because they are well-known to be from dramatically Frangioni and Gentile, 2006; Frangioni and Gentile, 2007 to significantly Frangioni and Gentile, 2009 less effective than the ones employed here.

The results are reported in Table 4.1 for the two diagonal \( D_s \). In the table we report the (average) total B&C time and root time when using P/C. For AP^2R and AP^2R+ we report the (average) total number of B&C nodes, total B&C time, root node time and root gap (in percentage). As predicted by Theorem 2 the root node gap of AP^2R+ and P/C was identical, which is why we do not report it for P/C. The total time of AP^2R+ already includes the P/C root time, since it is needed to compute \( \lambda^* \) prior to performing the reformulation, and therefore starting the AP^2R+ B&C.
Table 4.1 shows that $\text{AP}^2\text{R}^+$ is highly competitive with $\text{AP}^2\text{R}$, and a fortiori with P/C, reducing total time by up to an order of magnitude. While the exact ratio depends on $n$, the type of instance and the diagonal, the trend is clear: the improvement is due to the much reduced number of nodes, itself a consequence of the much improved bound, while the computing time per node remains the same.
Chapter 5

Conclusion

This HDR document is composed by 4 chapters. The first Section 1 is the introduction and it details my professional research activities of the last years, i.e., my period of research after my PhD at LAMSADE, the computed science department of Paris-Dauphine University. In this first section, I presented all my main research results gathered in several interconnected topics: Automatic Dantzig-Wolfe Decomposition, Two-Dimensional Cutting Problems, Generalizations of the Vertex Coloring and Knapsack Problems, Non-Linear Programming Problems and Real-world Applications. This chapter gave me the opportunity to present and organize all my articles grouping them by thematic areas. In addition, I tried to point out the backbone principle of my research which has its roots in the golden-rule principle of “divide and rule”. This concept refers to a strategy aiming at breaking up complex problems into a series of smaller and easier problems which can be then solved in cascade. Translating this idea into Mathematical Optimization, it means to Decompose and Reformulate Mixed Integer Linear and Nonlinear Programs. Often, hard Combinatorial Optimization Problems cannot be effectively solved by directly attacking them but, once these problems are decomposed and reformulated, they become easier to handle. In all my articles the principle of “divide and rule” has been the inspiration of exact algorithms which managed to outperform standard and classical approaches. The ultimate goal of all my research was then to deliver efficient exact algorithms which are able to solve large real-world instances. For this reason, I also devoted a large part of my research to real-world applications in collaboration with leader companies in different industrial sectors.

Presenting in details all my articles was not possible for reason of space, I decided instead to present 3 topics, pointing out some of the results I managed to achieve. In particular, I decided to present in Section 2 the reformulation of Coloring Problems on Graphs, in Section 3 the reformulation of Knapsack Problems and in Section 4 the reformulation of a class of convex separable MINLP with semi-continuous variables. These sections present very recent articles which have been published in 2017 or that are currently under revision.

Writing this HDR document has been a very challenging task, which required me an in depth analysis of my research activities. But, at the same time, it allowed me to develop a global critical vision of my work. In this documents I tried to present the most successful decomposition techniques that I have used in order to achieve my principal results. Hopefully, it might also represent a source of inspiration for other researches seeking to effectively solve other related hard combinatorial optimization problems.
Decomposition techniques have played and are playing an important role in many different fields of mixed-integer linear and non-linear optimization, multi objective optimization, optimization under uncertainty, bilevel optimization, etc. Despite the tremendous amount of research on these topics, the mathematical optimization community is constantly challenged by new theoretical aspects and real world applications that require the development of new advanced tools. I strongly believe that these research themes will play a central role also in the future, particularly for hard combinatorial problem and massive data sets. I am glad to be a part of this active and stimulating community of researchers.
Bibliography


Pferschy, U. and R. Scatamacchia ((forthcoming)). “Improved dynamic programming and approximation results for the knapsack problem with setups”. In: International Transactions in Operational Research.
