The Multiplicative Weights Update Algorithm for Mixed Integer NonLinear Programming: Theory, Applications, and Limitations

Luca Mencarelli

NNT : 2017SACLSX099

Thèse de doctorat de l'Université Paris-Saclay préparée à l'École Polytechnique

Ecole doctorale n°580
Sciences et Technologies de l'Information et de la Communication (STIC)
Spécialité de doctorat : Informatique

par

M. Luca Mencarelli

L'Algorithme Multiplicative Weights Update pour la Programmation non linéaire en nombres entiers: Théorie, Applications et Limites

Thèse présentée et soutenue à Palaiseau, le 4 Décembre 2017.

Composition du Jury :

Mme.	Sourour Elloumi	Professeur ENSTA ParisTech	(Présidente du jury)
Mme.	Ivana Ljubic	Professeur ESSEC Business School	(Rapporteur)
M.	Lucas Létocart	Maître de conférences HDR LIPN, Université Paris 13	(Rapporteur)
M.	Emiliano Traversi	Maître de conférences LIPN, Université Paris 13	(Examinateur)
Mme.	Claudia D'Ambrosio	Chargée de recherche CNRS, École Polytechnique	(Co-Directrice de thèse)
M.	Leo Liberti	Directeur de recherche CNRS, École Polytechnique	(Directeur de thèse)

This PhD thesis has been realized with the LaTeX distribution on Mac OS X using the ClassicThesis style by André Miede, inspired by the book "The Elements of Typographic Style" [34] by Robert Bringhurst. The graphic design of this thesis can be reproduced by compiling the example TesiClassica on http://www.lorenzopantieri.net/LaTeX.html.

Apprendre sans désir, c'est désapprendre à désirer.

— Raoul Vaneigem

To my extraordinary family.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

This thesis presents a new algorithm for Mixed Integer NonLinear Programming, inspired by the Multiplicative Weights Update framework and relying on a new class of reformulations, called the pointwise reformulations. The thesis is divided in three main parts: a foreword consisting in Chapter 1, a theoretical foundation of the new algorithm in Chapter 2, and the application of this new methodology to two real-world optimization problems, namely the Mean-Variance Portfolio Selection in Chapter 3, and the Multiple NonLinear Knapsack Problem in Chapter 4.

Mixed Integer NonLinear Programming is a hard and fascinating topic in Mathematical Optimization both from a theoretical and a computational viewpoint. These problems are characterized by nonlinear objective function and constraints, and continuous and integer decision variables. Many real-world problems can be cast this general scheme and, usually, are quite challenging in terms of efficiency and solution accuracy with respect to the solving procedures. Another very important tool in Mathematical Optimization is represented by formulations and reformulations: in particular, we introduce a new family of reformulations, namely pointwise reformulations, depending on a given parameter, which are easier to solve than the original formulation. A remarkable characteristic we look for in the pointwise reformulation is exactness, i.e., the existence of a given value of the parameter such that a global optimum of the original problem is also a global optimum for the reformulation. The basic idea to heuristically solve Mixed Integer NonLinear Problems consists in finding the optimum of the (easier) exact pointwise reformulation, which immediately yields the corresponding global optimum of the original problem. We employ the Multiplicative Weights Update algorithm in order to identify the correct value of the parameter for the pointwise reformulation.

In Chapter 1 we give an overview of the mathematical concepts and entities we use in the rest of the thesis. Chapter 2 is devoted to illustrate the general scheme of the new algorithm, the Multiplicative Weights Update for Mixed Integer NonLinear Programming, and its main theoretical properties. Moreover, in this chapter we define several automatic building procedures to determine the pointwise reformulation of a given Mixed Integer NonLinear Problem for specific, but broad, classes of optimization problems.

In the rest of the thesis we deal with two real-world challenging optimization problems: Mean-Variance Portfolio Selection and Multiple NonLinear Knapsack Problems. In Chapter 3 we give a survey on the models, formulations and reformulations, and exact methods for the single-objective single-

period Mean-Variance Portfolio Selection problem. Among all the versions for the portfolio problems proposed in the specialized literature, we choose the most addressed class, namely cardinality constrained portfolio selection with semi-continuous variables. We consider also a possibly non-convex non-concave transaction cost function, with the only hypothesis of separability, which is quite natural in the context of financial markets. The Multiplicative Weights Update for Mixed Integer NonLinear Problems behaves sufficiently better than the benchmarks with respect to the quality of the solution produced and the number of assets which compose the optimal portfolios. In general, in fact, minimizing the number of assets in the optimal portfolio is a second goal for the optimal selection procedures.

Then, in Chapter 4 we consider the Multiple NonLinear Knapsack Problem, addressed here for the first time in its entirely. We adapt the Multiplicative Weights Updated framework to this problem, proposing a new pointwise reformulation. Unfortunately, extensive computational experiments show this algorithmic approach is not well suitable to solve challenging instances of this knapsack problem. Hence, we illustrate a different heuristic method based on the discretization of the solution space and on the surrogate relaxation. The method consists of three phases: we propose a constructive greedy procedure, and two procedures for the feasibility recovering of the surrogate solution. A local search post-procedure is also implemented in order to improve the overall quality of the solution produced by the heuristics. Computational experiments indicate that this method prevails over the benchmarks both in terms of quality of the solution and of total computational elapsed time.

RÉSUMÉ

L'objectif de cette thèse consiste à présenter un nouvel algorithme pour la programmation non linéaire en nombres entiers, inspirée par la méthode Multiplicative Weights Update et qui compte sur une nouvelle classe de reformulations, appelées les reformulations ponctuelles. La thèse est divisée en trois parties principales: une introduction composée par le Chapitre 1, une définition théorique du nouvel algorithme dans le Chapitre 2 et l'application de cette nouvelle méthodologie à deux problèmes concrets d'optimisation, tels que la sélection optimale du portefeuille avec le critère moyenne-variance dans le Chapitre 3 et le problème du sac à dos multiple non linéaire dans le Chapitre 4.

La programmation non linéaire en nombres entiers est un sujet très difficile et fascinant dans le domaine de l'optimisation mathématique à la fois d'un point de vue théorique et computationnel. Ces problèmes sont caractérisés par une fonction objective et des contraintes non linéaires, ainsi

que des variables de décision continues et entières. Il est possible de formuler de nombreux problèmes dans ce schéma général et, habituellement, ils posent de réels défis en termes d'efficacité et de précision de la solution obtenue quant aux procédures de résolution. Un autre outil très important dans l'optimisation mathématique est représenté par les formulations et reformulations. En particulier, nous introduisons une nouvelle famille de reformulations, appelées reformulations ponctuelles, en fonction d'un paramètre donné. Elles sont plus simples à résoudre que la formulation originale. Une caractéristique remarquable recherchée dans la reformulation ponctuelle est l'exactitude, c'est-à-dire l'existence d'une valeur donnée du paramètre telle que un optimum global du problème d'origine est aussi un optimum globale pour la reformulation. L'idée de base pour résoudre heuristiquement les problèmes non linéaires en nombres entiers consiste à trouver l'optimum des reformulations ponctuelles exactes (les plus faciles), qui produit immédiatement l'optimum global correspondant du problème d'origine. Nous employons alors l'algorithme Multiplicative Weights Update afin d'identifier la valeur correcte du paramètre pour la reformulation ponctuelle.

Dans le Chapitre 1, nous définissons les concepts et les objets mathématiques utilisés dans le corps de la thèse. Le Chapitre 2 est consacré à illustrer le cadre général du nouvel algorithme, le Multiplicative Weights Update pour la programmation non linéaire en nombres entiers et ses principales propriétés théoriques. En outre, dans ce chapitre, nous définissons plusieurs procédures de construction automatique pour déterminer la reformulation ponctuelle d'un problème non linéaire en nombres entiers pour des classes spécifiques, mais larges, de problèmes d'optimisation.

Dans le corps de la thèse, nous nous occupons de deux problèmes d'optimisation difficiles: la sélection du portefeuille moyenne-variance et le problème du sac à dos multiple non linéaire. Dans le Chapitre 3, nous donnons un résumé des modèles, formulations et reformulations, ainsi que les méthodes spécifiques orientées sur le problème de la sélection du portefeuille moyenne-variance avec un seul objectif et sur une période unique. Parmi toutes les versions du problème de portefeuille proposé dans la littérature spécialisée, nous avons choisi la catégorie la plus abordée, appelée la sélection de portefeuille avec contrainte de cardinalité avec variables semi-continues. Nous considérons également une fonction de coût de transaction non-concave et non-convexe, avec la seule hypothèse de séparabilité, ce qui est tout à fait naturel dans le contexte des marchés financiers. Le Multiplicative Weights Update pour les problèmes non linéaires en nombres entiers fait preuve d'un meilleur comportement par rapport aux autres méthodes de résolution, notamment en termes de qualité de la solution produite et du nombre d'actions qui composent le portefeuille optimal. En général, en fait, la minimisation du nombre d'actions dans le portefeuille optimal est un deuxième objectif pour les procédures optimales de sélection.

Par conséquent, dans le Chapitre 4, nous considérons le problème du sac à dos multiple non linéaire, abordé ici pour la première fois dans son intégralité. Nous adaptons la structure du Multiplicative Weights Update à ce problème, proposant une nouvelle reformulation ponctuelle. Malheureusement, des expériences computationnelles poussées montrent que cette approche algorithmique n'est pas bien adaptée pour résoudre les cas difficiles de ce problème de sac à dos. Ensuite, nous illustrons une méthode heuristique différente basée sur la discrétisation de l'espace de solutions et sur sa relaxation agrégée. La méthode consiste en trois phases: nous proposons une procédure gloutonne constructive et deux algorithmes pour la récupération de la faisabilité de la relaxation agrégée. Une post-procédure de recherche locale est également exécutée afin d'améliorer la qualité globale de la solution produite par les heuristiques. Les expériences de calcul indiquent que cette méthode prévaut sur les autres tant en termes de qualité de la solution que de temps total de calcul.

ACKNOWLEDGMENTS

The last four years have been an unforgettable experience for me and actually this is due to many persons I had the chance to meet. First of all I want to thank my supervisors for their incredible help and support also during the moments of greater difficulty.

To Claudia D'Ambrosio for her extraordinary patience and guidance. She is one of the kindest person and successful researcher I met in my life. I am very glad for having been one of her PhD students.

To Leo Liberti for the ideas and the advices he shared with me, for the talent to light my passion for the research and for keeping me grounded. Thanks for all your illuminating suggestions.

To all the professors and researchers I had the chance to work with: Angelo Di Zio, Silvano Martello, Mathieu Van Vyve ...

To Sonia, Raouia, Claire, Gustavo, Andrea, Pierre-Louis, Olivier, Kostas, Ky, Youcef, Naveen, Eduardo, Maria, Christos, Panagiotis, Konstantinos, Fragkiskos, and all the guys I met at LIX during the last four years: I will never forget all the days at École Polytechnique I spent with you, friends.

To Maryam, Maribel, Malwina, Sven, Matteo, Andrea, Bartosz, Francesco, Ruobing, and Ahmadreza, and all MINO Initial Training Network people! To the Marie Curie 7th European Framework Programme for the financial support.

To the fantastic research groups of CORE at University of Louvain-la-Neuve (Manuela, Andrea, Ignacio, Cyrille) and of M.A.I.O.R. Srl (Samuela, Francesco, Leopoldo, Giuliano).

To Stefano Lucidi and Laura Palagi for the having indicated me this PhD open position and for your fantastic reference letters. Everything started while I was writing my Master thesis and without you I definitively would have lost this great opportunity.

To Jeff Linderoth, Jon Lee, Andrew R. Conn, Amitabh Basu and all the fantastic researchers I had the opportunity to meet during seminars or conferences: all those discussions with you have been an inspiring source for me.

To Flavia, Martina, Frédèric and all my friends in Rome: every time I return to my hometown, you make me feel really at home.

To my extraordinary family for all the unique support. Grazie e ancora grazie!

Paris, december 4th, 2017

ACRONYMS

BB	Branch-and-Bound
BC	Branch-and-Cut
DGP	Distance Geometry Problem
HUC	Hydro Unit Commitment
KKT	Karush-Kuhn-Tucker
LMI	Linear Matrix Inequality
L(MNLKP,λ)	Lagrange Relaxation of MNLKP
LP	Linear Problem
`LP`	Linear Programming
MILP	Mixed Integer Linear Problem
`MILP`	Mixed Integer Linear Programming
MNLKP	Multiple Linear Knapsack Problem
MINLP	Mixed Integer NonLinear Problem
`MINLP`	Mixed Integer NonLinear Programming
MIQP	Mixed Integer Quadratic Problem
`MIQP`	Mixed Integer Quadratic Programming
MNKP	Multiple NonLinear Knapsack Problem
MP	Mathematical Programming
MS	MultiStart
MVPS	Mean-Variance Portfolio Selection
MWU	Multiplicative Weights Update
NLKP	NonLinear Knapsack Problem
NLP	NonLinear Problem
`NLP`	NonLinear Programming
PRS	Pure Random Search
SDP	Semidefinitive Problem

SDP	Semidefinitive Programming
S(MNLKP,π)	Surrogate Relaxation of MNLKP
SOC	Second Order Cone
SOCC	Second Order Cone Constraint
SOCP	Second Order Cone Problem
SOCP	Second Order Cone Programming
SQP	Sequential Quadratic Programming

Part I.

Overview

1 INTRODUCTION

This thesis is devoted to introduce and analyze a new methodology to solve optimization problems, i.e., a broad class of problems, in which we want to find the minimum of a single deterministic objective function, in a finite or infinite set of feasible points, described by a finite number of inequalities and possibly implicit constraints such as integrality or membership in given polyhedra. In particular, we consider Mixed Integer NonLinear Problems (MINLPs), i.e., optimization problems involving continuous and discrete decision variables and possibly nonlinear terms in the objective function and the constraints.

The methodology we employ in this thesis involves three main ingredients: MultiStart algorithm, Multiplicative Weights Update algorithm, and reformulations. The MultiStart algorithm is a simple random (heuristic) procedure to globally solve optimization problems. The Multiplicative Weights Update algorithm can be explained as a stochastic (heuristic) method for a decision maker to iteratively take a decision among different choices, by observing the prediction of a finite number of *advisors*. Reformulations are a fundamental tool in optimization both from a theoretical and applicative viewpoint. The formulation describes the structure of a Mathematical Programming (MP). Reformulations change the symbolic structure of a MP formulation while keeping some of its mathematical properties invariant.

This chapter constitutes a foreword to the rest of the thesis: we introduce the notation and the mathematical entities we will use in the other chapters. The rest of this chapter is organized as follows. In Section 1.1 we give the mathematical formal definition of Mixed Integer NonLinear Programming (MINLP) and we remark its general properties. In Section 1.2 we describe a general classification for the algorithms for MINLP. In Sections 1.3 and 1.4 we describe the MultiStart and the Multiplicative Weights Update algorithms, respectively. In Section 1.5 we present a definition of formulations and reformulations of a given optimization problem, and we propose a general classification, based on the relationship between the original problem and its reformulated versions. Finally, in Section 1.6 the thesis structure is drawn with all the complete references from which the others chapters are sourced.

1.1 MIXED INTEGER NONLINEAR PROGRAMMING

Let $x \in \mathbb{R}^n$ be an n-dimensional vector of continuous decision variables and $y \in \mathbb{Z}^p$ be a p-dimensional vector of integer variables. The general MINLP is defined as follows:

$$\begin{aligned} \min \ & f(x,y) && \text{(1.1a)}\\ \text{s.t. } & g(x,y) \leqslant 0 && \text{(1.1b)}\\ & x \in X && \text{(1.1c)}\\ & y \in Y \cap \mathbb{Z}^p, && \text{(1.1d)} \end{aligned}$$

where $f(x,y) : \mathbb{R}^{n+p} \to \mathbb{R}$ and $g(x,y) : \mathbb{R}^{n+p} \to \mathbb{R}^m$ represent the objective function and the constraints, respectively. The sets $X \subseteq \mathbb{R}^n$ and $Y \subseteq \mathbb{R}^p$ are two polyhedra of suitable dimensions. Let $\mathcal{F}$ be the feasible set of the MINLP, i.e.,

$$\mathcal{F} := \{(x,y) \ : \ g(x,y) \leqslant 0, x \in X, y \in Y \cap \mathbb{Z}^p\}. \tag{1.2}$$

Moreover, we define projections of the feasible set over the continuous and discrete variables, respectively:

$$\begin{aligned} \mathcal{F}_X :=& \{x \in X \ : \ \exists y \in Y \cap \mathbb{Z}^p \text{ such that } g(x,y) \leqslant 0\} && \text{(1.3a)}\\ \mathcal{F}_Y :=& \{y \in Y \cap \mathbb{Z}^p \ : \ \exists x \in X \text{ such that } g(x,y) \leqslant 0\}. && \text{(1.3b)} \end{aligned}$$

Definition 1.1.1. *With a slight abuse of notation, we say that a MINLP is convex if $f(x,y)$ and $g_\ell(x,y)$ are convex for all $\ell \in \{1,\dots,m\}$, otherwise the MINLP is non-convex.* □

Remark 1.1.2. *We emphasize that the continuous relaxation of a convex MINLP, i.e., the nonlinear problem obtained by removing the integrality requirements of the variable y, has a convex feasible set.* □

Definition 1.1.3. *With a slight abuse of notation, we say that a MINLP is strictly convex if $f(x,y)$ and $g_\ell(x,y)$ are strictly convex for all $\ell \in \{1,\dots,m\}$.* □

If $p = 0$, the MINLP reduces to NonLinear Problem (NLP). If the objective function and the constraints are linear, the MINLP reduces to Mixed Integer Linear Problem (MILP). Finally, if $p = 0$ and the objective function and the constraints are linear, the MINLP reduces to Linear Problem (LP).

Then we introduce the definitions of neighborhoods of a given point $(\tilde{x},\tilde{y}) \in \mathbb{R}^n \times \mathbb{Z}^p$.

Definition 1.1.4. *Let $(\tilde{x},\tilde{y}) \in \mathbb{R}^n \times \mathbb{Z}^p$, then*

$$\begin{aligned} \mathcal{B}((\tilde{x},\tilde{y});\varepsilon) :=& \{(x,y) \in \mathbb{R}^n \times \mathbb{Z}^p \ : \ y = \tilde{y}, \|x - \tilde{x}\|_q \leqslant \varepsilon\} && \text{(1.4a)}\\ \mathcal{N}(\tilde{x},\tilde{y}) :=& \{(x,y) \in \mathbb{R}^n \times \mathbb{Z}^p \ : \ x = \tilde{x}, \|y - \tilde{y}\|_0 \leqslant 1\}, && \text{(1.4b)} \end{aligned}$$

where $q \in \mathbb{N}$ and $\varepsilon > 0$. □

Henceforth, *solving a MINLP* means finding at least one *global solution* for the optimization problem, i.e., a feasible point whose objective value is the minimum among all the points in the feasible set.

Definition 1.1.5. *A point* $(x^*, y^*) \in \mathcal{F}$ *is a* global solution *of a MINLP if* $f(x^*, y^*) \leqslant f(x, y)$ *for all* $(x, y) \in \mathcal{F}$. □

We introduce also the definition of *local optimum* of a MINLP.

Definition 1.1.6. *A point* $(x^*, y^*) \in \mathcal{F}$ *is a* local solution *of a MINLP if, for some* $\varepsilon > 0$, $f(x^*, y^*) \leqslant f(x, y)$ *for all* $x \in \mathcal{B}((x^*, y^*); \varepsilon) \cap \mathcal{F}_X$ *and* $y \in \mathcal{N}(x^*, y^*) \cap \mathcal{F}_Y$. □

A global solution is given by the best solution in all the feasible set: this definition takes into account the global behavior of the objective function and the entire feasible set. A local solution, on the contrary, is a solution restricted to a small subset of the feasible set, with respect to which the local behavior of the objective function is considered.

While a strictly convex NLP has at most one global solution, the same property does not necessarily hold for strictly convex MINLPs [154, 166].

MINLP is **NP**-hard because it includes Mixed Integer Linear Programming (MILP) [102, 139] and Mixed Integer Quadratic Programming (MIQP) [69] as special cases, when constraints are affine and objective function is linear or quadratic, respectively. MINLP is, in general, undecidable [130], even for $p \geqslant 10$, when the objective function is linear and the constraints are polynomial [67]. For a survey on computational complexity of MINLP we refer the interested reader to Hemmecke et al. [115] and Köppe [149]. However, if the MINLP is convex or polyhedra X and Y are bounded, the undecidability is fortunately avoided.

Usually, solving MINLPs in practice could be extremely difficult, as the following example shows.

Example 1.1.7. *Fermat's Last Theorem, formulated by Pierre de Fermat in 1637 on the margin of his copy of Arithmetica by Diophantus of Alexandria and published by his son Samuel de Fermat in Tolosa in 1670 (see [64, p. 62]) states:*

> *"Cubum autem in duos cubos, aut quadratoquadratum in duos quadratoquadratos et generaliter nullam in infinitum ultra quadratum potestatem in duos eiusdem nominis fas est dimidere cuius rei demonstrationem mirabilem sane detexi. Hanc marginis exiguitas non caperet.".*[1]

This conjecture, finally proved by Wiles [242], asserts the Diophantine equation $x^n + y^n = z^n$ *has no integer solution when* $n \in \mathbb{Z} \cap [3, +\infty)$ *and* $x, y, z \in \mathbb{Z}_+ \cap$

1 Translation: "It is impossible to separate a cube into two cubes, or a fourth power into two fourth powers, or in general, any power higher than the second, into two like powers. I have discovered a truly marvelous proof of this, which this margin is too narrow to contain". (see [114, p. 144-145]).

$[1, +\infty)$. *Now, Fermat's conjecture is false if and only if the optimum value of the following MINLP is zero [198]:*

$$\min_{x,y,z,n} \ (x^n + y^n - z^n)^2 \tag{1.5}$$
$$s.t. \ x \in \mathbb{Z}_+ \cap [1, +\infty) \tag{1.6}$$
$$y \in \mathbb{Z}_+ \cap [1, +\infty) \tag{1.7}$$
$$z \in \mathbb{Z}_+ \cap [1, +\infty) \tag{1.8}$$
$$n \in \mathbb{Z} \cap [3, +\infty). \tag{1.9}$$

□

For simplicity, from now on, we assume all the optimization problems we introduce are characterized by lower and upper bounds on the decision variables, i.e., by constraints such as $x \in [\underline{x}, \overline{x}]$ and $y \in [\underline{y}, \overline{y}]$, where the underline symbol indicates the lower bounds and the upperline symbol indicates the upper bounds.

1.2 MINLP ALGORITHMS

The algorithms proposed in the literature to solve optimization problems can be subdivided into two different classes, with respect to the type of solution produced:

- *local* algorithms produce *local solutions*;
- *global* algorithms produce *global solutions*.

Optimization algorithms can also be subjected to a further classification:

- *exact* algorithms;
- *heuristic* algorithms.

Exact algorithms produce a solution which meets a given optimization criterion which characterized a local or a global solution [30, 94, 201], while heuristic algorithms produce, in general quickly, a *good* feasible solution for the problem. Global algorithms consider the entire feasible set and they try to explore it in order to find a global optimum, exactly or heuristically. According to the way they search the feasible set, they can be:

- *deterministic* methods;
- *stochastic* methods.

Deterministic methods are such that, if applied several times to the same instance of the optimization problem, they produce the same output every

time; instead, stochastic methods are characterized by probabilistic procedures, which may produce different solutions each time the algorithm is executed on the same input.

One of the first general heuristic methods for *global* optimization problems, i.e., mathematical programs in which we want to find the *global minimum*, is the Pure Random Search (PRS) algorithm. The PRS (see Algorithm 1) is based on a simple restart procedure: at each iteration a new (feasible) point is generated (Step 3) and it is valued with respect to the objective function, if the current point has a better objective function value, the current returned point is updated (Steps 4-8).

Algorithm 1 Pure Random Search

1: Initially set t=1, $(x_0^*, y_0^*) := (x_0, y_0)$ and $U := f(x_0, y_0)$
2: **while** termination condition is not met **do**
3: generate a feasible point (x_t', y_t')
4: **if** $f(x_t', y_t') < U$ **then**
5: set $x_t^* := x_t'$, $y_t^* := y_t'$, $U := f(x_t^*, y_t^*)$
6: **else**
7: set $x_t^* := x_{t-1}^*$, $y_t^* := y_{t-1}^*$
8: **end if**
9: t := t+1
10: **end while**
11: **return** point (x_{t-1}^*, y_{t-1}^*) and upper bound U.

Step 3 can be implemented in many different ways from pure systematic generation to pure randomization procedure [183]. The termination condition is, in general, given by the maximum number of iterations or, in the case the global minimum value is known, the distance between the current function value and the global one [220].

Remark 1.2.1. *Although Algorithm 1 is quite simple, the implementation of this method cannot be a trivial task, since generating points in a feasible set is generally a non easy operation. One possible implementation is to generate a point in a box containing the feasible set and accept the point only if it is feasible.*

Assumption 1.2.2. *We assume the feasible set is defined only by lower and upper bounds constraints: in this case Step 3 can be implemented in a pure randomization fashion, i.e., the current point is generated according to a uniform distribution over the feasible set.* □

Proposition 1.2.3. *Let Assumption 1.2.2 hold, and $\{(x_t, y_t)\}$ be a sequence of uniformly random distributed points on the feasible set $\mathcal{F}$ of the optimization problem. Let $\mathcal{Z}_t = \{(x_1, y_1), \ldots, (x_t, y_t)\}$. Then, for all subset $\mathcal{A} \subset \mathcal{F}$ with strictly positive Lebesgue measure,*

$$\lim_{t \to \infty} \mathbb{P}\left(\mathcal{Z}_t \cap \mathcal{A} \neq \emptyset\right) = 1. \tag{1.10}$$

Proof. Let $\mathrm{meas}(\mathcal{A})$ denote the Lebesgue measure of set $\mathcal{A}$. $p = \mathrm{meas}(\mathcal{A})/\mathrm{meas}(\mathcal{F}) \in (0,1)$ is the probability that a point sampled in $\mathcal{F}$ belongs to $\mathcal{A}$. Therefore, we have:

$$\mathbb{P}\left(\mathcal{Z}_t \cap \mathcal{A} \neq \emptyset\right) = 1 - (1-p)^t. \tag{1.11}$$

The statement follows. □

Proposition 1.2.3 guarantees the entire feasible set is covered by the sequence of random points: no part of the feasible set remains unexplored. The following proposition states that the Algorithm 1 converges to the global optimum in probability as the number of iterations goes to infinity.

Proposition 1.2.4. *Let Assumption 1.2.2 hold and suppose the problem is feasible. Let $\{(x_t^*, y_t^*)\}$ be a sequence of points generated by the Algorithm 1. Then, for any $\varepsilon > 0$, we have*

$$\lim_{t\to\infty} \mathbb{P}\left((x_t^*, y_t^*) \in \{(x,y) \in \mathcal{F} : f(x,y) \leqslant f^* + \varepsilon\}\right) = 1, \tag{1.12}$$

where f^* is the optimal value of the optimization problem.

Proof. Let $\mathcal{A} = \{(x,y) \in \mathcal{F} : f(x,y) \leqslant f^* + \varepsilon\}$. The statement follows by invoking Proposition 1.2.3. □

The serious drawback of Algorithm 1 consists in the fact that, in order to reach a point in the neighborhood of the global solution of the optimization problem, many iterations may be necessary: in the worst case the number of iterations is infinite. The probability of finding a point (x_t, y_t) such that $f(x_t, y_t) \leqslant f^* + \varepsilon$ is $\varepsilon/\mathrm{meas}(\mathcal{F})$ and the probability of finding such point in N iterations is

$$1 - \left(1 - \frac{\varepsilon}{\mathrm{meas}(\mathcal{F})}\right)^N. \tag{1.13}$$

In order to produce a point in the neighborhood of the optimal solution of the problem with a confidence level α, i.e., such that $1 - \left(1 - \frac{\varepsilon}{\mathrm{meas}(\mathcal{F})}\right)^N = \alpha$, we need

$$N = \frac{\log(1-\alpha)}{\log(1 - \varepsilon/\mathrm{meas}(\mathcal{F}))} \tag{1.14}$$

iterations. Therefore, as pointed out by Locatelli and Schoen [165], if the optimization problem is pure continuous, i.e., $p = 0$, the feasible set $\mathcal{F}$ is a unit box, and the neighborhood of the global minimum is a box of edge length ℓ, the number of iterations needed is

$$\frac{\log(1-\alpha)}{\log(1-\ell^n)} = \mathcal{O}\left(\frac{1}{\ell^n}\right). \tag{1.15}$$

1.3 THE MULTISTART ALGORITHM

In order to speed up Algorithm 1 a *local search phase* can be also implemented, defining the MultiStart (MS) algorithm: at Step 4 in Algorithm 2 a local algorithm is applied to the randomly generated point. Obviously, in practical implementation, this local phase is crucial for the computational behavior of the algorithm since it is, in general, much more expensive than the global one [158].

Remark 1.3.1. *Note that, when the mathematical program is a non-convex NLP, Step 4 is* **NP***-hard [95, 204], however it is practically fast. Moreover, we can consider local NLP optimization as tractable as long as we limit ourselves to constraints involving only factorable functions over a simple operator alphabet such as* $+$, $-$, $*$, $/$, log, exp.

Algorithm 2 MultiStart Algorithm

1: Initially set $t := 1$, $(x_0^*, y_0^*) := (x_0, y_0)$ and $U := f(x_0, y_0)$
2: **while** termination condition is not met **do**
3: generate a feasible point (x_t', y_t')
4: apply a local algorithm from (x_t', y_t'), obtaining point (x_t'', y_t'')
5: **if** $f(x_t'', y_t'') < U$ **then**
6: set $x_t^* := x_t''$, $y_t^* := y_t''$, $U := f(x_t^*, y_t^*)$
7: **else**
8: set $x_t^* := x_{t-1}^*$, $y_t^* := y_{t-1}^*$
9: **end if**
10: $t := t+1$
11: **end while**
12: **return** point (x_{t-1}^*, y_{t-1}^*) and upper bound U.

In order to avoid too many local minimization procedures, variants of Algorithm 2 have been proposed: in particular, the local phase could be performed only if the randomization phase produces a better point in terms of the objective function [158].

Moreover, a *population* of points could be randomly generated in the randomization phase and a certain subset of points could be selected by means of *clustering techniques* (see [22, 158, 222]) and local minimizations can be performed, considering the selected points as starting points [158]. In other versions of the MS, an *escaping strategy* is also implemented in order to avoid local minima: for instance, Simulated Annealing [145] (see also [189, 43, 144, 68, 164]) is considered in the context of non-convex MINLP by Cardoso et al. [40] and Tabu Search [103, 104, 105] is applied to non-convex MINLP by Munawar et al. [197] (see also [160, 196]).

Finally, the generation of the random points can be implemented by means of non uniform distributions [230], such as in Simulated Annealing: in this

case the following proposition guarantees the convergence of the algorithm in probability.

Proposition 1.3.2. (Solis and Wets [230]) *Let* $\mathrm{meas}(A)$ *be the Lebesgue measure of set* A*. Let* $\{m_t(\cdot)\}$ *be a sequence of probability measures such that, for all (Borel) subset* $\mathcal{A} \subset \mathcal{F}$ *with* $\mathrm{meas}(A) > 0$*,*

$$\prod_{t=1}^{\infty} (1 - m_t(\mathcal{A})) = 0. \tag{1.16}$$

Then, if f *is a Lebesgue-measurable function and* $\mathcal{F} \subseteq \mathbb{R}^n$ *is a Lebesgue-measurable set, the sequence of random point* (x_t^*, y_t^*) *generated by the algorithm is such that*

$$\lim_{t\to\infty} \mathbb{P}\left((x_t^*, y_t^*) \in \{(x,y) \in \mathcal{F} \,:\, f(x,y) \leqslant f^* + \varepsilon\}\right) = 1, \tag{1.17}$$

where $\varepsilon > 0$*.*

1.4 THE MULTIPLICATIVE WEIGHTS UPDATE ALGORITHM

The Multiplicative Weights Update (MWU) algorithm is a "meta-algorithm", i.e., it could be adapted to many different settings, with a broad application in Optimization, Machine Learning, and Game Theory. In particular, we point out the Plotkin-Shmoys-Tardos Algorithm [206] for fractional packing and covering LPs, which can be derived by the MWU framework. The Plotkin-Schmoys-Tardos Algorithm is connected with the Lagrangian relaxation approach, in which several "complicated constraints" are relaxed. Let $A \in \mathbb{R}^{n\times n}$, $x, b \in \mathbb{R}^n$ and $\mathcal{P}$ be a convex subset of suitable dimension. The Plotkin-Schmoys-Tardos Algorithm solves the following feasibility problem:

$$\exists ? x \in \mathcal{P} : Ax \geqslant b \tag{1.18}$$

in which $x \in \mathcal{P}$ represent the "easy constraints", while $Ax \geqslant b$ indicate the "complicated constraints". The Algorithm calls repetitively a sub-procedure (Oracle) which deals with the following feasibility problem:

$$\exists ? x \in \mathcal{P} : p^\top Ax \geqslant p^\top b, \tag{1.19}$$

where p is a vector of suitable dimensions. Arora et al. [3] show that, if the sub-procedure is an (ℓ, ρ)-bounded Oracle, then there exists an algorithm which solves the problem (1.18) up to an additive error ε or derives that the problem (1.18) is infeasible. The algorithm calls the Oracle only $\mathcal{O}(\ell\rho \log(m)/\epsilon^2)$ times.

For a general explanation of MWU we refer the reader to the excellent survey by Arora et al. [3], from which we borrow the following example.

In a very simple stock market with just one stock characterized by two possible daily price movements (up and down), an investor has the possibility to observe the prediction of q *experts*. The investor wants to gain as

much as possible in terms of overall return, according to the prediction of the *best expert* which, however, is of course not known a priori. The first (trivial) algorithm one could think about consists in operating in the market, selling or buying the stock, according to the *majority opinion* of the experts. Nevertheless, the experts could be correlated or even not really experts in finance, so the majority opinion could be systematically wrong.

The MWU algorithm (see Algorithm 3) corrects the trivial one: the investor chooses his/her financial strategy according to the *weighted majority opinion* of the experts. Initially all the experts have the same weights w_i ($i \leqslant q$) (Step 1), but, as the time goes on, the algorithm gives a *gain* to the experts who made the correct prediction, and gives a *cost* to ones who made a wrong prediction (Step 4). For technical reasons the costs/gains must be in $[-1, 1]$. If this is not the case, a step to suitably scale the costs/gains has to be implemented. The *weight* of each expert is updated in a multiplicative fashion (Step 5). At each iteration, the investor chooses on the basis of the predictions in a random way according to the probability distribution induced by the weights (Step 3).

Algorithm 3 Multiple Weights Update Algorithm

1: Initially set t:=1, $\eta \leqslant \frac{1}{2}$ and $w_{i,t} := 1$ for all $i \leqslant q$
2: **while** termination condition is not met **do**
3: sample $i \leqslant q$ from the distribution $p_t \sim (w_{i,t} : i \leqslant q)$.
4: each decision incurs a cost/gain $\psi_t \in [-1, 1]^q$.
5: update weights $w_{i,t+1} := w_{i,t}(1 - \eta\psi_{i,t})$.
6: $t := t + 1$.
7: **end while**

Even though the MWU is a quite simple strategy, it is possible to show an upper bound for the overall expected cost $\sum_{t \leqslant T} p_t \psi_t$.

Theorem 1.4.1 (Arora et al. [3]). *Let T be the number of iterations, $\psi_{i,t} \in [-1, 1]$, for all $i \in q$ and $t \leqslant T$, be the cost/gain associated to expert i at time t and $\eta \leqslant \frac{1}{2}$. The MWU Algorithm guarantees an overall expected cost*

$$E_{MWU} := \sum_{t \leqslant T} \psi_t p_t \leqslant \min_{i \leqslant q} \left(\sum_{t \leqslant T} \psi_{i,t} + \eta \sum_{t \leqslant T} |\psi_{i,t}| \right) + \frac{\ln q}{\eta} \tag{1.20}$$

1.5 FORMULATIONS AND REFORMULATIONS

Intuitively a formulation is a way to write down a given optimization problem. Specific types of formulation are, for instance, required by the implemented solvers in order to address a specific type of problems, such as LPs, NLPs, MILPs, and MINLPs. Formulations can be:

- *flat* formulations;
- *structured* formulations.

In the objective and constraints of *structured* formulations, quantifiers, such as $\forall$, $\sum$, $\prod$, appear; in *flat* formulations no quantifier is present. When a formulation P is cast into another formulation Q, we say that Q is a *reformulation* of P.

Several definitions for reformulation have been proposed (see, e.g., Audet et al. [5] and Sherali [227]). Generally, reformulations are defined in such a way that several properties of the original formulation are preserved, such as the set of the optimal solutions or the set of the feasible points. A systematic theory for reformulations is presented in Liberti [157], which proposes the following classification.

Definition 1.5.1. *A reformulation* Q *of a formulation* P *is a* relaxation *if its feasible set contains the feasible set of* P.

Definition 1.5.2. *A reformulation* Q *of a formulation* P *is* exact *if it shares all the optimization properties (local optima, global optima, feasible set) with* P.

Moreover, in the next chapters we will consider also *bounding* reformulations [187].

Definition 1.5.3. *A reformulation* Q *of a formulation* P *is* bounding *if, when solved to optimality, produces a lower bound for* P *and its feasible set contains the feasible set of* P.

Remark 1.5.4. *Note that all the* relaxations *are also* bounding *reformulations.*

1.6 THESIS STRUCTURE

This thesis is based on several published papers, namely [187, 188], and other working papers [185, 186] submitted to the international refereed journals Computers & Operations Research and International Transactions in Operational Research, respectively. In particular, Chapter 2 is sourced from [187], Chapter 3 from [185], and Chapter 4 from [186, 188].

Beside the introductory section, the thesis is structured in two main parts: a theoretical part (Chapter 2), and an applicative one (Chapters 3 and 4). In the theoretical part we introduce a new algorithm to solve MINLP, and we explain its fundamental steps and its theoretical properties. In the second part we apply the methodology to two different optimization problems, namely the Mean-Variance Portfolio Selection and the Multiple NonLinear Knapsack Problems, both modeled as MINLPs and difficult to solve in practice. We will see that for the first problem the algorithm performs quite well with respect to the benchmarks, while this is not the case for the second problem.

Hence, for the sake of completeness, we describe a constructive heuristic procedure to find good solutions in a reasonable amount of computational time.

Part II.

Theory

2 THE MWU ALGORITHM FOR MINLP

2.1 INTRODUCTION

The MWU algorithm can be described as a special case of the MS algorithm to solve MINLPs. The MS algorithm is composed of two main steps: the choice of a random starting point, and a local optimization procedure, meaning we solve the problem with a local solver (for instance, we can heuristically solve non-convex problems by means of a solver which solves convex problems exactly). In the MWU algorithm for MINLPs we follow the same structure of the MS, but we introduce a strategy for generating *promising points* for the objective function and for the constraints. It is quite clear what "promising point" means in terms of objective function: if we consider a minimization problem and two feasible points, the one with the lowest objective function value is more *promising* than the other. Intuitively, being a "promising point" for the constraints corresponds to a sufficiently little violation of the constraints in terms of the difference between the value of the left hand side of the constraint and the value of the right hand side.

Moreover, since in the MWU algorithm for MINLPs we generate "promising points" according to the problem formulation, in order to guarantee low objective function values and low violation of the constraints, we say that MWU algorithm for MINLPs is a *matheuristic* [171], i.e., a heuristic algorithm based on the MP formulation. In other words, we generate the promising point by solving an auxiliary formulation of the original problem, where several terms are fixed to the values given by the MWU, we call a *pointwise reformulation*. The solution we obtain is the starting point for a local optimization procedure. Then, we iterate the two steps (solving the pointwise reformulation and applying a local procedure) for a given number of iterations or until a given optimality criterion is satisfied. If the pointwise reformulation is built in such a way that there exists a value of the parameter for which the reformulation has the same optimum as the original one, i.e., the pointwise reformulation is *exact*, then we only need to guess the correct value of the parameter and solve the simpler reformulation. If the pointwise reformulation is *bounding*, at each iteration we obtain a lower bound for the optimal value of the original formulation, by simply solving the reformulated problem.

The rest of this chapter is organized as follows. In Section 2.2 we formally introduce the pointwise reformulation, describing its theoretical properties and characterizations. In Section 2.3 we illustrate several examples of *automatic procedures* to build the pointwise reformulation for specific classes of

MINLPs. Finally, in Section 2.4 a complete description of the MWU algorithm for MINLPs is specified.

2.2 POINTWISE REFORMULATIONS

Definition 2.2.1. *Given a MINLP* P *as in* (1.1)*, a pointwise reformulation* $R^{\theta} = \underset{t \leftarrow t'(\theta)}{\text{ptw}}\ (P)$ *is a family of MINLP formulations, depending on a parameter* $\theta = (\theta_s \mid s \leqslant r)$*, which are obtained by replacing terms* $t_1, \dots, t_r$ *in* P *by corresponding parametrized terms* $t'_s(\theta_s)$ *(for* $s \leqslant r$*).* □

Given a MINLP P as in (1.1), a term of P is a symbolic expression in its expression tree, i.e., it is represented by a set of adjacent nodes in its expression tree. The expression tree of a MINLP is a way to computationally represent the problem in a tree, having as leaves the variables and the constants of the problem and as the other nodes the logical operators ($+$, $\times$, $/$, sin, etc.) linking variables and constants [54].

Both the original terms t and the substituting ones t' are functions of the decision variables x and y; furthermore, the substituting terms t' are also functions of the parameters θ: hence, we extensively indicate the substituted terms as $t(x, y)$ and the substituting terms as $t'(x, y; \theta)$.

Definition 2.2.2. *Given a MINLP* P *and* $R^{\theta} = \underset{t \leftarrow t'(\theta)}{\text{ptw}}\ (P)$*, both defined on a vector* x *of continuous decision variables in* $\mathbb{R}^n$ *and a vector* y *of discrete decision variables in* $\mathbb{Z}^p$*:*

(a) For every replaced term t_s *(for* $s \leqslant r$*) in Definition 2.2.1, let* D_s *be the range of* $t_s(x, y)$*, where the term is interpreted as a function of the decision variables* x *and* y *of* P *ranging in the respective domains.*

(b) For every replacement term t'_s *(for* $s \leqslant r$*) in Definition 2.2.1, let* $D'_s(\theta_s)$ *be the range of* $t'_s(x, y; \theta_s)$ *when the term is interpreted as a function of the decision variables* x *and* y *of* P *ranging in the respective domains.*

(c) For $s \leqslant r$*, let* Θ_s *be the range of the corresponding parameter* θ_s*, and let* $\Theta = (\Theta_s \mid s \leqslant r)$*.* □

Finally, given a parameter $\theta \in \Theta$ and a function ϕ, we indicate with ϕ^{θ} the function obtained by replacing the terms $t_s(x, y)$ with $t'_s(x, y; \theta)$ ($s \leqslant r$). Let X^{θ} and Y^{θ} be the sets obtained by replacing the terms $t_s(x, y)$ into the

inequalities defining sets X and Y, respectively. With the previous notation, we rewrite the pointwise reformulation as follows:

$$\min \; f^{\theta}(x,y) \tag{2.1a}$$
$$\text{s.t. } g^{\theta}(x,y) \leqslant 0 \tag{2.1b}$$
$$x \in X^{\theta} \tag{2.1c}$$
$$y \in Y^{\theta} \cap \mathbb{Z}^{p'}. \tag{2.1d}$$

Remark 2.2.3. *The number of the constraints in the pointwise reformulation (2.1) is the same as in the original formulation (1.1); however, since in the pointwise reformulation (2.1) several terms are substituted by means of parameters θ, the number of variables could be different from the one of the original formulation (1.1).* □

We introduce the following classification for the pointwise reformulations.

Definition 2.2.4. *Given a MINLP P and its pointwise reformulation R^{θ}:*

(a) R^{θ} *is* spanning *if, for any $x \in \mathbb{R}^n$ and $y \in \mathbb{Z}^p$, there are values of θ such that evaluating the functions of P and of R^{θ} at (x,y) determines the same value, i.e., such that*

$$\forall s \leqslant r \qquad D_s \subset \bigcup_{\theta_s \in \Theta_s} D'(\theta_s) \quad \wedge \quad t'_s(x,y;\theta_s) = t_s(x,y) \tag{2.2}$$

(b) R^{θ} *is* exact *if, for each globally optimal solution (x^*,y^*) of P, there is at least one vector $\theta' \in \Theta$ such that (x^*,y^*) is also an optimal solution of $R^{\theta'}$;*

(c) R^{θ} *is* efficient *if there is a polynomial-time algorithm for approximately solving R^{θ} (for $\theta \in \Theta$) to within a given $\varepsilon > 0$ approximation factor.*

□

Lemma 2.2.5. *Let P be a MNLP cast in form (1.1) and $R^{\theta} = \underset{t \leftarrow t'}{\mathrm{ptw}}(P)$ be a spanning reformulation, then we have:*

$$\mathrm{feas}(P) \subseteq \bigcup_{\theta \in \Theta} \mathrm{feas}(R^{\theta}). \tag{2.3}$$

where $\mathrm{feas}(P)$ is the feasible set of problem P.

Proof. Let $(x',y') \in \mathrm{feas}(P)$. Given that R^{θ} is spanning, there exists a parameter $\xi \in \Theta$ such that $t_s(x',y') = t_s(x',y';\xi_s)$ for each $s \leqslant r$. Hence it follows $g^{\xi}_{\ell}(x^*,y^*) = g_{\ell}(x^*,y^*)$ for all $\ell \leqslant m$. Note $(x',y') \in \mathrm{feas}(P)$, $g_{\ell}(x',y') \leqslant 0$ for all $\ell \leqslant m$, therefore (x',y') is also feasible for R^{ξ}. Since for all $\xi \in \Theta$ $\mathrm{feas}(R^{\xi})$ is a subset of $\bigcup_{\theta \in \Theta} \mathrm{feas}(R^{\theta})$, the statement follows. □

Lemma 2.2.6. *Let P be a given formulation and R^{θ} be a spanning pointwise reformulation, then there exists $\xi \in \Theta$ such that R^{ξ} is a* bounding reformulation *of P.*

Proof. In the proof of Lemma 2.2.5 we saw that, if $\xi_s \in \Theta$ is such that $t_s(x^*, y^*) = t_s(x^*, y^*; \xi_s)$ for all $s \leqslant r$, then $(x^*, y^*) \in \mathrm{feas}(R^{\xi})$. Analogously, it is possible to prove $f^{\xi}(x^*, y^*) = f(x^*, y^*)$, and therefore:

$$\mathrm{val}(R^{\xi}) \leqslant f^{\xi}(x^*, y^*) = f(x^*, y^*) = \mathrm{val}(P), \tag{2.4}$$

where $\mathrm{val}(P)$ indicates the value of the optimal solution of P. The statement of the theorem follows. □

2.3 GENERATING POINTWISE REFORMULATIONS

2.3.1 A First-Order Reformulation

In this section we derive a pointwise linear reformulation for a general non-convex MINLP. We assume functions f and g_ℓ ($\ell \leqslant m$) are at least once continuously differentiable. Moreover, we suppose without loss of generality that the objective functions of the original problem P and of its pointwise reformulation R^{θ} are the same, i.e., $f^{\theta} \equiv f$. If this is not the case, we add a dummy variable $\gamma \in \mathbb{R}$ to P, and we consider the following exact reformulation, in the sense of [157], of the original MINLP problem:

$$\begin{aligned}
\min\ & \gamma && (2.5a)\\
\text{s.t. } & f(x, y) \leqslant \gamma && (2.5b)\\
& g(x, y) \leqslant 0 && (2.5c)\\
& x \in X && (2.5d)\\
& y \in Y \cap \mathbb{Z}^p. && (2.5e)
\end{aligned}$$

We assume all the inequalities describing sets X and Y are included on inequalities $g(x, y) \leqslant 0$ with $g(x, y) : \mathbb{R}^{n+p} \to \mathbb{R}^m$. We replace each non-convex multivariate function in (2.5) with an affine approximation, i.e., with a first-order approximation at a given point $(\tilde{x}, \tilde{y})$:

$$\begin{aligned}
\min\ & \gamma && (2.6a)\\
& \nu_{00} + \nu_{10}{}^{\top} \begin{pmatrix} x - \tilde{x} \\ y - \tilde{y} \end{pmatrix} \leqslant \gamma && (2.6b)\\
& \nu_{0\ell} + \nu_{1\ell}{}^{\top} \begin{pmatrix} x - \tilde{x} \\ y - \tilde{y} \end{pmatrix} \leqslant 0 \qquad \forall \ell \leqslant m && (2.6c)\\
& y \in \mathbb{Z}^p. && (2.6d)
\end{aligned}$$

Remark 2.3.1. *In the pointwise reformulation (2.6), the parameter θ is the matrix $((\tilde{x}\ \ \tilde{y})^{\top}, \nu_{0k}{}^{\top}, \nu_{1k}{}^{\top})$ ($k \in \{0, \ldots, m\}$), whose dimensions are $(n + p) \times (2m + 3)$.* □

For simplicity of notation, in the rest of this subsection, we set $g_0(x, y) := f(x, y)$.

Lemma 2.3.2. *The pointwise reformulation (2.6) is spanning.*

Proof. For each $k \in \{0, \dots, m\}$, we note that the replacement term $v_{0k} + v_{1k}{}^{\top}(x - \tilde{x}\ \ y - \tilde{y})^{\top}$ and the replaced term $g_k(x, y)$ have the same value for all values of θ such that $(\tilde{x}, \tilde{y}) = (x, y)$ and $v_{0k} = g_k(x, y)$. □

The previous approach to generate a pointwise reformulation has been successfully applied to a special class of the Hydro Unit Commitment (HUC) problems arising in energy industry in the paper [187].

2.3.2 Polynomial MINLPs

In this section, we consider polynomial MINLPs, i.e., formulations (1.1) where functions f and g_ℓ ($\ell \leqslant m$) are polynomials of (x, y). For simplicity in this section we assume $p = 0$. All the results, except where explicitly indicated, are valid also in case discrete decision variables are present.

A Trivial Pointwise MILP Reformulation

A trivial method to derive a pointwise MILP reformulation for any polynomial MINLP P consists in turning every variable but one to a parameter in every monomial of P. For two integer sequences $\mathbf{h} = (h_1, \dots, h_t)$ and $\mathbf{d} = (d_1, \dots, d_t)$, let

$$\mu_{\mathbf{h}}^{\mathbf{d}} = \prod_{j \leqslant t} x_{h_j}^{d_j}$$

be a monomial occurring in P. Let us see the MILP pointwise reformulation first:

1. introduce new parameters $\theta_1, \dots, \theta_t$;
2. for each monomial $\mu_{\mathbf{h}}^{\mathbf{d}}$ in P:
   a) choose $\ell \leqslant t$ as $\ell = \operatorname{argmin}_{j \leqslant t}\{d_j\}$
   b) replace μ by the linear term $a_\mu x_{h_\ell}$, where

$$a_\mu = \theta_{h_\ell}^{d_\ell - 1} \prod_{j \neq \ell} \theta_{h_j}^{d_j}.$$

Since every monomial is reduced to a linear term, but the rest of the constraints remains unchanged, it is immediate to show that the formulation obtained by the above procedure is a pointwise MILP reformulation of P.

We point out that, if, with respect to the monomial $\mu_{\mathbf{h}}^{\mathbf{d}}$, there exists an index $k \leqslant t$, such that $d_k = 1$, because of the rule (a), we do not introduce θ_{h_k} in the pointwise reformulation keeping the number of the parameters as low as possible.

Remark 2.3.3. *By construction, the previous pointwise reformulation is spanning since, for each feasible $x_{\mathbf{d}}$, the replaced term μ is always equal to the replacement term $x_{h_\ell}\theta_{h_\ell}^{d_\ell - 1}\prod_{j \neq \ell}\theta_{h_j}^{d_j}$ for $\theta_{h_j} = x_{h_j}$ ($j \leqslant t$).* □

2.3.3 Bilinear MINLPs

The trivial pointwise MILP reformulation appears particularly well suited to the case of bilinear MINLPs where the incident graph corresponding to the matrix of the quadratic form is bipartite, i.e., polynomial MINLPs of degree 2 where the variables can be partitioned into two sets I and J such that every monomial $x_i x_j$ has $i \in I$ and $j \in J$. Then the procedure reduces to choose I (resp. J) and replace x_i with θ_i (resp. x_j with θ_j) for all $i \in I$ (resp. $j \in J$).

2.3.4 Quadratic MINLPs

Consider MINLPs involving quadratic terms only and assume without loss of generality $p = 0$:

$$\min\ x^\top Q^0 x + q^0 x \tag{2.7a}$$
$$\text{s.t. } x^\top Q^\ell x + q^\ell x \leqslant b_\ell \qquad \forall \ell \leqslant m \tag{2.7b}$$

Let L be the set of indices in $\{0, \dots, m\}$ such that Q^ℓ is indefinite. For each $\ell \in L$, we look for two matrices A^ℓ, B^ℓ such that:

(i) A^ℓ is positive semidefinite;

(ii) B^ℓ is sparse;

(iii) $A^\ell + B^\ell = Q^\ell$.

Next, we rewrite $x^\top Q^\ell x + q^\ell x$ as $x^\top A^\ell x + x^\top B^\ell x + q^\ell x$, introduce as many parameters θ as there are variables x, and define the pointwise reformulation by replacing every term $x^\top B^\ell x$ by $\theta^\top B^\ell x$. This yields a pointwise convex Mixed Integer Quadratic Problem (MIQP) reformulation, since all the non-convex terms $x^\top B^\ell x$ have been turned into linear terms. Thereby, given the indefinite matrix Q^ℓ, we aim the following optimization problem to recover $A^\ell \succeq 0$ and B^ℓ:

$$\min\ \|B^\ell\|_0 \tag{2.8a}$$
$$\text{s.t. } A^\ell \succeq 0 \tag{2.8b}$$
$$A^\ell + B^\ell = Q^\ell. \tag{2.8c}$$

A convex relaxation of (2.8) can be obtained by replacing, in the objective function, the ℓ_0-norm with ℓ_1-norm, namely

$$\min\ \|B^\ell\|_1 \tag{2.9a}$$
$$\text{s.t. } A^\ell \succeq 0 \tag{2.9b}$$
$$A^\ell + B^\ell = Q^\ell. \tag{2.9c}$$

Remark 2.3.4. *Let $\lambda_{\min}$ be the minimum eigenvalue of Q^ℓ, and $\mathbf{1}_{n\times n}$ indicate the identity matrix of order n. $B^\ell = \lambda_{\min}\mathbf{1}_{n\times n}$ is a good solution for both (2.8) and (2.9) (see Section 4 in [97]).* □

Remark 2.3.5. *By construction, the pointwise convex MIQP reformulation previously introduced is spanning for all* $\theta = x$. □

2.4 MWU ALGORITHM FOR MINLPS

The pseudocode of the MWU algorithm for MINLPs is shown in Algorithm 4. It takes a MINLP formulation P as input and produces a local solution as output. In the next sections we analyze in more detail the steps of the algorithm.

Algorithm 4 MWU Algorithm for MINLPs

1: assign weights $w_1 := \mathbf{1}$, incumbent $(x^*, y^*) := (\infty, \infty)$ and $t := 1$
2: **while** $t \leqslant T$ **do**
3: sample θ_t from the distribution $p_t \sim (w_{i,t} \mid i \leqslant q)$
4: solve $R^{\theta_t} = \underset{t \leftarrow t'(\theta_t)}{\text{ptw}}\ (P)$, get solution (x_t, y_t)
5: optionally refine (x_t, y_t) (e.g. using local descent)
6: if (x_t, y_t) is better than the incumbent, set $(x^*, y^*) = (x_t, y_t)$
7: (x_t, y_t) yields decision costs $\psi_t \in [-1, 1]^q$
8: update weights for next iteration: $w_{i,t+1} \leftarrow w_{i,t}(1 - \frac{\psi_{i,t}}{2})$
9: increase t.
10: **end while**
11: **return** (x^*, y^*)

2.4.1 Sampling

In MWU, a weight is maintained for each *advisors*. In this case we want to estimate parameters θ: hence, we have to take q decisions at each step, not only one. In Step 3 we randomly select r values for the parameters according to the distribution of the weights: we have one weight, and therefore one *expert*, for each decision.

In general, the number r of the parameters and the dimension q of the weights w could be different. Techniques of aggregation, if $r < q$, or disaggregation, if, on the contrary, $r > q$, must be implemented before the sampling step. For instance, in [187] an example of disaggregation methodology is given for the Euclidian Distance Geometry Problem (DGP).

2.4.2 Solution and Refinement

Usually, solving the pointwise reformulation is not an easy task if the reformulation is not efficient. However, if the reformulation is *good* and easier to solve than the original formulation, the proposed methodology can be successfully exploited. This is the case, for instance, when the original prob-

lem P is a non-convex MINLP and the reformulation is a convex MINLP or even a MILP.

Remark 2.4.1. *In order to obtain an upper bound, we do not need to solve the pointwise reformulation exactly, but we can use* any heuristic we like. *Several termination criteria can be considered, such as the number of iterations or the optimality gap.* □

The refinement procedure (Step 5) is not expressly needed to guarantee the upper bound on the cumulative error of MWU algorithm. Nevertheless, computational experiments show it is necessary in practice to speed up the algorithm. Moreover, if we consider Algorithm 4 as a special case of the two-phase MS algorithm, the refinement step coincides with a local phase in which we locally improve the initial solution.

2.4.3 Computing MWU Costs/Gains

This is the most critical step of the MWU algorithm, since it can influence the performance guarantee. It has basically two requirements: (a) each cost vector ψ_t should have components in $[-1, 1]$; (b) if θ_t replaces several terms in the original formulation, ψ_t reflects the contribution of its value to optimality and feasibility. Requirement (a) is necessary for the MWU performance guarantee to hold, while requirement (b) tries to correlate E_{MWU} (see (1.20)) to suboptimality and infeasibility.

Hence, $\psi_{i,t}$ consists of a scaled contribution α_t to the error of (x_i, y_i) from the suboptimality of (x_t, y_t) in P and of β_t from its infeasibility. Since it is not necessary to reward "better feasibility", we consider β_t belonging to [0,1].

Let $f^\theta(x, y)$ and $g^\theta_\ell(x, y) \leqslant 0$ ($\ell \leqslant m$) be respectively the objective function and the constraints of R^θ.

After Steps 4-5, we can evaluate the current solution (x_t, y_t) in the original formulation P by computing $f(x_t, y_t)$ and $g_\ell(x_t, y_t)$ for each ℓ. We let α_t be proportional to $f^\theta(x_t, y_t) - f(x_t, y_t)$ and $\beta_t = (\beta_{\ell,t} : \ell \leqslant m)$ to $(\max(g^\theta_\ell(x_t), 0) : \ell \leqslant m)$. These quantities have, however, to be scaled in order to guarantee $\psi_t = [-1, 1]^q$. For simplicity, in the rest of the Section, we assume the original problem is continuous, i.e., $p = 0$. Formally, here is how ψ_t is computed:

1. for $t = 1$, let $\alpha_t = \operatorname{sgn}(f^\theta(x_t) - f(x_t))$ and $\beta_{\ell,t} = \operatorname{sgn}(\max(g_\ell(x_t), 0))$ for all ℓ;
2. for each $t > 1$ let:

$$\alpha_t = \frac{f^\theta(x_t) - f(x_t)}{\max\limits_{s \leqslant t} |f^\theta(x_s) - f(x_s)|} \quad \text{and} \quad \beta_{\ell,t} = \frac{\max(g_\ell(x_t), 0)}{\max\limits_{s \leqslant t}(\max(g_\ell(x_s), 0))}$$

for all ℓ;

3. $\Psi_t = \frac{1}{m+1}(\alpha_t + \sum_{\ell \leqslant m} \beta_{\ell,t})$;
4. for each $s \leqslant r$ let $\psi_{s,t} = \Psi_t \frac{|t_s(x_t, y_t) - t'_s(x_t, y_t; \theta_t)|}{\max(|t_s(x_t, y_t)|, |t'_s(x_t, y_t; \theta_t|)}$.

Remark 2.4.2. *Ψ_t could be the result of many different ways to combine of α_t and the $\beta_{\ell,t}$ depending from the specific application.* □

Remark 2.4.3. *Since $\Psi_t \in [-1, 1]$ and $\frac{|t_i(x_t, y_t) - t'_i(x_t, y_t; \theta_t)|}{\max(|t_i(x_t, y_t)|, |t'_i(x_t, y_t; \theta_t|)} \in [0, 1]$, we have $\psi_t \in [-1, 1]$.* □

2.5 CONCLUSIONS

In this chapter we have given a theoretical insight with regard to the MWU algorithm for MINLPs. In particular we have introduced the pointwise reformulation of a given MP formulation and we have stated its main properties. Moreover, we have drawn several building procedures for the pointwise reformulations with respect to several classes of MINLPs. Finally, we have discussed in detail the main steps of the MWU algorithm.

Part III.

Applications

3 MEAN-VARIANCE PORTFOLIO SELECTION PROBLEM

3.1 INTRODUCTION

The first milestone in modern single-period portfolio selection theory is undoubtedly Harry Markowitz's 1952 seminal paper [177] (for an historic perspective, see also [214] and surveys [241, 55, 80]), in which the mean-variance portfolio optimization model was proposed for the first time. Although several ideas and results are already introduced by de Finetti [65] (see [7] for the English translation of the first chapter "The problem in a single accounting period"), the contribution of the Italian mathematician was discovered only recently by the financial international community (see [213, 212, 209]) and was acknowledged by Harry Markowitz himself [175].

The mean-variance approach is based on the fundamental observation that, according to what Markowitz states in [177], the investors should try to increase their portfolio return and contemporaneously to decrease, as much as possible, its volatility or its risk (see also [178, 180]). The portfolio variance is the most widely used measurement of the portfolio volatility; other possible risk measurements are reported, for instance, in [48, 120, 173]. If the expected returns of the assets follow a Gaussian distribution [112] or the investor's utility function is quadratic [41], then the mean-variance criterion is theoretically compatible with the expected utility hypothesis originally introduced by Bernoulli [15] (see [232] for the English translation and see also [93, 121, 240] for more modern approaches). As pointed out by Markowitz in [176, 179], the previous assumptions are sufficient, but not necessary conditions. However, the assumption of Gaussian asset returns might be unrealistic: the probability distribution for the expected returns is generally leptokurtic [191].

The resulting mathematical program might not represent completely the problem solved nowadays by the practitioners, but it can be enriched with various constraints to model the different characteristics of the modern financial markets. Moreover, the mean-variance approach considers only the first- and second-order moments of the probability distribution of the returns: consequently, in specific situations, this approach might lead to counterintuitive or even paradoxical solutions [56].

Kallberg and Ziemba [137] compare the effects of different utility functions with respect to the optimal portfolios when the distribution of the expected returns is Gaussian, and show empirically that utility functions with similar absolute risk aversion indices – defined by Arrow [4] and Pratt [208], but

originally introduced by de Finetti [66] (see also [41, 194]) – give rise to similar optimal portfolios.

The reminder of the chapter is organized into two parts. In the first part we survey convex MIQP approaches to solve the Mean-Variance Portfolio Selection (MVPS) problem: in the next section we present the mathematical formulation of the portfolio problem with quadratic risk measure according to Markowitz [177] and we analyze its main disadvantages; Sections 3.3, 3.4, and 3.5 provide several ways to enrich the original formulation in order to overcome its drawbacks; several equivalent mathematical reformulations for the mean-variance *probabilistic* portfolio problem are described in Section 3.6; finally, in Section 3.7 we summarize exact methods proposed in the literature to solve MVPS problems.

In the second part of the chapter, in Section 3.8, we choose a specific family of portfolio problems with cardinality constraint and transaction costs and we apply the MWU algorithm to this class of uncountable many MVPS problems. The aim of this part consists in a computational study of the performances of MWU, with a particular focus on the behavior of MWU depending on the *degree of nonlinearity* of the cost function. We propose a promising pointwise reformulation for this class of problems and a procedure to compute costs/gains in the MWU framework. Computational experiments show that the algorithm outperforms the benchmarks with respect to the quality of the solution produced. A summary of the main notation used throughout the chapter is reported in the Appendix A.

Sections 3.3.2 and 3.4.1-3.4.3 are entirely based on the papers by Bonami and Lejeune [26] and Lejeune [152]. Section 3.6.1 is completely based on [26], while Sections 3.6.2-3.6.3 on [89].

3.2 PORTFOLIO OPTIMIZATION

We consider r possibly risky assets characterized by a mean return vector $\bar{\mu} \in \mathbb{R}^r$. Let $x \in \mathbb{R}^r_+$ be the vector, whose generic entry j ($j = 1, \dots, r$) represents the fraction of the portfolio value invested in asset j. For the moment, following Markowitz [177], we assume that the entries of $\bar{\mu}$ and of the covariance return matrix $\bar{\Sigma} \in \mathbb{R}^{r \times r}$ are known precisely. In the mean-variance approach, we aim to minimize the portfolio variance $x^\top \bar{\Sigma} x$ under the constraint that the portfolio return is at least equal to a given level $R > 0$. Therefore, the problem we aim to solve can be stated as follows:

$$\min \; x^\top \bar{\Sigma} x \tag{3.1a}$$

$$\text{s.t. } \bar{\mu}^\top x \geqslant R \tag{3.1b}$$

$$\mathbf{1}_r{}^\top x = 1 \tag{3.1c}$$

$$x \geqslant 0, \tag{3.1d}$$

where $\mathbf{1}_r \in \mathbb{R}^r$ is the all-one vector. Then, by repeatedly solving problem (3.1) for different values of return R, we can compute the efficient frontier, i.e., the set of the non-dominated portfolios in the sense of Pareto optimality. In several cases (see, e.g., [26]) an additional non-risky asset with mean $\overline{\mu}_0$ and zero variance is also considered, in order to algorithmically derive the efficient frontier (see Section 3.4.8). Several papers (see, e.g., [61]) consider an equality version for the return constraint (3.1b), namely $\overline{\mu}^\top x = R$.

Constraint (3.1c) ensures that the whole capital available is invested in the portfolio and in several papers (see, e.g., [38]) is substituted by

$$\mathbf{1}_r{}^\top x \leqslant 1. \tag{3.2}$$

Constraint (3.1d) prevents short selling, i.e., the possibility for the investor to sell financial assets not already in his/her portfolio. This financial operation is generally performed with speculative intents when the investor expects a bearish trend in the financial stock market. In case short selling is allowed, (3.1c) can be replaced by the constraint $\mathbf{1}_r{}^\top x = 0$, which defines the so-called dollar neutral portfolio, by requiring the exposure on long part of the portfolio to be equal to the one on the short part. Several authors consider the case where the decision variables x represent the absolute amount invested per asset so that the inequality (3.1c) becomes $\mathbf{1}_r{}^\top x \leqslant B$, where B is the investor's total initial budget.

In [37], Buchheim et al. introduce the budget constraint:

$$v^\top x \leqslant B, \tag{3.3}$$

where the decision variables x are the units of financial asset held in the investor's portfolio and $v \in \mathbb{R}^r_+$ is the vector of the costs per unit of corresponding asset.

Problem (3.1) is a convex continuous linearly constrained quadratic program, because, by definition, matrix $\overline{\Sigma}$ is symmetric and positive semidefinite; hence, we have a computationally tractable problem. However, the main drawback of this model consists in the sensitivity of the optimal solutions with respect to the input parameters (expected returns and covariance matrix), which are clearly unknown in real-world applications (see, e.g., [35, 36, 42, 50, 86, 131, 136, 132, 190]). Furthermore, when $\overline{\Sigma}$ is estimated starting from empirical measurements, it might happen that semidefiniteness is not directly satisfied and some ad-hoc procedures are required (see [60, 117, 222]).

Chopra [52] empirically analyzes the effects of slight differences in the estimate. Best and Grauer [19] conduct a theoretically rigorous analysis with computational results about the sensitivity of mean-variance efficient portfolios with respect to possible changes in asset means.

Several papers study the instability and ill-conditioning of problem (3.1): for instance, Kallberg and Ziemba [138] consider estimation errors in the investor's utility function and the mean vector and covariance matrix of the return distribution for normally distributed portfolio selection problems and observe that errors in mean vector give rise to significant problems. Chopra and Ziemba [53] show that the estimating errors with respect to the expected return means is generally one order of magnitude larger than the one corresponding to estimating errors in asset variances or covariances, assuming negative exponential utility function with joint normal distribution of returns.

3.3 ROBUST AND PROBABILISTIC APPROACHES

3.3.1 Robust Approaches

The robust version of the mean-variance problem (3.1) has been considered in quite recent works (see the surveys [81, 101]). It consists in assuming that the expected returns are uncertain and their expected values and variances belong to a given set. By supposing that the unknown input parameters belong to a given uncertainty set (see [13, 14, 77, 79]), it is possible to show some theoretical results: for instance, Goldfarb and Iyengar [106] and Tütüncü and Koenig [237] established the robust portfolio selection problem can be formulated as a Second-Order Cone Problem (SOCPs) (see [1, 12, 31, 163]) for ellipsoidal and box uncertain sets, respectively.

Under the assumption that the return mean belongs to a convex polytope, whose vertices are known, Costa and Paiva [57] prove that program (3.1) can be formulated as a Linear Matrix Inequalities (LMI) problem (see [32, 200]). Moreover, El Ghaoui et al. [78] show that, when the mean and the covariance are unknown, but bounded, the worst-case mean-variance portfolio selection problem can be reformulated as a Semidefinite Program (SDP) (see [12, 200, 217, 238]).

Finally, Ye et al. [246] introduce uncertain sets both for the mean vector and the second moment matrix of the returns, showing the connection between the fully robust portfolio selection problem with box uncertain set for the mean and ellipsoid uncertain set for the second moment of the returns and SOCP, SDP, and Semi-Infinite Programming (see [247]).

3.3.2 Probabilistic Approach

Bonami and Lejeune [26] take into account the uncertainty in the expected assets returns by dealing with a probabilistic problem and by introducing a probabilistic constraint, which imposes that the expected return of the op-

timal portfolio should be not less than a given return level R with a high probability $p > 0$.

Let ξ be the random vector representing the expected returns of the r risky assets. We assume that the random vector ξ admits a probability density function and the density function of $\xi^\top x$ is strictly positive. Moreover, let $\mu \in \mathbb{R}^r$ with $\mu = \mathbb{E}[\xi]$ and $\Sigma = \mathbb{E}\left[(\xi - \mu)(\xi - \mu)^\top\right]$ be the mean and the covariance matrix for the r-variate distribution of ξ, respectively. Formulation

$$\begin{aligned} \min\ & x^\top \Sigma x && \text{(3.4a)}\\ \text{s.t. } & \mathbb{P}\left(\xi^\top x \geqslant R\right) \geqslant p && \text{(3.4b)}\\ & \mathbf{1}_r{}^\top x = 1 && \text{(3.4c)}\\ & x \geqslant 0 && \text{(3.4d)} \end{aligned}$$

is usually referred to as the probabilistic Markowitz formulation and its deterministic equivalent defines a NLP (see [26, 89, 90, 153]). Let $\psi = (\xi^\top x - \mu^\top x)/\sqrt{x^\top \Sigma x}$ be the standardized random variable representing the normalized portfolio return. Equation (3.4b) can be equivalently rewritten, as follows:

$$\mathbb{P}\left(\xi^\top x \geqslant R\right) = \mathbb{P}\left(\psi \geqslant \frac{R - \mu^\top x}{\sqrt{x^\top \Sigma x}}\right) = 1 - F_{(x)}\left(\frac{R - \mu^\top x}{\sqrt{x^\top \Sigma x}}\right), \tag{3.5}$$

where $F_{(x)}(\cdot)$ is the cumulative distribution of the standardized portfolio return. We assume that $F_{(x)}(\cdot)$ is a continue strictly increasing function. Moreover, we point out that the analytic form of the probability distribution F depends on the portfolio weights x. It follows that the probabilistic constraint (3.5) becomes

$$\begin{aligned} 1 - F_{(x)}\left(\frac{R - \mu^\top x}{\sqrt{x^\top \Sigma x}}\right) \geqslant p &\iff 1 - p \geqslant F_{(x)}\left(\frac{R - \mu^\top x}{\sqrt{x^\top \Sigma x}}\right)\\ &\iff \mu^\top x + F^{-1}_{(x)}(1-p)\sqrt{x^\top \Sigma x} \geqslant R, \end{aligned} \tag{3.6}$$

where $F^{-1}_{(x)}(\cdot)$ is the inverse of the cumulative distribution $F_{(x)}(\cdot)$ and $F^{-1}_{(x)}(1-p)$ is the $(1-p)$-quantile of $F_{(x)}(\cdot)$. Therefore, the deterministic equivalent of optimization problem (3.4) corresponds to the following NLP [141]:

$$\begin{aligned} \min\ & x^\top \Sigma x && \text{(3.7a)}\\ \text{s.t. } & \mu^\top x + F^{-1}_{(x)}(1-p)\sqrt{x^\top \Sigma x} \geqslant R && \text{(3.7b)}\\ & \mathbf{1}_r{}^\top x = 1 && \text{(3.7c)}\\ & x \geqslant 0. && \text{(3.7d)} \end{aligned}$$

In the following, we survey for which classes of probability distributions the problem can be reformulated as a SOCP. We thus recall several definitions in probability theory and convex optimization.

Definition 3.3.1. (Serfling [225]) *Let $\xi \in \mathbb{R}^r$ be a random variable, whose probability density function is $f : \mathbb{R}^r \to \mathbb{R}$. If $f(\xi - \theta) = f(\theta - \xi)$, then ξ has a distribution that is centrally symmetric about $\theta \in \mathbb{R}^r$.*

Definition 3.3.2. (Boyd and Vandenberghe [31]) *Let $x \in \mathbb{R}^n$ be the decision variables and $A_i \in \mathbb{R}^{(n_i-1)\times r}$, $H \in \mathbb{R}^{p\times r}$ and $h, c_i \in \mathbb{R}^r$, $\beta_i \in \mathbb{R}^{n_i-1}$, $g \in \mathbb{R}^p$, $d_i \in \mathbb{R}$ $(\forall i \in \{1,\dots,n\})$ be the parameters of a given convex continuous optimization problem and $\|\cdot\|_2$ indicates the Euclidean norm. If a convex continuous optimization problem can be (equivalently) rewritten as follows,*

$$\begin{aligned} \min\ & h^\top x \\ s.t.\ & \|A_i x + \beta_i\|_2 \leqslant {c_i}^\top x + d_i, \qquad i = 1,\dots,n \\ & Hx = g, \end{aligned} \tag{SOCP}$$

then it is an SOCP. A constraint is a Second-Order Cone Constraint (SOCC) of dimension n_i, if it can be equivalently rewritten as

$$\|A_i x + \beta_i\|_2 \leqslant {c_i}^\top x + d_i. \tag{SOCC}$$

Remark 3.3.3. (Boyd and Vandenberghe [31]) *Observe that SOCP generalizes Quadratically Constrained Programming, i.e., the case when $c_i = 0$ for all $i = 1,\dots,n$: a Quadratically Constrained Problem can be obtained by squaring the constraints. SOCP generalizes also Liner Programming (LP), i.e., the case when instead $A_i = \mathbf{0}_{(n_i-1)\times r}$ for all $i = 1,\dots,n$, where $\mathbf{0}_{(n_i-1)\times r}$ is the zero matrix of suitable dimensions.* □

Definition 3.3.4. (Lobo et al. [163]) *A convex set $C \subseteq \mathbb{R}^r$ is SOC-representable if it is equivalent to the intersection of a finite number of SOCC, i.e., there exist parameters $A_i \in \mathbb{R}^{(n_i-1)\times(r+m)}$, $\beta_i \in \mathbb{R}^{n_i-1}$, $c_i \in \mathbb{R}^{r+m}$, and $d_i \in \mathbb{R}$ such that*

$$x \in C \iff \exists y \in \mathbb{R}^m : \left\| A_i \begin{pmatrix} x \\ y \end{pmatrix} + \beta_i \right\|_2 \leqslant {c_i}^\top \begin{pmatrix} x \\ y \end{pmatrix} + d_i \quad i = 1,\dots,n.$$

Moreover, a given function $f : \mathbb{R}^r \to \mathbb{R}$ is SOC-representable if the set $\{(x,t) : f(x) \leqslant t\}$ is SOC-representable.

Ultimately, given an objective function $f : \mathbb{R}^r \to \mathbb{R}$ and a feasible convex set $C \subseteq \mathbb{R}^r$, which are SOC-representable, then the corresponding convex optimization program, i.e.,

$$\begin{aligned} \min\ & f(x) \\ \text{s.t. } & x \in C, \end{aligned} \tag{SOC}$$

can be dealt with as a SOCP by means of efficient interior point methods (see, for instance, [98, 199, 200, 207, 244]), characterized by polynomial time computational complexity [108, 200]. Bonami and Lejeune [26] showed some convexity results for problem (3.7) that we briefly discuss in the following for the sake of completeness.

Theorem 3.3.5. (Bonami and Lejeune [26]) *Let* $p \in [0.5, 1)$. *If the probability distribution of* $\xi^\top x$ *is centrally symmetric, then the deterministic constraint* (3.7b), *equivalent to* (3.5), *is a SOCC.*

Therefore, optimization problem (3.4) is an SOCP because its objective function is convex quadratic and its feasible region is described by the intersection of a second-order cone and several linear constraints. Constraint $\mu^\top x \geqslant R - F^{-1}_{(x)}(1-p)\sqrt{x^\top \Sigma x}$ ensures that the expected portfolio return is greater than the given return plus a penalty term, which is function of the portfolio variance and is increasing with the confidence level p [89].

We recall also the definition of the skewness of a multi-variate distribution of a real-valued random variable ξ with mean μ and standard deviation σ [6]:

$$\operatorname{skew}(\xi) = \frac{\mathbb{E}\left[\xi - \mu\right]^3}{\sigma^3}, \tag{3.8}$$

The skewness is basically an asymmetry index of the distribution: perfectly symmetric distributions have zero skewness.

Definition 3.3.6. (Bonami and Lejeune [26]) *A probability distribution of an* r*-variate real-valued random vector* ξ *with mean* μ *and median* m *has positive skewness if*

$$\mathbb{P}(0 \geqslant \psi) \geqslant \mathbb{P}(m \geqslant \psi) \Longleftrightarrow F^{-1}_{(x)}(\alpha) \leqslant 0, \quad \alpha \leqslant 0.5,$$

where $\mathbb{E}[\psi] = \mathbb{E}[\xi - \mu] = 0$ *and* $F_{(x)}(m) = \mathbb{P}(m \geqslant \psi) = 0.5$.

Theorem 3.3.7. (Bonami and Lejeune [26]) *Let* $p \in [0.5, 1)$. *If the skewness of the probability distribution of* $\xi^\top x$ *is positive, then the deterministic constraint* (3.7b), *equivalent to* (3.5), *is a SOCC.*

The exact value of the $(1-p)$-quantile, $F_{(x)}(1-p)$, is known only for few probability distributions. If we assume, for example, that the distribution of the expected returns is Gaussian, which is a quite restrictive assumption (see, for example, [84, 85, 170, 210]), but rather common in several theoretical frameworks (see [112, 137, 194]), then the numerical values of quantiles $F^{-1}_{(x)}(1-p)$ of the normalized portfolio return ψ are computationally known.

3.4 ADDITIONAL CONSTRAINTS

Beyond the ill-conditioning of problem (3.1), the other serious drawback of Markowitz's original proposal is represented by the mismatch with the problems faced by practitioners in real-world applications (see, e.g., [60, 220]). Nevertheless, we can consider additional constraints to problems (3.1) or (3.4), which describe the most common restrictions observed in real-world financial markets (see [82, 146, 153, 162, 173]). However, this kind of constraints could make the efficient frontier discontinuous and more challenging to compute [133].

3.4.1 Buy-in Thresholds

Generally, investors avoid extremely small long positions in their portfolios, because, on one side, they have a limited impact on the return value of the portfolio and, on the other side, they could be quite expensive with respect to finance fees and monitoring costs [219]. Long positions not belonging to a given range $[\underline{x}_j, \overline{x}_j] \subset [0, 1]$ of the total initial budget B can be prevented by the simple range constraint

$$\underline{x}_j \leqslant x_j \leqslant \overline{x}_j, \qquad j = 1, \dots, r. \tag{3.9}$$

Several authors (see, e.g., [44, 49, 97, 133]) require x to be a semi-continuous variable [234], i.e., they require $x_j \in [\underline{x}_j, \overline{x}_j] \cup \{0\}$ for all $j = 1, \dots, r$: they introduce extra binary variables $\delta \in \{0, 1\}^r$ such that, for all $j = 1, \dots, r$, $\delta_j = 1$ if the investor holds the asset j, i.e. if $x_j > 0$, and add the following constraints avoiding too small or huge holding positions:

$$\underline{x}_j \delta_j \leqslant x_j \leqslant \overline{x}_j \delta_j, \qquad j = 1, \dots, r. \tag{3.10}$$

Note that constraints (3.10) directly imply $0 \leqslant x_j \leqslant \delta_j$ for all $j = 1, \dots, r$.

3.4.2 Round Lot Purchasing

Usually, investors manage only given lots of shares and other financial agreements, because of the facility in monitoring and purchasing/selling operations. Furthermore, for small private investors, splitting a large lot involves a premium, that has to be paid to the broker. The higher the cost of splitting large batch in single shares, the greater the impact of this kind of cost with respect to the optimal portfolios. Round lot purchasing constraints prescribe that investors hold, for the risky asset j ($j = 1, \dots, r$), batches or lots of S_j stocks.

Let us define $\gamma \in \mathbb{Z}_+^r$ as a vector of general integer variables. We require that the number of the shares of asset j ($j = 1, \dots, r$), namely $\eta_j \in \mathbb{Z}_+$, is an integer multiple of the lot-size S_j:

$$\eta_j = \gamma_j S_j, \qquad j = 1, \dots, r. \tag{3.11}$$

Let q_j be the market value of asset j ($j = 1, \dots, r$) held in portfolio, then we have $\eta_j = x_j B / q_j$ and constraint (3.12) can be equivalently rewritten as follows,

$$x_j = \frac{q_j \gamma_j S_j}{B}, \qquad j = 1, \dots, r. \tag{3.12}$$

The reader is referred to [26] for further discussion.

Mansini and Speranza [174] have shown that finding a feasible solution of problem (3.1) with round lot constraints (3.11), upper bound on γ_j, i.e., the number ($j = 1, \dots, r$) of minimum lots, and bound constraints with respect to the total portfolio expenditure is **NP**-complete.

3.4.3 Sector Diversification

Generally, either there exist law limitations about the risk exposure (this is the case, for instance, of pension funds) or investors try to hold a representative portion of their portfolio in a prescribed number of asset categories or industrial sectors. However, in general, optimal portfolios for problem (3.1) are not well-diversified [109]. Usually, given are lower bound on the fraction of portfolio value held in specific sets of shares. For classical empirical analysis about financial benefits of a well-diversified portfolio, we refer the reader to [56, 83, 231].

Let us assume that every asset can be allocated to a specific financial category and let C_k ($k = 1, \dots, n$) be the index set of all risky assets connected with the category k. Moreover, we suppose that sets C_k define a partition of $\{1, \dots, r\}$. We introduce a binary variable $\zeta_k \in \{0, 1\}$ for each financial category, such that $\zeta_k = 1$ if and only if the investment in financial category k ($k = 1, \dots, n$) is above a prescribed minimum level $\underline{s}$:

$$\underline{s}\zeta_k \leqslant \sum_{j \in C_k} x_j \leqslant \underline{s} + (1 - \underline{s})\zeta_k. \tag{3.13}$$

Moreover, we have to consider an additional constraint in order to satisfy the diversification prescription [26], which requires to hold portions of assets in at least $\underline{n} > 0$ categories:

$$\sum_{k=1}^{n} \zeta_k \geqslant \underline{n}. \tag{3.14}$$

3.4.4 Cardinality Constraints

Beyond diversification requirements, asset managers (for instance in index tracking funds) wish to replicate as accurately as possible a market index with a limited number of financial agreements, namely $\overline{K} > 0$. This can be modeled through the following cardinality constraint:

$$\|x\|_0 = \sum_{j=1}^{r} \operatorname{sgn}(|x_j|) \leqslant \overline{K}. \tag{3.15}$$

By introducing additional decision variables δ_j, already presented for constraints (3.10), we can straightforwardly reformulate the previous constraint in the following equivalent form [153]:

$$\sum_{j=1}^{r} \delta_j \leqslant \overline{K}. \tag{3.16}$$

Bienstock [20] (see also [226]) shows that problem (3.1) with cardinality constraint (3.16) is **NP**-hard, even when $r = 3$. Several authors (see, e.g., [49,

70, 87, 229, 243]) consider an equality version for cardinality constraint (3.16) and propose mainly heuristic methods to solve the corresponding problem:

$$\sum_{j=1}^{r} \delta_j = \overline{K}. \tag{3.17}$$

Moreover, finding the $\overline{K}$ assets that should be included in the optimal portfolio is, in general, an **NP**-hard problem [195].

Using the theoretical results in [224, 236] and extending [45, 46, 47], Cesarone et al. [44] have shown that the problem (3.1) with cardinality constraints (3.16) has the same optimal solution of problem (3.1) with equality cardinality constraints (3.17) and reduce this kind of programs to Standard Quadratic Programming Problem (see [23, 24]), avoiding to explicitly introduce binary variables and considering an exact tailored solving procedure, called Increasing Set Algorithm. The Standard Quadratic Programming Problem is an **NP**-hard problem when the Hessian matrix of the objective function is indefinite, i.e., if the Hessian matrix of the objective function is neither positive nor negative semidefinite [23].

Di Gaspero et al. [71] consider an "interval" version for the cardinality constraint (3.16): $\underline{K} \leqslant \sum_{j=1}^{r} \delta_j \leqslant \overline{K}$, where $\underline{K}$ and $\overline{K}$ are such that $1 \leqslant \underline{K} \leqslant \overline{K} \leqslant r$. Cardinality constraints are closely related to buy-in threshold constraints [133]. Finally, in several papers (see, e.g., [44, 49, 133]) it is observed that the problem (3.1), with cardinality constraints (3.16) and with minimum and maximum buy-in thresholds (3.10) can be straightforwardly reformulated as a convex MIQP.

3.4.5 Sector Capitalization

Sector capitalization constraints are introduced by Soleimani et al. [229], in order to mathematically formulate the behavior of investors generally inclined to hold assets in financial sectors with higher capitalization value to reduce the total portfolio risk.

Let ℓ be the number of economic sectors and suppose, without loss of generality, that they are sorted in non-increasing way according to their capitalization value. Define L_l as the set of assets for economic sector l ($l \in \{1, \dots, \ell\}$). We introduce additional binary variables y_l such that

$$\frac{1}{M} \sum_{j \in L_l} \delta_j \leqslant y_l \leqslant M \sum_{j \in L_l} \delta_j \qquad l \in \{1, \dots, \ell\} \tag{3.18a}$$

$$\sum_{j \in L_l} \overline{\mu}_j + (1 - y_l) \geqslant \sum_{j \in L_{l+1}} \overline{\mu}_j \qquad l \in \{1, \dots, \ell - 1\}, \tag{3.18b}$$

where $M \in \mathbb{R}_+$ is a sufficiently large positive number. The "big-M" constraints (3.18) ensure that the assets belonging to the sectors with higher

capitalization values have basically higher probability to be in the optimal portfolios than the ones belonging to sectors with less capitalization values.

3.4.6 Turnover and Trading

Frequently, investors already hold a portfolio $x^{(0)}$ and, because of mutations in the financial market or others, they want to change their portfolio, by considering the new financial environment and by limiting, however, the variations with respect to the portfolio already held [205].

Crama and Schyns [61] propose to introduce restrictions on purchasing and selling variations. In particular, let $\overline{P}_j$ and $\overline{S}_j$ be respectively the maximum purchasing and selling levels for asset j ($j = 1, \dots, r$), turnover constraints can be stated as follows:

$$\max\left\{x_j - x_j^{(0)}, 0\right\} \leqslant \overline{P}_j \qquad j = 1, \dots, r \tag{3.19a}$$

$$\max\left\{x_j^{(0)} - x_j, 0\right\} \leqslant \overline{S}_j \qquad j = 1, \dots, r. \tag{3.19b}$$

Because of fixed transaction costs (see Section 3.4.1), additional constraints are, generally, introduced in order to prevent small variations between portfolios. Let $\underline{P}_j$ and $\underline{S}_j$ be respectively the minimum purchasing and selling levels for asset j, trading disjunctive constraints can be stated as follows, $\left(x_j = x_j^{(0)}\right) \vee \left(x_j \leqslant x_j^{(0)} + \underline{P}_j\right) \vee \left(x_j \leqslant x_j^{(0)} - \underline{S}_j\right)$ for all $j = 1, \dots, r$.

3.4.7 Benchmark Constraints

Often, investors want to obtain a portfolio which is as close as possible to a benchmark (or target) portfolio x^B [17]. With respect to economic sector diversified investments, Bertsimas and Shioda [18] introduce the following additional constraints in order to bound variances between the optimal and the target portfolios:

$$\left|\sum_{j \in S_l} (x_j - x_j^B)\right| \leqslant \varepsilon_l \qquad l = 1, \dots, \ell. \tag{3.20}$$

3.4.8 Collateral Constraints

Di Gaspero et al. [73] (see also [127]) discuss the following legal constraints for short selling portfolios imposed by US Regulation T, a set of US laws concerning the margin requirements for the collateral agreement. The complete text of the regulation is available at `https://www.ecfr.gov/cgi-bin/text-idx?tpl=/ecfrbrowse/Title12/12cfr220_main_02.tpl`. In particular,

they introduce a free-risk asset with mean $\overline{\mu}_0$ and zero variance, the so-called collateral agreement, such that

$$x_0 \geqslant -a \sum_{j=1}^{r} \min\{0, x_j\} \tag{3.21a}$$

$$\sum_{j=0}^{r} |x_j| \leqslant 2 \tag{3.21b}$$

where $a \in \mathbb{N}_+$ is the security level for the collateral agreement. In this case the decision variables x are not constrained to be positive, since short-selling is allowed, and variables δ defined in (3.10) are replaced by ternary variables $z \in \{-1, 0, 1\}^r$, such that, for each j ($j = 1, \ldots, r$), $z_j = 1$ if the investor bought the asset j, i.e., if $x_j > 0$, $z_j = -1$ if the investor sold the asset j, i.e., if $x_j < 0$, and $z_j = 0$ if the investor does not hold asset j. Therefore, cardinality constraint (3.16) becomes $\sum_{j=1}^{r} |z_j| \leqslant \overline{K}$.

3.5 OBJECTIVE FUNCTIONS

Besides (3.1a), several different objective functions have been proposed in the literature in order to make problems (3.1) and (3.4) simpler with respect to computational tractability or to better model real behaviors of investors and money savers. We consider only objective functions involving quadratic risk measure, namely portfolio variance (for an exhaustive survey on approaches proposed for portfolio selection problem with linear risk measures we refer the interested reader to the paper [173] and to the recent book [172]).

3.5.1 Penalty Functions

In order to define an unconstrained NLP, Bartholomew-Biggs and Kane [8] introduce the following penalty function for problem (3.1) with minimum buy-in threshold constraints (3.9) with $\underline{x}_j := \underline{x}$ and $\overline{x}_j := 1$ for all $j \in \{1, \ldots, r\}$,

$$\phi(x_j) = \frac{4\, x_j (x_j - \underline{x})}{\underline{x}^2}, \qquad j = 1, \ldots, r \tag{3.22}$$

which is non negative when $x_j \leqslant 0$ or $x_j \geqslant \underline{x}$. Moreover, $-1 \leqslant \phi(x_j) < 0$ when $x_j \in (0, \underline{x})$, so that additional constraint (3.9) can be replaced by the following continuous one:

$$\phi(x_j) \geqslant 0, \qquad j = 1, \ldots, r. \tag{3.23}$$

Therefore, an unconstrained NLP can be easily defined, by introducing additional continuous variables $s \in \mathbb{R}^r$, such that $x_j := s_j^2$ for all $j = 1, \ldots, r$ and considering the resulting objective function, adjoint with three penalty terms, one replacing each set of constraints:

$$x^\top \overline{\Sigma} x + \rho(1 - \mathbf{1}_r{}^\top x)^2 + \rho\left(\frac{\overline{\mu}^\top x}{R} - 1\right)^2 + \tau \sum_{j=1}^{r} \kappa_j(x_j)^2, \tag{3.24}$$

where ρ and τ are suitable positive parameters and $\kappa_j(x_j) := \min\{0, \phi(x_j)\}$ for all $j = 1, \ldots, r$.

A similar approach is stated also for the round lot purchasing constraints (3.11) or (3.12), that can be replaced by the following constraints:

$$\kappa'_j(x_j) = \left(\frac{Bx_j}{q_j} - \left\lfloor \frac{Bx_j}{q_j} \right\rfloor\right)\left(1 - \left(\frac{Bx_j}{q_j} - \left\lfloor \frac{Bx_j}{q_j} \right\rfloor\right)\right) = 0, \qquad j = 1, \ldots, r \tag{3.25}$$

where $\lfloor \nu \rfloor$ denotes the integer part of $\nu \in \mathbb{R}$.

However, round lot purchasing constraints (3.11) might make impossible satisfy at the same time request (3.1c): consequently, the following new quadratic risk measure [192] is considered:

$$\frac{x^\top \overline{\Sigma} x}{(\mathbf{1}_r{}^\top x)^2}, \tag{3.26}$$

leading to an alternative definition of (3.24):

$$\frac{x^\top \overline{\Sigma} x}{(\mathbf{1}_r{}^\top x)^2} + \rho\big(\min\{0, 1 - \mathbf{1}_r{}^\top x\}\big)^2 + \rho\left(\frac{\overline{\mu}^\top x}{R} - 1\right)^2 + \tau \sum_{j=1}^{r} \kappa'_j(x_j)^2. \tag{3.27}$$

Bartholomew-Biggs and Kane [8] apply a DIRECT (DIviding RECTangles) type global algorithm (see [91, 99, 100, 134, 135]) to the previous unconstrained NLPs (3.24) and (3.27).

3.5.2 Balanced Objective Functions

Mean-variance portfolio selection problems (3.1) and (3.4) are naturally multi-objective optimization programs since usually investors want to gain the maximum profit at the minimum risk: these are, of course, conflicting targets, that have to be considered at the same time.

Several authors (see, e.g., [49]) use standard (linear) scalarization techniques such as the Weighted Sum approach (see, for example, [76]). Namely, they consider the "balanced" objective function

$$\lambda(x^\top \overline{\Sigma} x) - (1 - \lambda)(\overline{\mu}^\top x), \tag{3.28}$$

where $\lambda \in [0, 1]$ is a parameter which represents investor's risk aversion. Let $\theta_1, \theta_2 \in \mathbb{R}_+$ be two parameters, a more general variant is proposed by Schaerf [218]:

$$\theta_1 (x^\top \overline{\Sigma} x) + \theta_2 \max\{0, \overline{\mu}^\top x - R\}. \tag{3.29}$$

Bertsimas and Shioda [18] (see also [17]) introduce an extended "balanced" objective function, considering also trading and turnover requirements with respect to a given initial portfolio $x^{(0)}$:

$$\frac{1}{2} x^\top \overline{\Sigma} x - \overline{\mu}^\top x + \sum_{j=1}^{r} \iota_j \left(x_j - x_j^{(0)}\right)^2, \tag{3.30}$$

where $\iota_j > 0$ is a coefficient for asset j which models the symmetric purchasing/selling impact with respect to the stock price. Finally, Tadonki and Vial [235] and Shaw et al. [226] consider respectively constant and linear transaction costs embedded in a quadratic "balanced" objective function, namely respectively

$$\lambda_1 (x^\top \overline{\Sigma} x) - \lambda_2 (\overline{\mu}^\top x) + c^\top \delta \tag{3.31}$$

$$\lambda_1 (x^\top \overline{\Sigma} x) - \lambda_2 (\overline{\mu}^\top x) + c^\top x, \tag{3.32}$$

where $\lambda_1 \in \mathbb{R}_+$ and $\lambda_2 \in \mathbb{R}_+$ are two positive scalars, $c \in \mathbb{R}^r_+$ is a vector, whose entries represent the transaction costs for the portfolio assets and $\delta \in \mathbb{R}^r$ is the binary vector defined in constraints (3.10).

3.6 COMPACT REFORMULATIONS

In this section we present several different possible reformulations and approximations for the mean-variance portfolio optimization problem.

3.6.1 SOCC Inner Approximations

As observed in Section 3.3.2, given a probability distribution on the portfolio returns, it is not always possible to write the problem (3.7) in a closed form: the exact value for the quantile $F^{-1}_{(x)}(1-p)$ is known only for special distributions (e.g., normal distribution, Student distribution, uniform distribution on an ellipsoid). However, if the probability distribution of the expected returns is only partially known, the value of its quantiles can be approximately computed using several well-known probability inequalities [153], such as, e.g., Cantelli [26], Chebyshev [26], and Camp-Meidell [152] inequalities (see also [113, 125, 161, 181]).

Theorem 3.6.1. (Bonami and Lejeune [26]) *Assume the first and the second moment of the probability distribution of the portfolio return are finite. The SOCC*

$$\mu^\top x - \sqrt{\frac{p}{1-p}}\sqrt{x^\top \Sigma x} \geqslant R \tag{3.33}$$

is an approximation of the chance constraint (3.4b).

Theorem 3.6.2. (Bonami and Lejeune [26]) *Assume the first and the second moment of the probability distribution of the portfolio return are finite and the distribution is symmetric. The SOCC*

$$\mu^\top x - \sqrt{\frac{1}{2(1-p)}}\sqrt{x^\top \Sigma x} \geqslant R \tag{3.34}$$

is an approximation of the chance constraint (3.4b).

Theorem 3.6.3. (Lejeune [152]) *Assume the first and the second moment of the probability distribution of the portfolio return are finite and the distribution is symmetric and unimodal. The SOCC*

$$\mu^\top x - \sqrt{\frac{1}{9\,(1-p)}}\,\sqrt{x^\top \Sigma x} \geqslant R \tag{3.35}$$

is an approximation of the chance constraint (3.4b).

Remark 3.6.4. *(Bonami and Lejeune [26]) The approximation given by Theorem 3.6.2 for a symmetric probability distribution is tighter than the one given by Theorem 3.6.1 and that the approximation given by Theorem 3.6.3 for a symmetric unimodal probability distribution is tighter than the one given by Theorem 3.6.2.*

3.6.2 Variance Reformulation

Given the symmetric positive definite matrix Σ, we consider its Cholesky decomposition $\Sigma = CC^\top$, where $C \in \mathbb{R}^{r\times r}$ is a lower triangular matrix. From a computational viewpoint, the Cholesky decomposition is twice faster and more stable than LU factorization or Gauss elimination method (see [143, 184, 233]) and it is implemented in High Performance Computing numerical software libraries (see [2, 21, 75]).

Note that Cholesky decomposition exists and is unique if matrix Σ is positive definite (see [107, 114]) and this property is verified by variance-covariance matrix, if we exclude the case of exact collinearity of the random variables, i.e., we assume that none of the risky asset can be exactly replicated by a linear combination of the other ones. The hypotheses to apply Cholesky decomposition to positive semidefinitive matrices are identified in [116, 114, 193] and error analysis is instead formally stated in [193] for idempotent matrices and in [116] for the general case.

By assuming positive definiteness for covariance matrix Σ and introducing non negative decision variable $h \in \mathbb{R}_+$, we obtain the following problem, equivalent to (3.7):

$$\min_{x,h} \ \|C^\top x\|_2^2 \tag{3.36a}$$
$$\text{s.t. } \mu^\top x - R \geqslant h \tag{3.36b}$$
$$F_{(x)}^{-1}\,(1-p)\,\|C^\top x\|_2 \geqslant -h \tag{3.36c}$$
$$\mathbf{1}_r{}^\top x = 1 \tag{3.36d}$$
$$x \geqslant 0, \quad h \geqslant 0. \tag{3.36e}$$

Theorem 3.6.5. (Filomena and Lejeune [89]) *Program (3.36) is equivalent to the following NLP:*

$$\begin{aligned} &\min_{x,h} \ h \\ &\text{s.t. } (3.36b), (3.36c), (3.36d), (3.36e). \end{aligned} \tag{3.37}$$

3.6.3 Period-separable Reformulation

As pointed out by Filomena and Lejeune [89], the variance of the portfolio can be reformulated as the Euclidean norm of a vector, whose number of components T corresponds to the number of data points, by using the following preliminary result:

Theorem 3.6.6. (Konno and Suzuki [147]) *Let ν_{jt} be the (observed) return of asset j at time t and introduce the extra variables $b_t = \sum_{j=1}^{r}(\nu_{jt} - \mu_j)x_j$ $(t = 1, \dots, T)$. The variance of the portfolio return can be rewritten as*

$$x^\top \Sigma x = \frac{1}{T}\|b\|_2^2.$$

Therefore, the probabilistic Markowitz portfolio model (3.4) can be reformulated as the following convex NLP:

$$\min_{x,h,b} \ \frac{1}{T}\|b\|_2^2 \tag{3.38a}$$

$$\text{s.t. } \mu^\top x - R \geqslant h \tag{3.38b}$$

$$\frac{F_{(x)}^{-1}(1-p)}{\sqrt{T}}\|b\|_2 \geqslant -h \tag{3.38c}$$

$$b_t - \sum_{j=1}^{r}(\nu_{jt} - \mu_j)\,x_j = 0, \quad t = 1, \dots, T \tag{3.38d}$$

$$\mathbf{1}_r{}^\top x = 1 \tag{3.38e}$$

$$x \geqslant 0, \quad h \geqslant 0. \tag{3.38f}$$

Furthermore, Filomena and Lejeune [89] observe that in order to mathematically compute the variance in problems (3.4) and (3.37) the estimate of only $r(r+1)/2$ covariance terms is needed: this situation can lead to several coherence problems for the covariance matrix (see Section 3.1). Moreover, the approach described in this section does not require any assumption on matrix Σ. Finally, we can consider the corresponding equivalent epigraph formulation of problem (3.38):

$$\min_{x,h,b} \ h$$

$$\text{s.t. } (3.38b), (3.38c), (3.38d), (3.38e), (3.38f).$$

3.7 EXACT ALGORITHMS

Mean-variance portfolio selection problem with the constraints introduced in Section 3.4 gives rise to a convex MIQP, which is at least as difficult as **NP**-hard, because it includes MILP as special case [139, 102]. Nowadays, MIQPs can be solved via commercial and open-source solvers (see, e.g., [25, 62, 239]). In this section we overview specialized and more efficient computational procedures recently proposed in literature.

In [20] Bienstock proposes a tailored Branch-and-Cut (BC) procedure to solve the cardinality constrained portfolio problem, where (3.16) is replaced with the "surrogate" constraint

$$\sum_{j=1}^{r} \frac{x_j}{\overline{x}_j} \leqslant \overline{K}. \tag{3.39}$$

Several types of cutting planes, namely mixed-integer rounding inequalities, knapsack cuts, intersection cuts, and disjunctive cuts are also considered in the same paper.

Bertsimas and Shioda [18] develop a BC algorithm where at each node of Branch-and-Bound (BB) tree the convex continuous relaxation of problem (3.1) with cardinality (3.16) and buy-in (3.10) constraints is solved by means of Lemke's method [58]. The portfolio problem with objective function (3.31) and cardinality (3.16) and buy-in (3.10) constraints was solved by Tadonki and Vial [235] with BB techniques together with a Bender decomposition approach.

Lee and Mitchell [151] describe a parallel BB framework for the cardinality constrained portfolio selection problem, in which each node is approximated by means of Sequential Quadratic Programming (SQP) and each quadratic subproblem is solved via interior-point method (see, e.g., [199, 200]).

Frangioni and Gentile [97] solve problem (3.1) with minimum and maximum buy-in thresholds additional constraints (3.10) with a BC method improved by using Perspective Cuts (see also [96]), a family of valid inequalities, related to the perspective function (see [118, 119]) and to the convex envelope of the objective function (see [110]).

Zheng et al. [251] propose a difference of convex functions approach to the cardinality constrained quadratic program, by replacing the cardinality constraint (3.15) with the following piecewise linear approximation:

$$\frac{1}{\omega}\left(\|x\|_1 - \sum_{j=1}^{r} \max\{x_j - \omega, 0\} + \max\{-x_j - \omega, 0\}\right) \leqslant 0, \tag{3.40}$$

where $\omega > 0$ is a given parameter. Non-smooth approximation (3.1) with constraint (3.40) is solved by means of Successive Convex Approximation method. This algorithm determines a Karush-Kuhn-Tucker (KKT) point or defines a sequence of points converging to a KKT point for the ω-parametrized approximation. Moreover, the authors show that, letting $\omega \to 0^+$, the optimal value of the approximate problem approaches the optimal value of the original problem.

Shaw et al. [226] solve cardinality constrained portfolio problem under the assumption that vector $\overline{\mu}$ of assets returns can be decomposed according to

a multiple factor model [41], i.e., $\overline{\mu} = \Xi f + u$, where r' represents the number of different factors, $\Xi \in \mathbb{R}^{r \times r'}$ is the sensitivity-factor matrix, $f \in \mathbb{R}^{r'}$ is the factor-return vector, and $u \in \mathbb{R}^{r}$ is the asset-specific (non-factor) returns vector. A Lagrangian relaxation of the problem is then solved by means of sub-gradient procedure [211] and embedded in a BB framework.

In [26] Bonami and Lejeune deal with the deterministic equivalent (3.7) of the probabilistic portfolio selection problem with buy-in threshold (3.10), round lot purchasing (3.12), and diversification (3.13)-(3.14) constraints, by proposing a Nonlinear BB algorithm [11] with two specific branching rules:

1. Idiosyncratic Risk Branching, consisting in selecting the fractional variable δ_j or γ_j, which corresponds to the asset with the highest expected return;

2. Portfolio Risk Branching: consisting in selecting the fractional variable δ_j or γ_j, whose integer fixing has the highest impact on the objective function (3.1a).

Buchheim et al. [37] consider portfolio selection problem with objective function (3.28), constraints (3.1d) and (3.3) and integrality requirement on the decision variables, i.e.,

$$x \in \mathbb{Z}^{r}, \tag{3.41}$$

which represents the units of assets held in the investor's portfolio. They introduce a new BB algorithm where the continuous relaxation is solved through an efficient Frank-Wolfe type method with non-monotone Armijo line-search.

Burdakov et al. [38] deal with the cardinality constrained portfolio problem, by introducing a NLP reformulation, whose global minima are the same of the ones of the original problem. The NLP is solved via a sequence of regularized programs (see [140]).

We end this section with Table 1 that summarizes the main characteristics of the papers described above. In particular, the columns report the authors, the year of publication of the paper, the objective function and constraints of the tackled problem, the proposed algorithm, the competitors employed as benchmarks, and the instances that were used for the computational experiments.

1 Available at URL http://miplib.zib.de/
2 Available at URL http://people.brunel.ac.uk/~mastjjb/jeb/orlib/portinfo.html
3 Available at URL http://www.di.unipi.it/optimize/Data/MV.html
4 Available at URL http://w3.uniroma1.it/tardella/datasets.html

Author(s)	Year	Objective	Constraints	Algorithm	Benchmark(s)	Instances (number and type)
[20]	1996	(3.1a)	(3.1c), (3.1b), (3.1d), (3.9), (3.16)	Branch-and-Cut	–	13 problems (real-life data)
[151]	2000	(3.1a)	(3.1c), (3.1b), (3.1d), (3.16)	Branch-and-Bound	–	5 portfolio instances and 16 instances from MIPLIB[1]
[235]	2003	(3.31)	(3.1c), (3.1b), (3.1d), (3.10), (3.16)	Branch-and-Bound	–	OR-Library[2] [9, 10]
[97]	2006	(3.1a)	(3.1c), (3.1b), (3.1d), (3.10)	Branch-and-Cut	CPLEX 8.0	20 self-generated[3] (see [203])
[226]	2008	(3.32)	(3.9), (3.16)	Branch-and-Bound	CPLEX 8.1	8 real representative instances
[8]	2009	(3.24)	–	DIRECT	–	2 illustrative examples
[8]	2009	(3.27)	–	DIRECT	–	2 illustrative examples
[18]	2009	(3.30)	(3.1c), (3.1d), (3.9), (3.16), (3.20)	Branch-and-Bound	CPLEX 8.1	50 randomly generated
[26]	2009	(3.1a)	(3.1c), (3.1d), (3.7b), (3.10) (3.12), (3.13), (3.14), (3.41)	Branch-and-Bound	[27], CPLEX 10.1 MINLP_BB [155]	36 self-generated instances
[251]	2012	(3.1a)	(3.1c), (3.1b), (3.1d), (3.9), (3.15)	SQA Algorithm	[167]	[97]
[44]	2013	(3.1a)	(3.1c), (3.1b), (3.1d), (3.10), (3.16)	Increasing Set	[72], [195], [216], [218]	[49] 5 additional data sets[4]
[252]	2013	(3.1a)	(3.1c), (3.1b), (3.1d), (3.10), (3.16)	Branch-and-Cut	–	[97] OR-Library[2] [9, 10]
[37]	2015	(3.28)	(3.1d), (3.3), (3.41)	Branch-and-Bound with Frank-Wolfe	CPLEX 12.6	[44]
[38]	2016	(3.1a)	(3.1c), (3.1d), (3.2), (3.9), (3.16)	Regularization Method	GUROBI 5.6.2	[97]

Table 1: Deterministic exact approaches to mean-variance portfolio selection problem (see also [173] and references within). References are sorted in chronological order; papers published in the same year are sorted according to alphabetic order of the last name of the corresponding first author.

3.8 MWU FOR A CLASS OF MVPS PROBLEMS

The portfolio problem we are interested in can be formulated as follows. Let $x \in \mathbb{R}^r_+$ be the decision variables representing the fraction of asset $j \in \{1, \dots, r\}$ held in the portfolio. Each asset j is characterized by a mean expected return $\overline{\mu}_j$ ($j \in \{1, \dots, r\}$) and a non negative, possibly non-convex non-concave, cost function $C_j(x_j) : \mathbb{R}_+ \to \mathbb{R}_+$ ($j \in \{1, \dots, r\}$). The goal consists in maximizing the overall return of the portfolio in presence of cardinality constraint, risk constraint, and no-short-selling constraint. The risk of the portfolio is represented by its total variance $x^\top \overline{\Sigma} x$, where $\overline{\Sigma} \in \mathbb{R}^{r \times r}$ is the covariance matrix of the return distribution of the assets. The portfolio problem reads:

$$\begin{aligned}
\max \ & \overline{\mu}^\top x - C(x) && \text{(3.42a)}\\
\text{s.t. } & \mathbf{1}_r{}^\top x = 1 && \text{(3.42b)}\\
& x^\top \overline{\Sigma} x \leqslant \sigma && \text{(3.42c)}\\
& \|x\|_0 \leqslant K && \text{(3.42d)}\\
& x \in [\underline{x}, \overline{x}] \cup \{0\}, && \text{(3.42e)}
\end{aligned}$$

where $C(x) := \sum_{j=1}^r C_j(x_j)$, and σ and K are the maximum levels of risk and sparsity we aim for our portfolio, respectively.

Remark 3.8.1. *We assume both the mean vector $\overline{\mu}$ and the covariance matrix $\overline{\Sigma}$ are perfectly known or coherently estimated from historical data measurements.* □

Problem (3.42) generalizes the usual portfolio problem with concave transaction costs already considered in the literature (see, e.g., Konno and Wijayanayake [148] and Xue et al. [245]). To the best of our knowledge, this is the first time this kind of portfolio problems is addressed in its entirely.

The previous MVPS problem is **NP**-hard (see Section 3.4.4) and can be exactly reformulated, in the sense explained in [157], by introducing extra binary variables $\delta \in \{0, 1\}^r$ as follows:

$$\begin{aligned}
\max \ & \overline{\mu}^\top x - C(x) && \text{(3.43a)}\\
\text{s.t. } & \mathbf{1}_r{}^\top x = 1 && \text{(3.43b)}\\
& x^\top \overline{\Sigma} x \leqslant \sigma && \text{(3.43c)}\\
& \mathbf{1}_r{}^\top \delta \leqslant K && \text{(3.43d)}\\
& \delta^\top \underline{x} \leqslant x \leqslant \delta^\top \overline{x} && \text{(3.43e)}\\
& \delta \in \{0, 1\}^r. && \text{(3.43f)}
\end{aligned}$$

3.8.1 Pointwise Reformulation

A pointwise reformulation for problem (3.43) can be obtained by replacing each function $C_j(x_j)$ with the term $(1 + x_j)\,\theta_j$ parametrized by θ_j. The only

part of the problem, where the substituting terms appear, is the objective function:

$$\overline{\mu}^{\top} x - \sum_{j=1}^{r} (1 + x_j)\, \theta_j = (\overline{\mu} - \theta)^{\top} x - \mathbf{1}_r{}^{\top} \theta, \tag{3.44}$$

which becomes an affine function of the continuous decision variables x_j ($j \in \{1, \ldots, r\}$). Thus, the pointwise reformulation is:

$$-\mathbf{1}_r{}^{\top} \theta + \max \ (\overline{\mu} - \theta)^{\top} x \tag{3.45a}$$

$$\text{s.t. } \mathbf{1}_r{}^{\top} x = 1 \tag{3.45b}$$

$$x^{\top} \overline{\Sigma} x \leqslant \sigma \tag{3.45c}$$

$$\mathbf{1}_r{}^{\top} \delta \leqslant K \tag{3.45d}$$

$$\delta^{\top} \underline{x} \leqslant x \leqslant \delta^{\top} \overline{x}. \tag{3.45e}$$

Remark 3.8.2. *The pointwise reformulation* (3.45) *is spanning: the replacement terms* θ_j *perfectly matches the replaced terms* $\frac{C_j(x_j)}{x_j+1}$ *at each feasible point* x_j ($j \in \{1, \ldots, r\}$). *By Lemma 2.2.6, there exist values of* θ *which make* (3.45) *a bounding reformulation for the original problem* (3.42). □

Example 3.8.3. *The pointwise reformulation* (3.45) *is* not *exact. For instance, we consider a portfolio problem with* $r = 2$ *stocks,* $\overline{\Sigma} = I_2$, $K = 2$, $\underline{x} = 0$, $\overline{x} = 0.55$ *and* $\sigma = r$, *i.e., an essentially unconstrained portfolio problem, apart from the budget constrained* $x_1 + x_2 = 1$. *Let* $\overline{\mu} = (1, 0.40)^{\top}$ *and* $C(x) = (C_1(x_1), C_2(x_2))^{\top} = (x_1, 0)^{\top}$. *For all* $j \in \{1, 2\}$, $\theta_j = \frac{C_j}{1+x_j}$ *and, hence,* $\Theta_j = [C_j(\underline{x})/(1+\overline{x}), C_j(\overline{x})/(1+\underline{x})]$. *For the example, we have* $\Theta_1 = [0, 0.55]$ *and* $\Theta_2 = \{0\}$. *The objective function of the original formulation* (3.42) *is simply* $\max(0.40\, x_2)$ *and its global optimum is* $x^* = (0.45, 0.55)^{\top}$. *The objective function of the pointwise reformulation is* $\max((1 - \theta_1)\, x_1 + 0.50\, x_2 - \theta_1)$. *Now,* $x^* = (0.45, 0.55)^{\top}$ *is a global optimum of the pointwise reformulation if and only if* $(1 - \theta_1) \leqslant 0.40$, *i.e., if and only if* $\theta_1 > 0.60$: *however, these values of* θ_1 *do not belong to the set* Θ_1. □

Remark 3.8.4. *The pointwise reformulation* (3.45) *is* not *efficient (see Section 3.4.4). However, since convex MIQPs can be solved nowadays with reasonable efficiency, certainly more efficiently than non-convex non-concave MINLPs, the proposed reformulation is* good.

The previous approach can be easily extended to cardinality constrained portfolio selection problem with cost functions and fixed transaction costs, i.e., when a term $\sum_{j=1}^{r} c_j \delta_j$ is added to the objective function: in these problems the investor has to pay a fixed amount c_j ($j \in \{1, \ldots, r\}$) of money if he/she decided to buy a certain asset j ($j \in \{1, \ldots, r\}$). Moreover, since the previous strategy consists essentially in substituting each single term in the separable function with an affine function depending on a given parameter, it can be extended to general MINLPs with separable non-convexities and non-concavities.

3.8.2 Computing MWU Costs/Gains

Since the substituting terms appear only in the objective function, we do not need to address feasibility issues in computing MWU costs/gains. In particular, we define, at each iteration $t \leqslant T$, a r-dimensional vector $\alpha_t \in [-1,1]^r$ whose components are:

$$\alpha_{j,t} = \frac{C_j(x_{j,t}) - (x_{j,t}+1)\,\theta_{j,t}}{\max(|C_j(x_{j,t})|,\, |(x_{j,t}+1)\,\theta_{j,t}|)} \qquad (j \in \{1,\dots,r\}, t \leqslant T). \tag{3.46}$$

Remark 3.8.5. *Each component $\alpha_{j,t}$ represents the scaled cost/gain determined by each asset j ($j \in \{1,\dots,r\}$). In other words, each $\alpha_{j,t}$ takes into account the contribution of the fraction of each asset held in the portfolio to the overall cost function.* □

Remark 3.8.6. *We define α_t to be a vector, instead of a scalar as described in Section 2.4.3, because this definition is* more effective *in presence of separable non-convexities and non-concavities: it allows us to better follow the numerical behavior of each single cost function $C_j(x_j)$ for all $j \in \{1,\dots,r\}$*

In order to define the costs/gains, we simply set:

$$\psi_{j,t} := \alpha_{j,t} \qquad (j \in \{1,\dots,r\}, t \leqslant T). \tag{3.47}$$

3.8.3 Computational Experiments

The test-bed set with respect to which we analyze our algorithm and the benchmarks consists in the 20 real-world instances described by Chang et al. [49] publicly available through the OR-Library (see Beasley [9] and Beasley [10]) on the web site `http://www.brunel.ac.uk/~mastjjb/jeb/info.html`. Each instance is characterized by the number n of assets and the value of the risk level σ. We impose, as in Chang et al. [49], $\underline{x} = 0.01$, $\overline{x} = 1$ and $K = 10$.

Transaction Cost Functions

One of the aims of the computational experiments, in addition to analyze the behavior of the MWU algorithm against other methods to solve MVPS problems, consists in empirically evaluating the performances of the MWU algorithm depending from the *nonlinearity* of the replaced transaction cost functions. In particular, we consider the following five univariate functions (see Figure 1):

(a) $C_j(x_j) = -\overline{\mu}_j \ln\left(\frac{20-0.06(1+x_j)}{1+x_j}\right)$ for all $j \in \{1,\dots,r\}$: this cost function is increasing, concave and "almost linear".

(b) $C_j(x_j) = -\overline{\mu}_j \ln\left(\frac{0.2-0.01(0.00001+x_j)}{0.00001+x_j}\right)$ for all $j \in \{1,\dots,r\}$: this cost function is increasing, concave, and replicates the behavior of the transaction cost function described in Konno and Wijayanayake [148].

(c) $C_j(x_j) = \overline{\mu}_j(4x_j + 0.12\sin(40x_j))$ for all $j \in \{1, \dots, r\}$: this cost function has a sinusoidal behavior similar to a step function.

(d) $C_j(x_j) = \overline{\mu}_j(4x_j + 0.3\sin(40x_j))$ for all $j \in \{1, \dots, r\}$: this cost function is similar to the one in (c) but with a "stronger nonlinear behavior".

(e) $C_j(x_j) = \overline{\mu}_j(0.5x_j + \sin(50x_j))$ for all $j \in \{1, \dots, r\}$: this is the "most nonlinear" transaction cost function among which we tested the methods.

Remark 3.8.7. *We want to emphasize that the definition of* nonlinearity *of a given transaction cost function was not given in rigorous mathematical terms, but it is mostly qualitative: in fact, we carefully use quotes around each* nonlinearity *characterization. However, since our aim consists in empirically analyzing the behavior of the MWU algorithm with respect to a given cost function, we trust most readers will agree with our categorization, by inspection of Figure 1.*

Computational Environment

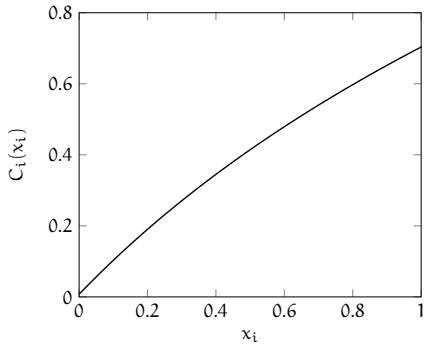
Since for the MVPS problem we consider in this section there are no tailored exact or heuristic methods to solve it and since the MWU algorithm is essentially a MS algorithm with a special strategy to choose the more promising initial points both in terms of feasibility and optimality, we compare the MWU with the MS.

We used $T = 20$ iterations for both MWU and MS. We adopted Bonmin [29] as local MINLP solver: since Bonmin exactly solve convex MINLPs, it is a reliable heuristic for non-convex MINLPs. In particular, we employ Bonmin's native Branch-and-Bound (B-BB) algorithm (see Bonami et al. [28] and Gupta and Ravindran [111]), since it is generally more stable for non-convex MINLPs. We used Cplex [124] as the convex `MIQP` solver for the pointwise reformulation (3.45), with a 600 seconds time limit, using only one thread. All of the computational experiments were performed on an Intel Xeon CPU E5649, 2.53 GHz, using only one processor.

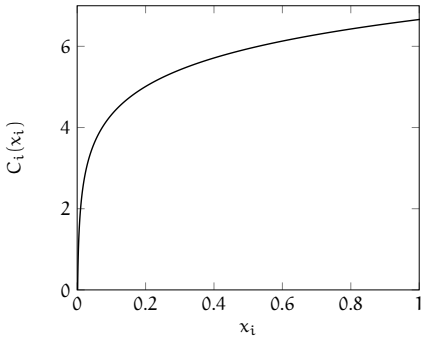
Localization of the MS Subsolver

Since we want to compare MWU against MS and MS is essentially a procedure composed by two steps: a random choice of a starting point, and a local descend method, we would like that Bonmin BB-B behaves like a local solver. From preliminary tests, however, the behavior of Bonmin BB-B was mostly similar to a global solver: Bonmin BB-B sets a cut-off value for the optimum based on the starting point and for our test-bed it found almost always the same solution point. In order to turn Bonmin B-BB into a truly local solver, we consider an adding local branching constraint (see Fischetti and Lodi [92]) for the original formulation, which basically defines an upper bound $\lfloor \nu n \rfloor$, where $\nu \in [0, 1]$, on the number of flips of binary variables δ:

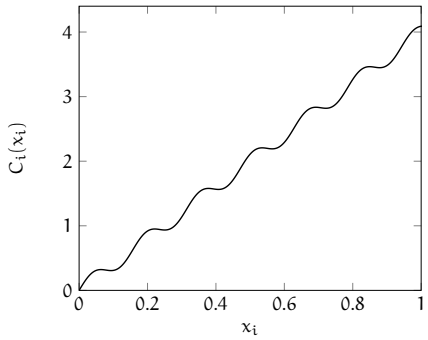
$$\sum_{\substack{i \leqslant r \\ \delta'_j = 0}} \delta_j + \sum_{\substack{j \leqslant r \\ \delta'_j = 1}} (1 - \delta_j) \leqslant \lfloor \nu\, n \rfloor, \tag{3.48}$$



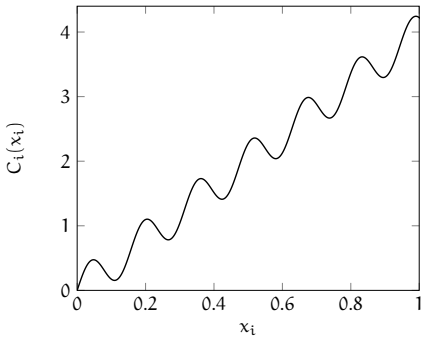
(a) Easy concave transaction costs.



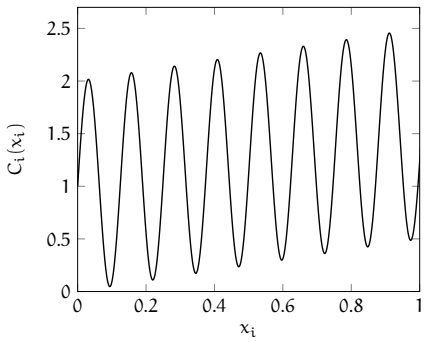
(b) Hard concave transaction costs.



(c) Easy trigonometric transaction costs.



(d) Medium trigonometric transaction costs.



(e) Hard trigonometric transaction costs.

Figure 1: Examples of transaction cost functions.

where δ' is the starting point. The spirit of the constraints (3.48) consists in enforcing a local exploration in the combinatorial neighborhood of the starting point δ'. After several computational experiments we decided to set $\nu = 0.96$, since lower values made the instance infeasible excessively often.

Tables 2-6 report the computational results for each transaction cost function. Their columns are as follows:

- instance name;
- maximum risk level σ;
- number r of assets quoted on the financial market;
- objective value for the MWU algorithm;
- CPU time (in seconds) for the MWU algorithm;
- objective value for the MWU algorithm with the local branching constraint;
- CPU time (in seconds) for the MWU algorithm with the local branching constraint;
- objective value for the MS algorithm with the local branching constraint;
- CPU time (in seconds) for the MS algorithm with the local branching constraint;
- relative objective value improvement from MS to MWU computed as

$$\Gamma = \frac{\text{val(MWU)} - \text{val(MS)}}{|\text{val(MS)}|}; \tag{3.49}$$

- time improvement ratio from MS to MWU:

$$\Lambda = \frac{\text{cpu(MS)}}{\text{cpu(MWU)}}; \tag{3.50}$$

- relative objective value improvement from MS to MWU with the local branching constraint (see Equation (3.49));
- time improvement ratio Λ from MS to MWU with the local branching constraint (see Equation (3.50)).

Computational Results

The comparison metrics are summarized in the last three lines with the sum ($\sum$), average (avg), and the standard deviation (std) across all 20 instances. For the CPU time we reported also the geometrical mean among all the instances.

			MWU			MWU+LB			MS			MWU vs. MS		MWU+LB vs. MS	
Instance	σ	n	objective	CPU	$\|x^*\|_0$	objective	CPU	$\|x^*\|_0$	objective	CPU	$\|x^*\|_0$	Γ	Λ	Γ	Λ
port-orlib1_000	0.0047755010	31	**1.0742596**	292.556	**1**	**1.0742596**	314.918	**1**	**1.0742596**	**31.203**	**1**	0.000%	0.107	0.000%	0.099
port-orlib1_050	0.0021522075	31	**1.0742596**	268.278	**1**	**1.0742596**	212.915	**1**	**1.0742596**	**31.469**	**1**	0.000%	0.117	0.000%	0.148
port-orlib1_100	0.0010585969	31	**1.0592082**	13.587	**3**	**1.0592082**	**12.915**	**3**	**1.0592082**	13.180	**3**	0.000%	0.970	0.000%	1.021
port-orlib1_150	0.0007158421	31	**1.0180977**	11.998	**5**	**1.0180977**	12.391	**5**	**1.0180977**	**11.764**	**5**	0.000%	0.980	0.000%	0.949
port-orlib2_000	0.0028352430	85	**1.8750415**	16000.371	**1**	**1.8750415**	18516.395	**1**	**1.8750415**	**1339.651**	**1**	0.000%	0.084	0.000%	0.072
port-orlib2_050	0.0004953237	85	**1.8741070**	330.941	**3**	**1.8741070**	196.865	**3**	**1.8741070**	**182.681**	**3**	0.000%	0.552	0.000%	0.928
port-orlib2_100	0.0002704062	85	**1.8587348**	7038.599	**5**	**1.8587348**	**5745.031**	**5**	**1.8587348**	7183.440	**5**	0.000%	1.021	0.000%	1.250
port-orlib2_150	0.0001663200	85	**1.7828905**	**214.691**	**9**	**1.7828905**	384.026	**9**	**1.7828905**	425.750	**9**	0.000%	1.983	0.000%	1.109
port-orlib3_000	0.0015166351	89	**2.3898574**	18019.824	**1**	**2.3898574**	17862.785	**1**	**2.3898574**	**1396.067**	**1**	0.000%	0.077	0.000%	0.078
port-orlib3_050	0.0005849758	89	2.3892174	2831.204	**2**	**2.3892246**	**213.402**	**2**	2.3892174	333.062	**2**	0.000%	0.118	0.000%	1.561
port-orlib3_100	0.0003215941	89	2.3622650	743.189	8	2.3622650	**173.656**	8	**2.3645800**	490.434	**7**	-0.098%	0.660	-0.098%	2.824
port-orlib3_150	0.0002239117	89	2.2652696	**3329.717**	**10**	2.2645169	4405.874	**10**	**2.2677086**	3238.561	**10**	-0.108%	0.973	-0.141%	0.735
port-orlib4_000	0.0029387241	98	2.9234597	29875.539	**1**	2.9234597	26538.931	**1**	**2.9234600**	**4777.167**	**1**	0.000%	0.160	0.000%	0.180
port-orlib4_050	0.0006828450	98	**2.9200045**	210.979	**3**	**2.9200045**	**192.266**	**3**	**2.9200045**	742.838	**3**	0.000%	3.521	0.000%	3.864
port-orlib4_100	0.0003059553	98	2.9067690	6997.871	5	2.9065884	**6364.984**	**4**	**2.9073338**	11884.665	5	-0.019%	1.698	-0.026%	1.867
port-orlib4_150	0.0001613979	98	**2.8526192**	895.863	10	**2.8526192**	**754.278**	10	**2.8526192**	6461.036	10	0.000%	7.212	0.000%	8.566
port-orlib5_000	0.0016485224	225	**4.6677119**	**3478.872**	**2**	**4.6677119**	5288.969	2	4.6673823	5736.267	4	0.007%	1.649	0.007%	1.085
port-orlib5_050	0.0005150277	225	**4.6668554**	5652.420	**3**	**4.6668554**	**4949.565**	**3**	**4.6668554**	10932.209	**3**	0.000%	1.934	0.000%	2.209
port-orlib5_100	0.0003918260	225	**4.6577793**	**3451.980**	**4**	**4.6577793**	3640.422	**4**	**4.6577793**	9833.002	**4**	0.000%	2.849	0.000%	2.701
port-orlib5_150	0.0003272876	225	4.6264568	**7934.132**	**8**	**4.6228591**	9564.905	**9**	4.6264568	10684.629	**8**	0.000%	1.347	-0.078%	1.117
∑			51.2448641	107592.611	**85**	51.2403402	105345.493	**85**	**51.2498536**	**75729.075**	86	-0.218%	**28.011**	-0.335%	**32.362**
avg			2.5622432	5379.631	**4.250**	2.5620170	5267.275	**4.250**	**2.5624927**	**3786.454**	4.300	-0.011%	**1.401**	-0.017%	**1.618**
std			(1.2414709)	(7721.167)	(63.143)	(1.2411631)	(7478.113)	(3.210)	(1.2413996)	(4287.608)	(3.045)	(0.032%)	(1.672)	(0.040%)	(1.928)
geo (CPU)				1324.256			1090.151			**909.83**					

Table 2: MVPS, comparative results of MS and MWU for the transaction cost function (a).

			MWU			MWU+LB			MS			MWU vs. MS		MWU+LB vs. MS	
Instance	σ	n	objective	CPU	$\|x^*\|_0$	objective	CPU	$\|x^*\|_0$	objective	CPU	$\|x^*\|_0$	Γ	Λ	Γ	Λ
port-orlib1_000	0.0047755010	31	**0.3283893**	1.502	**1**	**0.3283893**	**1.295**	**1**	**0.3283893**	4.896	1	0.000%	3.260	0.000%	3.781
port-orlib1_050	0.0021522075	31	0.3268002	1.698	2	0.3268002	**1.488**	2	**0.3268555**	5.310	**1**	-0.017%	3.127	-0.017%	3.569
port-orlib1_100	0.0010585969	31	**0.3264242**	**1.807**	**3**	**0.3264242**	1.977	**3**	**0.3264242**	5.493	**3**	0.000%	3.040	0.000%	2.778
port-orlib1_150	0.0007158421	31	**0.3255758**	**2.890**	**6**	**0.3255758**	3.002	**6**	**0.3255758**	5.649	**6**	0.000%	1.955	0.000%	1.882
port-orlib2_000	0.0028352430	85	**0.5696578**	7.404	**1**	**0.5696578**	**7.059**	**1**	**0.5696578**	43.570	1	0.000%	5.885	0.000%	6.172
port-orlib2_050	0.0004953237	85	**0.5677078**	19.638	**5**	0.5677075	**11.335**	8	0.5677077	52.401	**5**	0.000%	2.668	0.000%	4.623
port-orlib2_100	0.0002704062	85	0.5671808	611.424	**8**	0.5671809	776.712	10	**0.5671812**	**35.439**	10	0.000%	0.058	0.000%	0.046
port-orlib2_150	0.0001663200	85	**0.5668838**	**568.841**	**10**	**0.5668838**	1560.205	**10**	**0.5668838**	1737.240	**10**	0.000%	3.054	0.000%	1.113
port-orlib3_000	0.0015166351	89	**0.7248280**	**10.288**	**1**	**0.7248280**	13.463	**1**	**0.7248280**	37.717	**1**	0.000%	3.666	0.000%	2.802
port-orlib3_050	0.0005849758	89	0.7234229	19.936	3	**0.7234556**	**19.177**	**2**	**0.7234556**	46.714	**2**	-0.005%	2.343	0.000%	2.436
port-orlib3_100	0.0003215941	89	**0.7229391**	**17.926**	**7**	**0.7229391**	1210.131	**7**	0.7229587	651.578	**7**	-0.003%	36.348	-0.003%	0.538
port-orlib3_150	0.0002239117	89	**0.7225929**	**1165.350**	**10**	0.7225928	3643.605	**10**	**0.7225929**	2864.165	**10**	0.000%	2.458	0.000%	0.786
port-orlib4_000	0.0029387241	98	**0.8862800**	**11.850**	**1**	**0.8862800**	29.022	**1**	**0.8862800**	50.988	**1**	0.000%	4.303	0.000%	1.757
port-orlib4_050	0.0006828450	98	**0.8846300**	**28.928**	**2**	**0.8846300**	50.449	**2**	**0.8846300**	61.230	**2**	0.000%	2.117	0.000%	1.214
port-orlib4_100	0.0003059553	98	0.8840668	1005.380	**7**	0.8840668	**44.561**	**7**	**0.8840702**	681.456	9	0.000%	0.678	0.000%	15.293
port-orlib4_150	0.0001613979	98	0.8836949	**3778.496**	**10**	0.8836945	4306.084	**10**	**0.8836971**	4254.159	**10**	0.000%	1.126	0.000%	0.988
port-orlib5_000	0.0016485224	225	**1.4120658**	211.948	**2**	**1.4120658**	**117.720**	**2**	1.4120587	381.806	3	0.001%	1.801	0.001%	3.243
port-orlib5_050	0.0005150277	225	**1.4112207**	4895.417	**5**	**1.4112207**	3055.888	**5**	**1.4112207**	**892.922**	**5**	0.000%	0.182	0.000%	0.292
port-orlib5_100	0.0003918260	225	**1.4109834**	4291.445	**7**	**1.4109834**	2450.898	**7**	**1.4109834**	**777.179**	**7**	0.000%	0.181	0.000%	0.317
port-orlib5_150	0.0003272876	225	**1.4108196**	**755.727**	**7**	1.4108195	2242.568	**7**	**1.4108196**	1368.145	**7**	0.000%	1.810	0.000%	0.610
∑			15.6561636	17407.895	**98**	15.6561956	19546.639	102	**15.6562699**	**13958.057**	101	-0.024%	**80.060**	-0.020%	**54.240**
avg			0.7828082	870.395	**4.900**	0.7828098	977.332	5.100	**0.7828135**	**697.903**	5.050	-0.001%	**4.003**	-0.001%	**2.712**
std			(0.3736084)	(1542.238)	(3.243)	(0.3736081)	(1406.957)	(3.478)	(0.3736038)	(1121.600)	(3.517)	(0.004%)	(7.751)	(0.004%)	3.380
geo (CPU)				**71.630**			87.557			127.168					

Table 3: MVPS, comparative results of MS and MWU for the transaction cost function (b).

			MWU			MWU+LB			MS			MWU vs. MS		MWU+LB vs. MS	
Instance	σ	n	objective	CPU	$\|x^*\|_0$	objective	CPU	$\|x^*\|_0$	objective	CPU	$\|x^*\|_0$	Γ	Λ	Γ	Λ
port-orlib1_000	0.0047755010	31	**-0.0004356**	**1.024**	**1**	-0.0032985	3.179	10	-0.0039929	5.554	10	816.628%	5.424	21.053%	1.747
port-orlib1_050	0.0021522075	31	**-0.0004219**	**1.104**	**2**	-0.0030836	3.956	10	-0.0039929	5.697	10	846.405%	5.160	29.487%	1.440
port-orlib1_100	0.0010585969	31	**-0.0021266**	**1.455**	**8**	**-0.0021266**	2.363	**8**	-0.0041346	5.680	9	94.427%	3.904	94.427%	2.404
port-orlib1_150	0.0007158421	31	-0.0037476	**2.611**	7	**-0.0034443**	3.960	**5**	-0.0038742	3.893	7	3.377%	1.491	12.483%	0.983
port-orlib2_000	0.0028352430	85	-0.0001446	9.993	5	-0.0000687	17.275	2	-0.0002016	32.149	6	39.408%	3.217	193.362%	1.861
port-orlib2_050	0.0004953237	85	-0.0000917	**7.269**	**3**	**-0.0001101**	13.670	5	-0.0002016	32.370	6	119.742%	4.453	83.167%	2.368
port-orlib2_100	0.0002704062	85	**-0.0005385**	**33.373**	**9**	**-0.0005385**	42.610	**9**	**-0.0005385**	64.918	**9**	-0.002%	1.945	-0.002%	1.524
port-orlib2_150	0.0001663200	85	**-0.0024857**	1322.320	**10**	**-0.0024857**	262.913	**10**	**-0.0024857**	**1115.010**	**10**	-0.001%	0.843	-0.001%	4.241
port-orlib3_000	0.0015166351	89	-0.0001214	**7.340**	3	**-0.0001205**	21.984	**1**	-0.0001825	41.263	4	50.339%	5.622	51.466%	1.877
port-orlib3_050	0.0005849758	89	**-0.0001355**	**6.643**	**3**	**-0.0001355**	13.929	**3**	-0.0001825	37.436	4	34.663%	5.635	34.662%	2.688
port-orlib3_100	0.0003215941	89	**-0.0007110**	**17.459**	**6**	-0.0007768	50.735	7	-0.0007760	42.141	7	9.139%	2.414	-0.108%	0.831
port-orlib3_150	0.0002239117	89	-0.0032757	13349.900	**10**	-0.0032221	10905.202	**10**	**-0.0031912**	**2957.484**	**10**	-2.580%	0.222	-0.960%	0.271
port-orlib4_000	0.0029387241	98	-0.0001820	**11.705**	**3**	-0.0005766	16.740	7	**-0.0001414**	28.049	**3**	-22.293%	2.396	-75.477%	1.676
port-orlib4_050	0.0006828450	98	**-0.0001404**	**14.638**	**3**	-0.0003466	16.504	5	-0.0001414	26.859	**3**	0.719%	1.835	-59.208%	1.627
port-orlib4_100	0.0003059553	98	-0.0006288	**19.624**	**6**	-0.0006801	64.904	**6**	**-0.0006227**	33.050	**6**	-0.967%	1.684	-8.441%	0.509
port-orlib4_150	0.0001613979	98	**-0.0022239**	8371.655	**10**	**-0.0022239**	**6198.304**	**10**	**-0.0022239**	1570.873	**10**	0.002%	0.188	0.002%	0.253
port-orlib5_000	0.0016485224	225	**-0.0000647**	**72.465**	**7**	-0.0000712	149.351	8	-0.0000757	167.038	9	16.913%	2.305	6.265%	1.118
port-orlib5_050	0.0005150277	225	-0.0004978	**108.452**	8	-0.0004978	224.882	8	**-0.0001212**	291.879	**5**	-75.654%	2.691	-75.654%	1.298
port-orlib5_100	0.0003918260	225	-0.0013865	129.598	8	-0.0007129	**95.863**	8	**-0.0004068**	209.771	**6**	-70.660%	1.619	-42.937%	2.188
port-orlib5_150	0.0003272876	225	-0.0016889	180.181	**8**	-0.0016889	**169.206**	**8**	**-0.0010823**	546.634	**8**	-35.917%	3.034	-35.917%	3.231
∑			**-0.0210489**	23668.809	**120**	-0.0262090	18277.530	140	-0.0285696	**7217.748**	142	**1823.689%**	**56.082**	**227.670%**	**34.135**
avg			**-0.0010524**	1183.440	**6.000**	-0.0013104	913.877	7.000	-0.0014285	**360.887**	7.100	**91.184%**	**2.804**	**11.383%**	**1.707**
std			(0.0011464)	(3419.072)	(2.920)	(0.0012441)	(2722.510)	(2.753)	(0.0015780)	(737.085)	(2.469)	(257.358%)	(1.725)	(62.458%)	(0.985)
geo (CPU)				**31.288**			47.339			65.661					

Table 4: MVPS, comparative results of MS and MWU for the transaction cost function (c).

			MWU			MWU+LB			MS			MWU vs. MS		MWU+LB vs. MS	
Instance	σ	n	objective	CPU	$\|x^*\|_0$	objective	CPU	$\|x^*\|_0$	objective	CPU	$\|x^*\|_0$	Γ	Λ	Γ	Λ
port-orlib1_000	0.0047755010	31	**-0.0003929**	**0.825**	**2**	-0.0005442	3.319	9	-0.0008732	5.750	9	122.260%	6.970	60.446%	1.732
port-orlib1_050	0.0021522075	31	**-0.0003264**	**1.040**	**3**	-0.0006592	3.127	6	-0.0008732	6.008	9	167.496%	5.777	32.464%	1.921
port-orlib1_100	0.0010585969	31	-0.0007308	**1.780**	**9**	**-0.0006770**	4.030	**9**	-0.0010003	5.716	**9**	36.887%	3.211	47.760%	1.418
port-orlib1_150	0.0007158421	31	-0.0027051	**2.165**	**5**	-0.0016986	4.537	6	**-0.0010005**	3.970	8	-63.015%	1.834	-41.098%	0.875
port-orlib2_000	0.0028352430	85	-0.0000428	**4.970**	6	**-0.0000004**	13.170	**3**	-0.0000428	28.166	6	0.000%	5.667	11034.235%	2.139
port-orlib2_050	0.0004953237	85	-0.0000784	**4.272**	9	-0.0000544	16.765	7	**-0.0000464**	31.663	**6**	-40.774%	7.412	-14.598%	1.889
port-orlib2_100	0.0002704062	85	-0.0001552	**16.052**	**9**	-0.0001552	33.207	**9**	**-0.0001273**	96.677	**9**	-17.953%	6.023	-17.956%	2.911
port-orlib2_150	0.0001663200	85	-0.0007034	**165.865**	**10**	-0.0008274	3627.871	**10**	**-0.0007033**	327.515	**10**	-0.001%	1.975	-14.993%	0.090
port-orlib3_000	0.0015166351	89	-0.0001257	**6.590**	**1**	-0.0001605	13.743	9	**-0.0001232**	29.997	6	-1.977%	4.552	-23.212%	2.183
port-orlib3_050	0.0005849758	89	-0.0001166	**10.815**	**6**	**-0.0001025**	20.975	**6**	-0.0001232	33.973	**6**	5.667%	3.141	20.243%	1.620
port-orlib3_100	0.0003215941	89	**-0.0002009**	**22.068**	**7**	-0.0004476	29.281	**7**	**-0.0002009**	180.959	7	0.000%	8.200	-55.126%	6.180
port-orlib3_150	0.0002239117	89	**-0.0007063**	9872.064	**10**	-0.0009387	9067.600	**10**	**-0.0007063**	**775.900**	**10**	0.000%	0.079	-24.759%	0.086
port-orlib4_000	0.0029387241	98	**-0.0000210**	**8.926**	**2**	-0.0001277	23.203	7	-0.0001137	34.462	5	441.257%	3.861	-10.950%	1.485
port-orlib4_050	0.0006828450	98	**-0.0000737**	**9.197**	**3**	-0.0001134	24.853	6	-0.0001578	35.712	8	114.188%	3.883	39.146%	1.437
port-orlib4_100	0.0003059553	98	**-0.0002065**	**10.295**	9	-0.0002681	23.826	**7**	**-0.0002065**	36.192	9	0.000%	3.515	-22.957%	1.519
port-orlib4_150	0.0001613979	98	-0.0006690	**6321.129**	**9**	**-0.0005819**	3312.971	10	-0.0006155	4837.097	10	-8.008%	0.765	5.764%	1.460
port-orlib5_000	0.0016485224	225	-0.0000198	**93.510**	**9**	-0.0000172	124.442	**9**	**-0.0000155**	151.180	**9**	-21.759%	1.617	-9.970%	1.215
port-orlib5_050	0.0005150277	225	-0.0001077	823.956	8	-0.0000869	**257.904**	**7**	**-0.0000762**	1347.551	**7**	-29.266%	1.635	-12.290%	5.225
port-orlib5_100	0.0003918260	225	-0.0002729	**130.751**	7	-0.0006698	148.388	**6**	**-0.0001507**	638.984	**6**	-44.768%	4.887	-77.499%	4.306
port-orlib5_150	0.0003272876	225	**-0.0003806**	**104.124**	**8**	**-0.0003806**	142.845	**8**	-0.0002542	269.757	**8**	-33.199%	2.591	-33.199%	1.888
∑			-0.0080357	17610.394	**132**	-0.0085112	16896.057	151	**-0.0074109**	**8877.229**	157	**627.036%**	**77.594**	**10881.450%**	**41.580**
avg			-0.0004018	880.520	**6.600**	-0.0004256	844.803	7.550	**-0.0003705**	**443.861**	7.850	**31.352%**	**3.880**	**544.073%**	**2.079**
std			(0.0005941)	(2540.745)	(2.945)	(0.0004182)	(2203.036)	(1.820)	(0.0003567)	(1089.089)	(1.599)	(113.077%)	(2.276)	(2469.367%)	1.536
geo (CPU)				**25.737**			50.570			73.957					

Table 5: MVPS, comparative results of MS and MWU for the transaction cost function (d).

			MWU			MWU+LB			MS			MWU vs. MS		MWU+LB vs. MS	
Instance	σ	n	objective	CPU	$\|x^*\|_0$	objective	CPU	$\|x^*\|_0$	objective	CPU	$\|x^*\|_0$	Γ	Λ	Γ	Λ
port-orlib1_000	0.0047755010	31	0.0217170	**1.877**	**6**	0.0395563	14.952	10	**0.0487432**	43.180	10	-124.447%	23.005	-23.225%	2.888
port-orlib1_050	0.0021522075	31	0.0469337	**3.138**	**10**	0.0476659	56.161	**10**	**0.0488171**	37.351	**10**	-4.013%	11.903	-2.415%	0.665
port-orlib1_100	0.0010585969	31	0.0432059	**2.984**	**10**	0.0413710	4.962	**10**	**0.0498939**	58.515	**10**	-15.479%	19.610	-20.601%	11.793
port-orlib1_150	0.0007158421	31	**0.0399172**	**2.588**	**9**	0.0376060	6.172	**9**	0.0388779	6.757	**9**	2.604%	2.611	-3.382%	1.095
port-orlib2_000	0.0028352430	85	0.0136601	10.510	4	0.0133721	**8.726**	**3**	**0.0184047**	375.738	10	-34.734%	35.751	-37.635%	43.060
port-orlib2_050	0.0004953237	85	0.0383876	39.267	7	**0.0387789**	**33.822**	**6**	0.0186861	522.376	10	51.323%	13.303	51.814%	15.445
port-orlib2_100	0.0002704062	85	**0.0283756**	4226.211	**10**	0.0166890	4382.785	**10**	0.0207083	**524.517**	**10**	27.021%	0.124	-24.083%	0.120
port-orlib2_150	0.0001663200	85	**0.0315010**	3642.877	**10**	0.0307374	3989.962	**10**	0.0268662	**1599.415**	**10**	14.713%	0.439	12.594%	0.401
port-orlib3_000	0.0015166351	89	0.0116564	**9.548**	**2**	0.0029338	22.035	8	**0.0234077**	240.592	10	-100.814%	25.198	-697.857%	10.919
port-orlib3_050	0.0005849758	89	0.0347923	**19.026**	**8**	**0.0348732**	102.802	**8**	0.0239912	441.404	10	31.044%	23.200	31.204%	4.294
port-orlib3_100	0.0003215941	89	0.0342867	903.586	**9**	**0.0392775**	2142.790	10	0.0188460	**719.657**	10	45.034%	0.796	52.018%	0.336
port-orlib3_150	0.0002239117	89	**0.0284570**	15650.882	**10**	0.0269692	11231.379	**10**	0.0255798	**5050.864**	**10**	10.111%	0.323	5.152%	0.450
port-orlib4_000	0.0029387241	98	0.0130890	**4.641**	**2**	0.0123611	66.034	10	**0.0233697**	432.908	10	-78.545%	93.279	-89.058%	6.556
port-orlib4_050	0.0006828450	98	**0.0583659**	**32.832**	**10**	0.0541541	91.192	**10**	0.0239543	272.405	**10**	58.958%	8.297	55.766%	2.987
port-orlib4_100	0.0003059553	98	0.0135301	1082.307	**10**	0.0078843	696.938	**10**	**0.0248876**	**513.718**	**10**	-83.943%	0.475	-215.660%	0.737
port-orlib4_150	0.0001613979	98	0.0236738	15929.002	**10**	**0.0260514**	24604.167	**10**	0.0252765	**5982.823**	**10**	-6.770%	0.376	2.974%	0.243
port-orlib5_000	0.0016485224	225	0.0326302	1448.934	**6**	**0.032836**	**788.687**	**6**	0.0017349	5513.344	10	94.683%	3.805	94.717%	6.991
port-orlib5_050	0.0005150277	225	0.0031309	**302.041**	**9**	**0.0073133**	370.425	10	0.0018351	350.660	**9**	41.386%	1.161	74.907%	0.947
port-orlib5_100	0.0003918260	225	0.0078564	5516.043	**7**	0.0035668	4774.516	8	**0.0103946**	**1525.448**	10	-32.308%	0.277	-191.430%	0.319
port-orlib5_150	0.0003272876	225	**0.0143170**	968.684	**10**	0.0143169	**730.436**	**10**	**0.0143170**	913.920	**10**	0.000%	0.943	0.000%	1.251
$\sum$			**0.5394835**	49796.978	**159**	0.5283151	54118.943	178	0.4885919	**25125.592**	198	-104.177%	**264.875**	-924.200%	**111.494**
avg			**0.0269742**	2489.849	**7.950**	0.0264158	2705.947	8.900	0.0244296	**1256.280**	9.900	-5.209%	**13.244**	-46.210%	**5.575**
std			(0.0145719)	(4819.775)	(2.685)	(0.0155312)	(5846.227)	(1.917)	(0.0135588)	(1891.435)	(0.308)	(56.991%)	(21.729)	(172.079%)	9.900
geo (CPU)				**135.816**			247.786			426.464					

Table 6: MVPS, comparative results of MS and MWU for the transaction cost function (e).

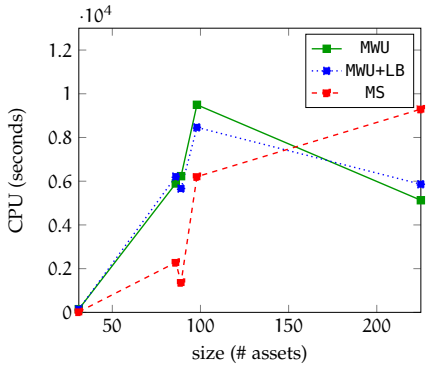
While with transaction costs (a) and (b) MS performs better than MWU, with transaction costs (c) and (e) MWU overcomes MS: we can reasonably imply that MWU behaves better with respect to "high nonlinear" transaction cost functions. In particular, for the geometric average values, the relative improvement we obtain is considerably high.

A secondary observation is about the number of assets, which composed the optimal portfolios. In MVPS problems, defining a portfolio with few assets could be a secondary goal, since for each asset we have several costs, such as transaction costs, monitoring costs, and brokerage fee (see Di Lorenzo et al. [74]). MWU algorithm produces small values of the number of assets compared to MS algorithm.

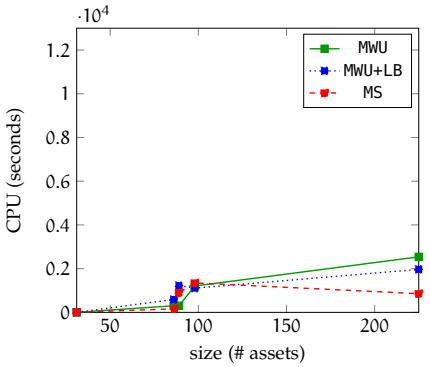
Finally, for the geometric mean for the CPU time, the MS defeats MWU only for the cost function (a), in all the other cases MWU overcomes MS.

3.9 CONCLUSIONS

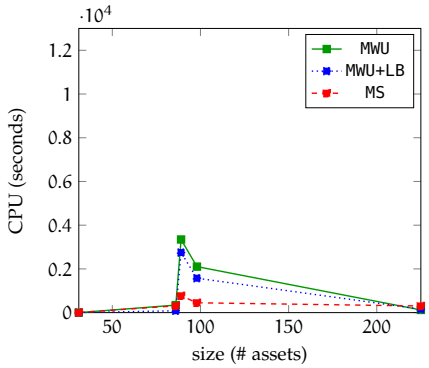
In this chapter we dealt with the Mean-Variance Portfolio Selection problem. At the beginning, we have illustrated the mathematical models proposed in the literature with a survey about the possible objective functions and constraints. Then, we have adapted the MWU algorithm to the real-world portfolio problem with separable transaction cost with respect to the strategy to define the pointwise reformulation and to compute the costs/-gains necessary for the algorithm. Computational experiments on real-world instances allow us to observe that the MWU algorithm performs better than the MS algorithm for "heavily nonlinear" instances.



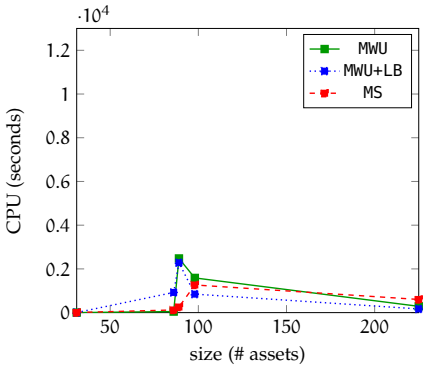
(a) "Easy" concave transaction costs.



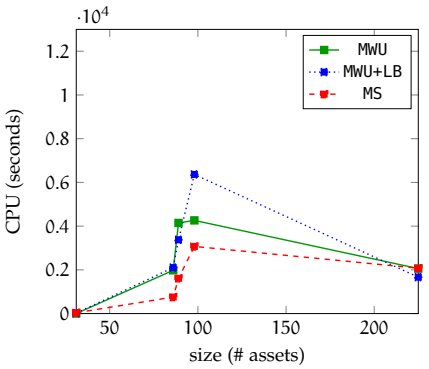
(b) "Hard" concave transaction costs.



(c) "Easy" trigonometric transaction costs.



(d) "Medium" trigonometric transaction costs.



(e) "Hard" trigonometric transaction costs.

Figure 2: MVPS, CPU time vs. size of the problem n (# assets).

4 MULTIPLE NONLINEAR KNAPSACK PROBLEMS

4.1 INTRODUCTION

Let x be an $m \times n$ array of non negative real variables $x = [x_{ij}]$ ($i = 1,\dots,m$, $j = 1,\dots,n$) and define $M = \{1,\dots,m\}$ and $N = \{1,\dots,n\}$. We consider a multiple nonlinear knapsack problem in which

- the objective function and the capacity constraints are expressed by separable, continuously differentiable functions $f_j(x_{ij})$ and $g_j(x_{ij})$ ($i \in M$, $j \in N$);
- the values of f and g do not depend on i, i.e., $f_j(x_{ij}) = f_j(x_{kj})$ and $g_j(x_{ij}) = g_j(x_{kj})$ when $x_{ij} = x_{kj}$ for $j \in N$ and $i, k \in M$;
- $f_j(x_{ij})$ and $g_j(x_{ij})$ are nonlinear non negative non-decreasing functions for $j \in N$ and $i \in M$;
- for each $j \in N$, the total value of x_{ij} over all $i \in M$ cannot exceed a given upper bound u_j;
- integrality requirements may be imposed on part of the variables.

Remark 4.1.1. *Note that there is no further assumption on $f_j(x_{ij})$ and $g_j(x_{ij})$ which, in general, can be non-convex and non-concave.* □

The Multiple NonLinear (Separable) Knapsack Problem (MNLKP) is:

$$\max \sum_{i\in M}\sum_{j\in N} f_j(x_{ij}) \tag{4.1a}$$

$$\text{s.t.} \sum_{j\in N} g_j(x_{ij}) \leqslant c_i \qquad i \in M \tag{4.1b}$$

$$\sum_{i\in M} x_{ij} \leqslant u_j \qquad j \in N \tag{4.1c}$$

$$x_{ij} \geqslant 0 \qquad i \in M,\ j \in N \tag{4.1d}$$

$$x_{ij} \text{ integer} \qquad i \in M,\ j \in \overline{N} \subseteq N, \tag{4.1e}$$

which can be informally described as follows. We are given m knapsacks and n items. Each item j has a *profit function* $f_j(x_{ij})$ and a *weight function* $g_j(x_{ij})$ ($i \in M$), and each knapsack i has a *capacity* c_i. For each item j we want to assign x_{ij} quantities (some restricted to integer values) to the knapsacks so that

- the overall assigned profit is maximized, see (4.1a);

- for each knapsack i the overall assigned weight does not exceed the corresponding capacity, see (4.1b);
- for each item j the overall assigned quantity does not exceed the corresponding upper bound, see (4.1c).

The MNLKP generalizes the classical 0-1 Multiple Linear Knapsack Problem (MLKP) (see, e.g., Martello and Toth [182] and Kellerer et a. [142]): in the MLKP x_{ij} are binary decision variables, i.e., $x_{ij} \in \{0, 1\}$ for all $i \in M$ and $j \in N$, and the profit and the weight functions are linear, i.e., $f_j(x_{ij}) = p_j x_{ij}$ and $g_j(x_{ij}) = w_j x_{ij}$, and $u_j = 1$ for all $j \in N$.

It follows that MNLKP is, at least, strongly **NP**-hard. Moreover, the MNLKP generalizes also the (single) NonLinear Knapsack Problem (NLKP) [63]: MNLKP reduces to NLKP when $m = 1$ and, consequently, objective function (4.1a) and constraints (4.1b) read $\sum_{j \in N} f_j(x_{ij})$ and $\sum_{j \in N} g_j(x_{ij}) \leqslant c_1$, respectively.

The nonlinear knapsack structure arises in many different real-world problems, such as portfolio selection, capacity and production planning, and resource allocation (see, e.g., Ibaraki and Katoh [123], Bretthauer and Shetty [33], and Li and Sun [156]). For instance, we assume we have m different economical resources and n products and we want to subdivide a certain amount of advertising budget c_i related to resource i in order to maximize the overall expected sales for all resources. Obviously, the profit is increasing with the advertising investment since more and more buyers happened to find out our advertisement. However, at some point, a saturation effect occurs when, despite we further increase the investments, no more buyers are interested in our products. In this example the profit function happens to be non-convex non-concave with a shape represented in Figure 3 where the advertisement cost could be linear, i.e., $g_j(x_{ij}) = x_{ij}$, if a constant unit cost is assumed or nonlinear if economies of scale are considered, i.e., unit costs decrease with size.

To the best of our knowledge no tailored exact methods or heuristics have been proposed for the MNLKP. Zhang and Hua [250] proposed an exact method for the minimization version of the convex continuous NLKP, i.e., when all the profit and weight functions are convex, and all variables are continuous, i.e., $\overline{N} = \emptyset$. Zhang and Chen [249] described exact and heuristic methods for the pure integer version of same problem, i.e., when $\overline{N} = N$.

The rest of the chapter consists of three other sections. In Section 4.2 we apply the MWU algorithm for MNLPs, by specializing its main steps, namely the construction of the pointwise reformulation and the definition of costs/gains, to the MNLKP. Since this method does not fit very well this kind of optimization problems, we propose also other heuristic procedures. First of all we introduce the surrogate and Lagrangian relaxations for MNLKPs in Section 4.3, while in Section 4.4 we discuss a constructive solution

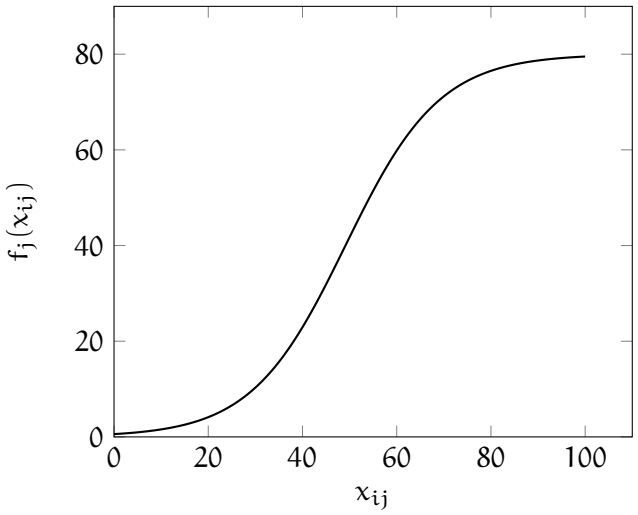


Figure 3: Example of profit function.

approach, whose main aspects are a *greedy* heuristic, two other heuristics based on the feasibility recovery of the solution produced by the surrogate relaxation, and a local search procedure to improve the quality of the heuristic solution. Extensive computational experiments are conducted both for the MWU heuristic and for the constructive one, with respect to challenging instances.

4.2 MWU FOR THE MNLKP

4.2.1 Pointwise Reformulation

In order to define a pointwise reformulation for the MNLKP, we replace the "complicated" non-convex non-concave terms $g_j(x_{ij})$ with affine terms in x_{ij}, i.e., with terms $\theta_{ij}x_{ij}$ for all $i \in M$ and $j \in N$:

$$\max \sum_{i\in M}\sum_{j\in N} f_j(x_{ij}) \tag{4.2a}$$

$$\text{s.t.} \sum_{j\in N} \theta_{ij}x_{ij} \leqslant c_i \qquad i \in M \tag{4.2b}$$

$$\sum_{i\in M} x_{ij} \leqslant u_j \qquad j \in N \tag{4.2c}$$

$$x_{ij} \geqslant 0 \qquad i \in M,\ j \in N \tag{4.2d}$$

$$x_{ij} \text{ integer} \qquad i \in M,\ j \in \overline{N} \subseteq N, \tag{4.2e}$$

Remark 4.2.1. *The pointwise reformulation* (4.2) *is spanning, since the replacement terms* θ_{ij} *correspond to the replaced terms* $\frac{g_j(x_{ij})}{x_{ij}}$ *for all* $i \in M$ *and* $j \in N$. *By Lemma* 2.2.6*, there exist values of* θ *which make the pointwise reformulation* (4.2) *a bounding reformulation for the original problem* (4.1). □

Remark 4.2.2. *The pointwise reformulation* (4.2) *is* not *efficient. Nevertheless, the feasible set of the reformulation* (4.2) *is polyhedral. We note that the problem* (4.1) *is always feasible since the zero solution, i.e.,* $x_{ij} = 0$ *for all* $i \in M$ *and* $j \in N$*, is always a feasible point for the problem.* □

The approach, we have adopted in this section, is quite similar to the one we followed for the MVPS problem with univariate cost functions: essentially we substitute the nonlinear non-convex non-concave terms with affine terms in the decision variables.

4.2.2 Computing MWU Costs/Gains

Since the replaced terms appear only in the constraint (4.2b), we do not take into account optimality issues in defining MWU costs/gains. In particular, at each iteration $t \leqslant T$, we implement the following strategy. We define a profit-to-weight ratio for each item x_{ij} for $i \in M$ and $j \in N$, as the ratio between its profit and weigh function values, representing the profitability of filling the knapsack i ($i \in M$) with x_{ij} units of item j ($j \in N$), as follows:

$$r_{ij,t} = \frac{f_j(x_{ij,t})}{g_j(x_{ij,t})} \qquad (i \in M,\ j \in N,\ t \leqslant T). \tag{4.3}$$

Then, we compute the feasibility costs/gains $\beta_{i,t}$ ($i \in N$, $t \leqslant T$) as the scaled difference from the left hand side and the right hand side of constrains (4.1b) calculated on the current point x_t (see Section 2.4.3):

$$\beta_{i,t} = \frac{\max(\sum_{j \in N} g_j(x_{ij,t}) - c_i,\ 0)}{\max_{s \leqslant t}(\max(\sum_{j \in N} g_j(x_{ij,s}) - c_i),\ 0)} \qquad (i \in M,\ t \leqslant T). \tag{4.4}$$

Finally, we need to spread the previous feasibility cost/gain to the item j ($j \in N$). We simply scale the feasibility cost/gain with respect to the profit-to-weight ratio, defining in this way our MWU costs/gains for the MNLKP:

$$\psi_{ij,t} = \beta_{i,t} \frac{r_{ij,t}}{\sum_{j \in N} r_{ij,t}} \qquad (i \in M,\ j \in N,\ t \leqslant T). \tag{4.5}$$

We choose the profit-to-weight ratio since in heuristic methods for NLKP it is used to sort the items (see D'Ambrosio and Martello [63]). The items with a greater ratio are more promising in terms of the trade-off represented by the objective function we want to maximize and the amount we have to pay in order to fill the knapsacks with those items. In the heuristic method proposed by D'Ambrosio and Martello [63] for the NLKP, they select the item with the best profit-to-ratio at first and then fill as much as possible the knapsack with that item.

4.2.3 Computational Experiments

The MWU method was experimentally compared with the MS algorithm. We use Ipopt [126] with its default options: Ipopt is an exact solver for convex NLPs, hence it can be used as local solvers for non-convex NLPs. The

number of iterations was set to $T := 10$ both for the MWU and MS. Moreover, we compare the MWU with Ipopt with one starting point and Couenne [59], which is an open-source global solver for MINLPs.

For the profit and the weight functions, we used the ones proposed in D'Ambrosio and Martello [63]. In particular, the profits were always obtained from

$$f_j(x_{ij}) = \frac{c_j}{1 + b_j e^{-a_j(x_{ij}+d_j)}} \tag{4.6}$$

by uniformly randomly generating a_j in $[0.1, 0.2]$, b_j and c_j in $[0, 100]$, and d_j in $[-100, 0]$. These functions replicate the sigmoid shape of the profit function depicted in Figure 3. The upper bounds on the variables x_{ij} were set to $u_j := 100$ for all $j \in N$. For the weight, we adopted the concave increasing functions:

$$g_j(x_{ij}) = \sqrt{p_j x_{ij} + q_j} - \sqrt{q_j}. \tag{4.7}$$

Moreover, we define two sets of instances depending on the way we generated the capacities. In order to obtain challenging instances, we adopted the numerical methods described in Chapter 6 of Martello and Toth [182]:

Similar capacities

$$c_i \text{ uniformly random in } \left[0.4 \sum_{j=1}^{n} \frac{g_j(u_j)}{m}, 0.6 \sum_{j=1}^{n} \frac{g_j(u_j)}{m}\right] \quad (i = 1, \ldots, m-1), \tag{4.8}$$

and *Dissimilar capacities*

$$c_i \text{ uniformly random in } \left[0, \left(0.5 \sum_{j=1}^{n} g_j(u_j) - \sum_{k=1}^{i-1} c_k\right)\right] \quad (i = 1, \ldots, m-1). \tag{4.9}$$

In both cases, the m-th capacity was set to:

$$c_m = 0.5 \sum_{j=1}^{n} g_j(u_j) - \sum_{i=1}^{m-1} c_i. \tag{4.10}$$

The value of the number of items n varied in the set $\{10, 20, 50\}$, while the value of the number of knapsacks m varied in the set $\{2, 5, 10\}$. For each combination of (n, m) 20 real instances, i.e., with $\overline{N} = \emptyset$ are generated. The total number of instance we tested was 360. All instances are available at `http://or.dei.unibo.it/library/multiple-nonlinear-knapsack-problem`. If the pointwise reformulation was not solved within the time limit, we set its solution to zero.

All the experiments were performed on an Intel Xeon, CPU 3220, 2.4 GHz, using only one processor. The local method (Ipopt), the global algorithm (Couenne), and the MWU were run with a time limit of one CPU hour per

problem.

Tables 7 and 9 report the average solution values for the group of instances with similar and dissimilar capacities, respectively. The entries give:

- number of knapsacks;
- number of items;
- average values produced by MWU, Ipopt with a single starting point (Ipopt_1), MS, i.e., Ipopt with ten starting points (Ipopt_10), and Couenne;

Tables 8 and 10 report the average CPU times (in seconds) for instances with similar and dissimilar capacities, respectively.

Tables 7 and 9 show that generally the solution produced by the MWU heuristic was better than the ones produced by Ipopt_1 and worse than the ones produced by Ipopt_10. For the bigger continuous similar instances, Table 7 shows the MWU algorithm was on average completely outperformed by Ipopt for all the instances with $m = 5$ and for the instances with $m = 10$ and $n = 20$. For the similar instances there is only one case in which the MWU algorithm is better than Ipopt_10: the dissimilar instances with $m = 2$ and $n = 50$.

Moreover, Tables 8 and 10 clearly indicate the average CPU times for MWU was one order of magnitude larger than the ones of Ipopt_10 and around two orders of magnitude larger than the ones of Ipopt_1.

We point out all the knapsack problems were solved to optimality within the time limit.

From the previous observations, we can argue that the MWU method was in general outperformed by the heuristic methods available for the MNLKPs both in terms of quality of the solution found and of CPU times needed. The behavior of the proposed heuristic was not satisfying, and this is the reason why in the next sections we will introduce a different approach to heuristically solve MNLKPs relying on the discretization of the solution space and on the surrogate relaxation.

4.3 RELAXATIONS

In this section we introduce some relevant relaxations of MNLKP.

Let $(\pi_1, \ldots, \pi_m)$ be an m-dimensional vector of non negative multipliers. By multiplying the i-th constraints (4.1b) and summing up all the new ca-

Table 7: MNLKP, nonlinear weights, similar capacities. Average solution values over 20 instances.

		Real Variables			
m	n	MWU	Ipopt_1	Ipopt_10	Couenne
2	10	326.57	313.50	351.70	362.63
2	20	622.24	569.20	635.35	593.43(12)
2	50	1,777.83	1,714.74	1,799.78	n/a(20)
2	total	2,726.64	2,597.44	2,786.83	–
5	10	266.09	271.61	312.63	299.06(5)
5	20	677.78	687.95	733.25	n/a(20)
5	50	1,705.66	1,705.86	1,782.99	n/a(20)
5	total	2,649.53	2,655.42	2,828.87	–
10	10	195.08	187.40	218.59	212.07(2)
10	20	658.91	664.58	702.21	n/a(20)
10	50	1,873.94	1,864.39	1,927.13	n/a(20)
10	total	2,727.93	2,716.37	2,847.93	–
total	total	8,104.10	7,979.23	8,463.63	–

Table 8: MNLKP, nonlinear weights, similar capacities. Average CPU times over 20 instances.

		Real Variables			
m	n	MWU	Ipopt_1	Ipopt_10	Couenne
2	10	1.48	0.05	0.51	1,113.88
2	20	8.08	0.10	1.18	3,601.81(12)
2	50	33.01	0.34	3.45	n/a(20)
2	total	42.57	0.49	5.14	–
5	10	8.75	0.16	1.72	3,600.61(5)
5	20	22.82	0.32	3.57	n/a(20)
5	50	68.97	1.17	11.76	n/a(20)
5	total	100.54	1.65	17.05	–
10	10	15.88	0.39	3.78	3,600.54(2)
10	20	43.02	0.93	8.57	n/a(20)
10	50	122.74	3.35	30.70	n/a(20)
10	total	181.64	4.67	43.05	–
total	total	324.75	6.81	65.24	–

Table 9: MNLKP, nonlinear weights, dissimilar capacities. Average solution values over 20 instances.

		Real Variables			
m	n	MWU	Ipopt_1	Ipopt_10	Couenne
2	10	311.18	298.24	339.50	355.19
2	20	600.98	547.68	610.20	564.65(9)
2	50	1,766.38	1,619.80	1,737.00	n/a(20)
2	total	2,678.54	2,465.72	2,686.70	–
5	10	288.27	285.28	318.68	329.51(1)
5	20	688.35	670.57	717.91	659.84(17)
5	50	1,654.93	1,614.39	1,723.24	n/a(20)
5	total	2,631.55	2,570.24	2,759.83	–
10	10	316.10	314.88	345.18	354.52
10	20	687.13	678.59	741.04	699.31(11)
10	50	1,825.00	1,807.00	1,909.65	n/a(20)
10	total	2,828.23	2,800.47	2,995.87	–
total	total	8,138.32	7,836.43	8,442.40	–

Table 10: MNLKP, nonlinear weights, dissimilar capacities. Average CPU times over 20 instances.

		Real Variables			
m	n	MWU	Ipopt_1	Ipopt_10	Couenne
2	10	1.84	0.05	0.53	794.39
2	20	8.59	0.11	1.14	3,307.95(9)
2	50	33.88	0.34	3.36	n/a(20)
2	total	44.31	0.50	5.03	–
5	10	9.27	0.14	1.53	2,790.28(1)
5	20	24.62	0.34	3.23	3,600.18(17)
5	50	68.75	1.16	10.25	n/a(20)
5	total	102.64	1.64	15.01	–
10	10	16.06	0.31	2.62	2,266.36
10	20	42.70	0.79	6.10	3,599.61(11)
10	50	126.30	2.89	23.01	n/a(20)
10	total	185.06	3.99	31.73	–
total	total	332.01	6.13	51.77	–

pacity constraints obtained, we define the surrogate relaxation, S(MNLKP,π), of the MNLKP:

$$\max \sum_{i\in M}\sum_{j\in N} f_j(x_{ij}) \tag{4.11a}$$

$$\text{s.t.} \sum_{i\in M} \pi_i \sum_{j\in N} g_j(x_{ij}) \leqslant \sum_{i\in M} \pi_i c_i \tag{4.11b}$$

$$\sum_{i\in M} x_{ij} \leqslant u_j \qquad j \in N \tag{4.11c}$$

$$x_{ij} \geqslant 0 \qquad i \in M,\ j \in N \tag{4.11d}$$

$$x_{ij} \text{ integer} \qquad i \in M,\ j \in \overline{N} \subseteq N. \tag{4.11e}$$

Let val(S(MNLKP,π)) denote the optimal value of (4.11) under given multipliers π. The *surrogate dual problem*

$$\min_{\pi\geqslant 0}\{\text{val}(\text{S(MNLKP,}\pi\text{)})\} \tag{4.12}$$

consists in finding the optimal vector of multipliers, i.e., the one producing the minimum optimal value for the surrogate relaxation, and hence the tighter upper bound for the MNLKP.

Remark 4.3.1. *While the surrogate relaxation of the 0-1 MLKP has strong duality property, i.e., the vector of multipliers which produced the minimum value for the surrogate relaxation is $\pi_i = k$ for all $i \in M$ [182], the same result does not hold for the MNLKP, as the following example shows. Let $m = 2$, $n = 1$, $u_1 = 100$, $c_1 = 10$, $c_2 = 2$, and, for $i \in \{1, 2\}$, $f_1(x_{i1}) = x_{i1}$, $g_1(x_{i1}) = 80/\left(1 + 50e^{-\frac{1}{10}(x_{i1}-10)}\right)$. For $\pi_1 = \pi_2 = 1$ the optimal solution to S(MNLKP,π) is $x_{11} = x_{21} \simeq 24$ (with $g_1(x_{11}) = g_1(x_{21}) \simeq 6$) and has value $\simeq 48$. For $\pi_1 = 1$ and $\pi_2 = 2$ such solution violates (4.11b) and the optimal solution is $x_{11} \simeq 26$ and $x_{21} \simeq 18$ (with $g_1(x_{11}) \simeq 7.2$ and $g_1(x_{21}) \simeq 3.4$), of value $\simeq 44$.* □

Even though the optimal surrogate multipliers are not known a priori, good multipliers can heuristically found (see Section 4.4.5) and, from the surrogate solution, a feasible solution can be easily defined (see Section 4.4.2).

Let $(\lambda_1, \ldots, \lambda_m)$ be an m-dimensional vector of non negative multipliers, a possible Lagrangian relaxation, L(MNLKP,λ), of the MNLKP can be obtained by relaxing (4.1b):

$$\sum_{i\in M} \lambda_i c_i + \max \sum_{i\in M}\sum_{j\in N} \left(f_j(x_{ij}) - \lambda_i g_j(x_{ij})\right) \tag{4.13a}$$

$$\text{s.t.} \sum_{i\in M} x_{ij} \leqslant u_j \qquad j \in N \tag{4.13b}$$

$$x_{ij} \geqslant 0 \qquad i \in M,\ j \in N \tag{4.13c}$$

$$x_{ij} \text{ integer} \qquad i \in M,\ j \in \overline{N} \subseteq N. \tag{4.13d}$$

Remark 4.3.2. *The Lagrangian relaxation (4.13) can be decomposed into n independent subproblems, one for each item j, with nonlinear objective functions. Other*

Lagrangian relaxations can be obtained by multiplying the upper bound constraints (4.1c) *by a vector of non negative multipliers or relaxing both the constraints* (4.1b) *and* (4.1c). *Preliminary computational experiments have shown that the solutions produced by the Lagrangian relaxations are generally worse than the ones produced by the surrogate relaxation.* □

4.4 CONSTRUCTIVE HEURISTICS

In this section we describe two different kinds of heuristic algorithms for the MNLKP: the first type of heuristic is represented by a constructive procedure based on the discretization of the solution space; the second one, instead, is based on feasibility recovery strategies to restore the feasibility of the surrogate solutions with respect to the relaxed constraints.

4.4.1 Discretization Heuristic

The constructive procedure extends the heuristic method proposed by D'Ambrosio and Martello [63] for the NLKP to the MNLKP.

We assume without loss of generality that the knapsacks are preliminary sorted in non-increasing order according to their capacities, i.e., $c_1 \geqslant c_2 \geqslant \cdots \geqslant c_m$. The algorithm is based on the discretization of the solution space. Let s be the number of sampling and $\delta_j = u_j/s$ for $j \in N$ (or $\delta_j = \max(1, \lfloor u_j/s \rfloor)$ if $j \in \overline{N}$) be the sampling step. We consider the profit-to-weight ratio, meaning the ratio between the profit functions and the weight functions evaluated over the sampling points:

$$r_{jk} = \frac{f_j(k\delta_j)}{g_j(k\delta_j)} \qquad (j \in N, k = 1, \ldots, s). \tag{4.14}$$

Moreover, we assume without loss of generality that the items are sorted in non-increasing order according to their maximum profit-to-weight ratios $\mu_j = \arg\max_{k=1,\ldots,s}\{r_{jk}\}$, i.e., $r_{1\mu_1} \geqslant r_{2\mu_2} \geqslant \cdots \geqslant r_{n\mu_n}$.

We apply the same strategy as in [63] by considering one single knapsack at each iteration. We take the first two items and we try to fill the knapsack as much as possible with the first one (see Procedure Construct(i)). We calculate the higher sampling point $\bar{\mu}_1$ such that the ratio of the first item is greater than the ratio of the second one (Step 5), we fill the current knapsack with $\bar{\mu}_1\delta_1$ units of the first item (Steps 6) and we update the remaining upper bound and the residual capacity (Step 7). Assume by the moment that the sampling points corresponding to μ_2 and μ_3 remain feasible. At the second iteration, the second item is taken into account and the knapsack is filled with $\bar{\mu}_2\delta_2$ units, where $\bar{\mu}_2$ is the analogue of $\bar{\mu}_1$ for the second item and, again, the upper bound and the capacity are updated, and so on. If instead (Step 8) for at least one of the next two items, say 2 and 3, the sampling

point corresponding to μ_2 or μ_3 is infeasible, an update of the μ values is performed on items 2, 3, ... (and, consequently, the item order might change).

Remark 4.4.1. *Note that, whenever a partial solution x_{ij} is determined, the ratios of all the unscanned items might be re-calculated, but it is not necessary to re-sort all the items: in fact, the algorithm only needs the items with the first and the second best ratio, which can be found in linear time.* □

The algorithm stops when only one element remains unscanned and, in this case, tries to fill the current knapsack with the last item as much as possible (Step 12).

Algorithm 5 Procedure Construct(i).

1: $\bar{c}_i := c_i$;
2: $\bar{u}_j := u_j$;
3: $j := 1$;
4: **while** $j < n$ **and** $\bar{c}_i > 0$ **do**
5: $\quad \bar{\mu}_j := \max\{k : r_{jk} \geqslant r_{(j+1)\mu_{j+1}}, \mu_j \leqslant k \leqslant s\}$;
6: $\quad x_{ij} := \bar{\mu}_j \, \delta_j$;
7: $\quad \bar{u}_j := \bar{u}_j - x_{ij}$, $\bar{c}_i = \bar{c}_i - g_j(x_{ij})$;
8: $\quad$ **if** $(g_{j+1}(\mu_{j+1} \, \delta_{j+1}) > \bar{c}_i$ or $g_{j+2}(\mu_{j+2} \, \delta_{j+2}) > \bar{c}_i)$ **then** update μ for items $j+1, j+2, \ldots$ (and possibly update their order);
9: $\quad j := j + 1$;
10: **end while**
11: **if** $\bar{c}_i > 0$ **then** {**comment:** fill the residual capacity with item n}
12: $\quad x_{in} := \min(g_n^{-1}(\bar{c}_i), \bar{u}_n)$;
13: $\quad \bar{u}_n = \bar{u}_n - x_{in}$, $\bar{c}_i = \bar{c}_i - g_n(x_{in})$;
14: **end if**

We simply assume that the weight functions g_j are strictly increasing and continuous, so that the inverse g_j^{-1} exists (Step 12). If it is not the case, we consider the pseudo-inverse of g_j with the largest value of the pre-image. In order to compute this value, we only need to calculate the zeros of the function $g_j(x_{ij}) - \bar{c}_i$, which can be evaluated in a CPU time bounded by a constant independent from the instance size.

The algorithm can be improved through a refined search for $\bar{\mu}_j$. Once it has been obtained (at Step 5), the interval $[\bar{\mu}_j, \bar{\mu}_{j+1}]$ can be searched with a smaller sampling step and new, more precise, profit-to-weight ratios for item j can be computed. In this way a more precise point $\bar{\mu}_j$ is obtained, and the process can be iterated by further decreasing the sampling step.

Steps 5–9 are iterated at most n times. At each time we have to determine the items with the first and the second best ratio, which can be calculated in $\mathcal{O}(n)$, and the (pseudo-)inverse of the function g_j, which can be effectuated,

as said before, in a computational time bounded by a constant. The main loop is executed up to n times (Step 4). If the refinement parameters, i.e., the number of sampling step and the number of the refinements are bounded by a constant (as usual in practice), the time complexity of Construct(i) is $\mathcal{O}(n^2)$.

Procedure Construct(i) is therefore executed for each knapsack i by considering, at each iteration, only those items whose quantities are still smaller than the upper bound (see Procedure Constructive). At the end, a greedy heuristic is applied to fill the knapsack as much as possible with the current item (Step 5). The overall time complexity of Procedure Constructive(i) is $\mathcal{O}(mn^2)$.

Algorithm 6 Procedure Constructive.

1: **for** $i := 1$ **to** m **do** Construct(i) {**comment:** optionally include the refined search};
2: **for** $i := 1$ **to** m **do**
3: **if** $\bar{c}_i > 0$ **then**
4: **for** $j := 1$ **to** n **do** {**comment:** increase x_{ij} as much as possible}
5: $x_{ij} := x_{ij} + \min(g_j^{-1}(\bar{c}_i), \bar{u}_j)$;
6: **if** $j \in \overline{N}$ **then** $x_{ij} := \lfloor x_{ij} \rfloor$;
7: $\bar{u}_j := u_j - \sum_{k \in M} x_{kj}$, $\bar{c}_i := c_i - \sum_{k \in N} g_j(x_{ik})$;
8: **end for**
9: **end if**
10: **end for**

4.4.2 Surrogate Heuristics

In this section we introduce two different heuristic procedures based on the feasibility recovery of the solution of the surrogate relaxation (4.11). Let us first consider the problem of determining good surrogate multipliers π. A series of preliminary experiments was performed on the benchmark instances adopted for the computational experiments of Section 4.4.5, with

(i) π_i uniformly random in $[0.0, 3.0]$ for all $i \in M$;

(ii) π_i uniformly random in $[0.8, 1.2]$ for all $i \in M$;

(iii) π_i uniformly random in $[0.9, 1.1]$ for all $i \in M$,

and

(iv) $\pi_i = 1$ for all $i \in M$.

It turned out that the surrogate solutions produced by (i) were dominated by the other generations, those produced by (ii) and (iii) had about the same quality, and those produced by (iv) were, on average, clearly the best ones. Additional tests were performed using (easier) convex and concave objective functions, globally obtaining the same results. It was thus decided to always

adopt option (iv). In Section 4.3 we have shown that identical multipliers (optimal solution of the surrogate dual for the linear case) are not necessary optimal for the nonlinear case. It is worth observing that they appear to be a good choice for such case too, at least for the objective functions we considered.

Let x^*_{ij} be the surrogate solution. We assume, as in Section 4.4.1, that the knapsacks are sorted in non-increasing order according to their capacities and the items are sorted in non-increasing order according to the profit-to-weight ratios evaluated over the surrogate solution, i.e.,

$$r_j = \frac{\sum_{i \in M} f_j(x^*_{ij})}{\sum_{i \in M} g_j(x^*_{ij})} \qquad (j \in N). \tag{4.15}$$

The first heuristic (see Procedure Surrogate-feas-1(x^*)) starts with a zero solution – note that this solution is always feasible for the MNLKP – and is divided into two main phases.

Algorithm 7 Procedure Surrogate-feas-1(x^*).

1: **for** $j := 1$ **to** n **do** $\bar{u}_j := u_j$;
2: **for** $i := 1$ **to** m **do**
3: $\quad \bar{c}_i := c_i$;
4: $\quad$ **for** $j := 1$ **to** n **do** $x_{ij} := 0$;
5: **end for**
6: **for** $i := 1$ **to** m **do**
7: $\quad$ **for** $j := 1$ **to** n **do**
8: $\quad\quad$ **if** $g_j(x^*_{ij}) \leqslant \bar{c}_i$ **then** $x_{ij} := x^*_{ij}$, $\bar{c}_i := \bar{c}_i - g_j(x^*_{ij})$, $\bar{u}_j := \bar{u}_j - x^*_{ij}$;
9: $\quad$ **end for**
10: **end for**
11: $\bar{\imath} = \arg\max_{i \in M}\{\bar{c}_i\}$, $\bar{c}_{max} := \bar{c}_{\bar{\imath}}$, $\bar{u}_{max} := \max_{j \in N}\{\bar{u}_j\}$;
12: **while** $\bar{c}_{max} > 0$ **and** $\bar{u}_{max} > 0$ **do**
13: $\quad$ **for** $j := 1$ **to** n **do**
14: $\quad\quad \bar{r}_j := \begin{cases} 0 & \text{if } \bar{u}_j = 0 \\ \frac{f_j(\min(x_{\bar{\imath}j}+\bar{u}_j, \max(0, \lfloor g_j^{-1}(\bar{c}_{\bar{\imath}}+g_j(x_{\bar{\imath}j}))\rfloor)))}{g_j(\min(x_{\bar{\imath}j}+\bar{u}_j, \max(0, \lfloor g_j^{-1}(\bar{c}_{\bar{\imath}}+g_j(x_{\bar{\imath}j}))\rfloor)))} & \text{if } \bar{u}_j > 0 \text{ and } i \in \bar{N}; \\ \frac{f_j(\min(x_{\bar{\imath}j}+\bar{u}_j, \max(0, g_j^{-1}(\bar{c}_{\bar{\imath}}+g_j(x_{\bar{\imath}j})))))}{g_j(\min(x_{\bar{\imath}j}+\bar{u}_j, \max(0, g_j^{-1}(\bar{c}_{\bar{\imath}}+g_j(x_{\bar{\imath}j})))))} & \text{if } \bar{u}_j > 0 \text{ and } i \notin \bar{N} \end{cases}$
15: $\quad$ **end for**
16: $\quad \bar{\jmath} = \arg\max_{j \in N}\{\bar{r}_j\}$;
17: $\quad x_{\bar{\imath}\bar{\jmath}} := \min(x_{\bar{\imath}\bar{\jmath}} + \bar{u}_{\bar{\jmath}}, \max(0, g_{\bar{\jmath}}^{-1}(\bar{c}_{\bar{\imath}} + g_{\bar{\jmath}}(x_{\bar{\imath}\bar{\jmath}}))))$;
18: $\quad$ **if** $\bar{\jmath} \in \overline{N}$ **then** $x_{\bar{\imath}\bar{\jmath}} := \lfloor x_{\bar{\imath}\bar{\jmath}} \rfloor$;
19: $\quad \bar{u}_{\bar{\jmath}} := u_{\bar{\jmath}} - \sum_{i \in M} x_{i\bar{\jmath}}$, $\bar{c}_{\bar{\imath}} := c_{\bar{\imath}} - \sum_{j \in N} g_j(x_{\bar{\imath}j})$;
20: $\quad \bar{\imath} = \arg\max_{i \in M}\{\bar{c}_i\}$, $\bar{c}_{max} := \bar{c}_{\bar{\imath}}$, $\bar{u}_{max} := \max_{j \in N}\{\bar{u}_j\}$;
21: **end while**

In the first phase (Steps 6-10) the current knapsack is filled with the surrogate solution if no relaxed constraint is violated (see Step 8). Since the items

are sorting according to their profit-to-weight ratios, the current knapsack is filled at first with the more promising items.

In the second phase we identify the knapsack $\bar{\imath}$ with the largest residual capacity (Step 11), we update the profit-to-weight ratios (Steps 13-15), and we determine the item $\bar{\jmath}$ with the best *residual* profit-to-weight ratio (Step 16). Knapsack $\bar{\imath}$ is then filled as much as possible with item $\bar{\jmath}$ (Steps 17-18). We iteratively update the upper bound and the residual capacity (Step 19), and we repeat the previous steps, as long as the largest residual capacity or the largest residual upper bound are strictly positive.

The two **for** loops 2-5 and 6-10 are executed mn times. The **while** loop is executed at most $\max(m, n)$ times. Moreover, if we assume, as in Section 4.4.1, that computing the (pseudo-)inverse of the weight function g_j takes a time bounded by a constant independent from the input size, each **for** loop 13-15 takes n times. Hence, the overall time complexity of Procedure Surrogate-feas-1(x^*) is $\mathcal{O}(mn + n^2)$.

In short, the Procedure Surrogate-feas-1(x^*) starts with an empty solution and tries to construct a feasible solution that replicates as much as possible the (infeasible) surrogate solution. Procedure Surrogate-feas-2(x^*) starts instead with the surrogate solution and considers the items in reverse order, i.e., according to *non-decreasing* r_j values (see (4.15)). The idea is to reduce the quantity x^*_{ij} of each item j in the current knapsack i, until the capacity constraints (4.1b) are satisfied (Step 5-6). This is obtained by iteratively reducing the quantity of the item with the worst profit-to-weight ratio, so as to undermine as little as possible the quality of the surrogate solution.

Remark 4.4.2. *The surrogate solution already satisfies the upper bound constraints* (4.1c)*, and therefore, if we reduce the units* x^*_{ij}*, the new solution still meets the constraints* (4.1c). □

The **for** loop 5-8 is executed mn times. If, we assume, as above, that time to compute the (pseudo-)inverse of function g_j can be bounded by constant independent from the input size, the overall time complexity of Procedure Surrogate-feas-2(x^*) is $\mathcal{O}(mn)$.

4.4.3 Local Search

In this section we introduce a post-processing local search procedure in order to improve the quality of the solution found by the heuristics. The local search implements pairwise exchanges of the amounts of items assigned to the same knapsack i.

For a given knapsack i, the local search considers a pair of items j and k and applies two small variations to them. Let ε be a new incremental step *smaller* than δ_j and δ_k (see Section 4.4.1). The local search simultaneously

Algorithm 8 Procedure Surrogate-feas-2(x^*).

for $i := 1$ **to** m **do for** $j := 1$ **to** n **do** $x_{ij} := x^*_{ij}$;
for $i := 1$ **to** m **do**
  $\bar{c}_i := \max(0, \sum_{j \in N} g_j(x_{ij}) - c_i)$;
  for $j := n$ **back to** 1 **do**
    $x_{ij} := \max(0, g_j^{-1}(g_j(x_{ij}) - \bar{c}_i))$;
    if $j \in \overline{N}$ **then** $x_{ij} := \lfloor x_{ij} \rfloor$;
    $\bar{c}_i := \max(0, \sum_{j \in N} g_j(x_{ij}) - c_i)$;
    if $\bar{c}_i \leqslant 0$ **then** break;
  end for
end for

increases (resp. decreases) x_{ij} and decreases (resp. increases) x_{ik} by ε units, respectively. Then, it computes the following variations for the objective function:

1. $\Delta^1 := (f_j(x_{ij} + \varepsilon) - f_j(x_{ij})) + (f_k(x_{ik} - \varepsilon) - f_k(x_{ik}))$;

2. $\Delta^2 := (f_j(x_{ij} - \varepsilon) - f_j(x_{ij})) + (f_k(x_{ik} + \varepsilon) - f_k(x_{ik}))$.

Further impose that a Δ^ℓ ($\ell = 1, 2$) takes the value 0 if the corresponding variation is infeasible, i.e., if either a right-hand side (u_j, u_k, or c_i) of inequalities (4.1b)-(4.1c) is exceeded or one of the two variables takes a negative value. Let $\Delta = \max(\Delta^1, \Delta^2)$:

- if $\Delta > 0$ the procedure, shown in Algorithm 9,
  (i) performs the corresponding variation, producing a new solution with objective function value increased by Δ;
  (ii) iterates the process, for the same couple of items and ε, obviously by only computing the Δ^ℓ ($\ell = 1$ or 2) that produced Δ;
- if instead $\Delta \leqslant 0$, i.e., both variations either worsen the solution value or are infeasible, the next couple of items is tested, or the next knapsack is considered (when $j = n - 1$ and $k = n$).

Preliminary computational experiments are conducted by varying the selection procedure for the knapsack i and for the items j and k: for instance, we select the knapsack and the couple of items randomly. However, the deterministic version produces the most satisfactory results.

Algorithm 9 Procedure Local Search.

for $i := 1$ **to** m **do**
  for $j := 1$ **to** $n - 1$ **do**
    for $k := j + 1$ **to** n **do**
      define an appropriate value $\varepsilon < \min(\delta_j, \delta_k)$;
      repeat
        compute Δ^1, Δ^2, and Δ;
        if $\Delta > 0$ **then** apply the variation corresponding to Δ;
      until $\Delta > 0$
    end for
  end for
end for

The inner **repeat-until** loop is executed mn^2 times. This loop can theoretically take a pseudo-polynomial time but, in practice, it is executed for a number of times bounded by a constant independent from the input size. Over the 3,360 instances we tested the algorithm (see Section 4.4.5), the number of iterations was normally between 1 and 2, and it never attained 10. The overall computational time of the local search is $O(mn^2)$ time.

4.4.4 Overall Algorithm

The overall heuristic algorithm for the MNLKP can be stated as follows:

Algorithm 10 DMM.

1: solve the surrogate relaxation (4.11) and let U and x^* be respectively the resulting upper bound and the corresponding solution;
2: execute procedure Constructive of Section 4.4.1 and let Z^h be the solution value;
3: **if** $Z^h = U$ **then** terminate;
4: execute procedures Surrogate-feas-1(x^*) and Surrogate-feas-2(x^*) of Section 4.4.2 and let Z^s be the best solution value;
5: **if** $Z^s = U$ **then** terminate **else** $Z := \max(Z^h, Z^s)$;
6: execute Local Search of Section 4.4.3 on the solution corresponding to Z and let Z^l be the solution value;
7: **return** the solution corresponding to Z^l.

4.4.5 Computational Experiments

In order to evaluate the computational behavior of the algorithms of Section 4.4.4, we compared it with open-source local solvers for nonlinear programming (Ipopt [126] for real instances and Bonmin [29], with option `bonmin.algorithm B-BB`, for integer instances) and with a global solver (Couenne [59], both for real and integer instances). Ipopt and Bonmin are exact solvers for convex NLPs and convex MINLPs, respectively: hence, they can be used

as heuristics for non-convex problems like MNLKPs.

The first test-bed with respect to which we computationally test the heuristic algorithm is the same as in Section 4.2.3. Furthermore, we consider a second test-bed, characterized by linear weight functions, namely

$$g_j(x_{ij}) = w_j x_{ij}, \tag{4.16}$$

with w_j uniformly random in $[1, 100]$.

The number of items n took the value in $\{10, 20, 50, 100, 200, 500, 1000\}$, while, as in the Section 4.2.3, the number of knapsacks m took the value in $\{2, 5, 10\}$. For each test-bed, we generate two groups of instances *Similar capacities* and *Dissimilar capacities* (see Section 4.2.3). For each combination of (n, m), 20 real instances, i.e., with $\overline{N} = \emptyset$, and 20 integer instances, i.e., with $\overline{N} = N$, are generated. The total number of tested instances was thus 3,360. All the experiments were performed on an Intel Core 2, CPU 6600, 2.4 GHz, 1.94 GB ram, using only one processor.

We ran all the algorithms (Ipopt, Bonmin, Couenne, and DMM) with a time limit of one CPU hour per instance. Couenne was executed with its default values (with an exception mentioned at the end of the present section) as the use of other options strongly increases its computing times. Couenne was executed only for $n \leqslant 50$, since already in those smaller instances its performances are quite poor. For DMM, we compute the solution of the surrogate relaxation (4.11) by means of Couenne with a time limit of $n/10$ seconds. If Couenne did not find a feasible solution within the time limit, we did not execute the surrogate heuristics. Moreover, the time limit of each local search (Step 6) is 5 CPU seconds.

We compare the heuristic against Bonmin and Ipopt with one and ten random starting points. We also tested Bonmin with ten random starting points at each branch-and-bound decision node, but this only resulted in few improvements for the small instances, and, however, in high computational time, so we do not report the corresponding computational results, which, however, can be found in the technical report [186].

Preliminary computational tests were conducted by using Scip [223], as exact solver for non-convex MINLPs, resulting in significantly worse performances than the ones obtained by Couenne.

Procedure Construct(i) (within Constructive) was executed for $s \in \{1, 10, 50, 100\}$, and the best solution was selected. The refined search for $\bar{\mu}_j$ (see Section 4.4.1) was obtained: (i) by trying up to 5 consecutive refinement rounds, each time dividing the current sampling step by 2; (ii) by trying a single refinement round twice (dividing the initial sampling step by 5 and 10, respectively), and (iii) taking the best solution. We set the value of ε

to $\min(\delta_j, \delta_k)/2$. All computations of the zero of a weight function needed by DMM were performed through a binary search over the definition range. (The impact on the overall CPU time was however negligible.)

Tables 11-12 report the results for nonlinear weights, with real and integer variables x_{ij} for similar capacities. The entries are, for real variables: the average values produced by DMM, Ipopt with a single starting point (Ipopt_1), Ipopt with 10 starting points (Ipopt_10), and Couenne, and for integer variables: the average values produced by DMM, Bonmin with a single starting point (Bonmin_1), Bonmin with 10 starting points at the root node (Bonmin_10), and Couenne. For the group of instances, for which the solvers do not succeed in finding a feasible solution within the time limit, the tables report also in brackets the number of non-solved instances.

Tables 13-14 report the same statistic for dissimilar capacities. The tables with odd numbering present the average solution values produced over the 20 generated instances, while those with even numbering report the corresponding average CPU times (in seconds).

Tables 15-18 report the same information for the case of linear weights for similar and dissimilar capacities.

Tables 11 and 13 clearly show that, on the nonlinear instances, a part for few smaller instances with $n = 10$ (and a single case for $n = 20$), the proposed algorithm almost always outperforms both the exact and the heuristic solvers. Tables 12 and 14 show that on the integer instances DMM is always the fastest method with regard to the elapsed CPU time. For the real instances Ipopt_1 is generally faster, but the solution values it produces are definitely worse (by over 10% on average). Overall, DMM seems to be a reasonable algorithm with respect to the trade-off between the quality of the produced solution value and the computational time.

For the instances with linear weight functions, Tables 15 and 17 show that Ipopt_10 and Bonmin_10 often provide better solutions for smaller instances, while, instead, DMM always performs better for $n \geqslant 200$. Concerning the average CPU times (Tables 16 and 18), DMM is again the clear winner on the integer instances. For the real instances, Ipopt_1 is faster for $m = 2$ (but it produces on the other hand worse solution values), while DMM always outperforms the open-source solvers for $m \geqslant 5$ (the difference is particularly high on the larger instances).

It turns out that the instances with linear weights are more difficult to solve to optimality than those with nonlinear weights. Although this can appear surprising, there is no theoretical result implying that one case must be easier than the other. Couenne, for example, transforms the objective function so as it becomes linear, while its nonlinear terms become additional

Table 11: MNLKP, nonlinear weights, similar capacities. Average solution values over 20 instances (# no solution).

		Real Variables				Integer Variables			
m	n	DMM	Ipopt_1	Ipopt_10	Couenne	DMM	Bonmin_1	Bonmin_10	Couenne
2	10	350.49	313.50	351.70	362.63	350.47	327.21	313.87	364.45
2	20	641.73	569.20	635.35	593.43(12)	645.63	591.26	589.88	561.83
2	50	1,900.41	1,714.74	1,799.78	n/a(20)	1,902.65	1,769.76	1,800.63	1,481.36
2	100	3,741.92	3,341.04	3,548.40	–	3,742.14	3,413.98	3,454.54	–
2	200	7,167.80	6,302.04	6,683.39	–	7,168.07	6,429.80(5)	6,419.05	–
2	500	18,375.35	16,153.10	16,942.30	–	18,376.30	16,286.80(13)	16,404.90(5)	–
2	1000	37,051.16	32,117.80	33,899.30	–	37,058.30	n/a(20)	n/a(20)	–
2	total	69,228.86	60,511.42	63,860.22	–	69,243.56	–	–	–
5	10	322.76	271.61	312.63	299.06(5)	321.18	288.19	281.05	299.35
5	20	729.46	687.95	733.25	n/a(20)	727.24	706.88	716.23	620.54(2)
5	50	1,822.58	1,705.86	1,782.99	n/a(20)	1,821.57	1,753.76(2)	1,765.72	1,245.52(1)
5	100	3,865.64	3,701.10	3,818.14	–	3,868.62	3,719.86(1)	3,825.23	–
5	200	7,846.03	7,372.56	7,681.43	–	7,850.18	n/a(20)	7,677.54(16)	–
5	500	19,272.60	18,180.80	18,825.70	–	19,273.73	n/a(20)	n/a(20)	–
5	1000	38,540.16	36,391.00	37,485.80	–	38,541.14	n/a(20)	n/a(20)	–
5	total	72,399.23	68,310.88	70,639.94	–	72,403.66	–	–	–
10	10	216.19	187.40	218.59	212.07(2)	232.58	183.37(2)	201.60(1)	222.29
10	20	734.75	664.58	702.21	n/a(20)	727.85	719.17(3)	681.76(2)	489.35(5)
10	50	1,983.10	1,864.39	1,927.13	n/a(20)	1,983.72	1,834.39(16)	1,937.89	1,275.43(16)
10	100	3,952.78	3,732.51	3,843.88	–	3,957.84	n/a(20)	3,852.28(19)	–
10	200	7,652.05	7,311.37	7,431.36	–	7,655.04	n/a(20)	n/a(20)	–
10	500	19,640.75	18,788.40	19,097.90	–	19,645.34	n/a(20)	n/a(20)	–
10	1000	39,717.69	36,316.50	38,266.70	–	39,721.93	n/a(20)	n/a(20)	–
10	total	73,897.31	68,865.15	71,487.77	–	73,924.30	–	–	–
total	total	215,525.40	197,687.45	205,987.93	–	215,571.52	–	–	–

Table 12: MNLKP, nonlinear weights, similar capacities. Average CPU times over 20 instances (# no solution).

		Real Variables				Integer Variables			
m	n	DMM	Ipopt_1	Ipopt_10	Couenne	DMM	Bonmin_1	Bonmin_10	Couenne
2	10	1.08	0.05	0.51	1,113.88	1.09	3.71	1.47	847.17
2	20	2.02	0.10	1.18	3,601.81(12)	2.03	15.58	4.53	3,601.20
2	50	5.08	0.34	3.45	n/a(20)	5.11	637.51	78.25	3,603.19
2	100	10.23	0.93	10.24	–	10.26	2,363.94	978.34	–
2	200	20.88	2.44	28.34	–	20.72	3,089.98(5)	2,693.71	–
2	500	57.45	10.60	112.08	–	54.14	3,600.48(13)	3,496.83(5)	–
2	1000	113.54	32.84	327.93	–	116.11	n/a(20)	n/a(20)	–
2	total	210.28	47.30	483.73	–	209.46	–	–	–
5	10	1.20	0.16	1.72	3,600.61(5)	1.24	60.58	8.34	3,600.88
5	20	2.03	0.32	3.57	n/a(20)	2.05	601.01	185.12	3,602.25(2)
5	50	5.15	1.17	11.76	n/a(20)	5.18	2,765.24(2)	996.16	3,602.47(1)
5	100	10.25	3.53	31.76	–	10.41	3,071.08(3)	3,419.49	–
5	200	20.79	9.72	87.03	–	21.35	n/a(20)	3,600.40(16)	–
5	500	54.26	51.12	403.15	–	57.29	n/a(20)	n/a(20)	–
5	1000	125.86	151.13	1,207.48	–	126.70	n/a(20)	n/a(20)	–
5	total	219.54	217.15	1,746.47	–	224.22	–	–	–
10	10	1.49	0.39	3.78	3,600.54(2)	1.54	827.80(2)	181.15(1)	3,600.17
10	20	2.14	0.93	8.57	n/a(20)	2.17	2,570.88(3)	1059.40(2)	3,602.81(5)
10	50	5.14	3.35	30.70	n/a(20)	5.31	3,601.42(16)	3,464.19	3,665.76(16)
10	100	10.37	10.38	86.37	–	10.94	n/a(20)	3,600.19(19)	–
10	200	23.51	32.98	278.67	–	23.43	n/a(20)	n/a(20)	–
10	500	87.11	147.12	1,197.91	–	86.49	n/a(20)	n/a(20)	–
10	1000	375.57	348.52	3,108.31	–	371.94	n/a(20)	n/a(20)	–
10	total	505.33	543.67	4,714.31	–	501.82	–	–	–
total	total	935.15	808.12	6,944.51	–	935.50	–	–	–

Table 13: MNLKP, nonlinear weights, dissimilar capacities. Average solution values over 20 instances (# no solution).

		Real Variables				Integer Variables			
m	n	DMM	Ipopt_1	Ipopt_10	Couenne	DMM	Bonmin_1	Bonmin_10	Couenne
2	10	340.62	298.24	339.50	355.19	339.89	299.21	308.60	354.62
2	20	634.68	547.68	610.20	564.65(9)	636.54	569.67	573.25	555.90
2	50	1,870.83	1,619.80	1,737.00	n/a(20)	1,871.45	1,688.48	1,653.26	1,362.62
2	100	3,722.84	3,236.18	3,437.91	–	3,722.43	3,340.35(1)	3,294.41	–
2	200	7,074.48	5,908.90	6,313.01	–	7,072.53	6,188.82(3)	5,905.41	–
2	500	18,280.25	15,286.70	16,081.20	–	18,287.28	15,883.50(14)	15,862.40(5)	–
2	1000	37,089.65	31,168.30	32,825.20	–	37,086.84	n/a(20)	n/a(20)	–
2	total	69,013.35	58,065.80	61,344.02	–	69,016.96	–	–	–
5	10	321.28	285.28	318.68	329.51(1)	313.70	305.05(1)	299.28(1)	319.22(1)
5	20	728.42	670.57	717.91	659.84(17)	720.97	692.01(2)	664.84	598.87(5)
5	50	1,792.76	1,614.39	1,723.24	n/a(20)	1,772.33	1,600.30(4)	1,627.85	1,228.18(4)
5	100	3,810.80	3,392.78	3,625.14	–	3,805.91	3,406.69(2)	3,400.23	–
5	200	7,647.44	6,778.96	7,180.81	–	7,595.80	n/a(20)	n/a(20)	–
5	500	19,108.46	17,131.00	17,967.30	–	19,100.00	n/a(20)	n/a(20)	–
5	1000	38,447.52	34,559.20	36,099.60	–	38,472.11	n/a(20)	n/a(20)	–
5	total	71,856.68	64,432.18	67,632.68	–	71,780.82	–	–	–
10	10	343.95	314.88	345.18	354.52	321.47	315.20(1)	307.11	340.21(1)
10	20	757.28	678.59	741.04	699.31(11)	737.34	703.57(3)	681.37(1)	633.95(8)
10	50	1,972.11	1,807.00	1,909.65	n/a(20)	1,883.62	1,817.64(9)	1,810.51(1)	1,499.67(7)
10	100	3,873.07	3,441.87	3,684.99	–	3,615.12	n/a(20)	3,804.82(19)	–
10	200	7,543.03	6,708.00	7,094.40	–	7,395.62	n/a(20)	n/a(20)	–
10	500	19,271.73	17,027.60	17,950.80	–	19,242.96	n/a(20)	n/a(20)	–
10	1000	39,486.36	35,248.30	37,237.50	–	39,403.01	n/a(20)	n/a(20)	–
10	total	73,247.53	65,226.24	68,963.56	–	72,599.14	–	–	–
total	total	214,117.56	187,724.22	197,940.26	–	213,396.92	–	–	–

Table 14: MNLKP, nonlinear weights, dissimilar capacities. Average CPU times over 20 instances (# no solution).

		Real Variables				Integer Variables			
m	n	DMM	Ipopt_1	Ipopt_10	Couenne	DMM	Bonmin_1	Bonmin_10	Couenne
2	10	1.08	0.05	0.53	794.39	1.09	2.91	1.11	364.44
2	20	2.02	0.11	1.14	3,307.95(9)	2.03	20.41	8.18	3,430.29
2	50	5.08	0.34	3.36	n/a(20)	5.30	277.57	69.66	3,602.78
2	100	10.23	0.89	9.96	–	10.26	1,731.33(1)	813.99	–
2	200	20.87	2.40	26.51	–	20.71	3,254.23(3)	3,099.98	–
2	500	57.34	10.64	105.76	–	54.07	3,265.7(14)	3,456.18(5)	–
2	1000	113.39	33.71	318.10	–	115.93	n/a(20)	n/a(20)	–
2	total	210.01	48.14	465.36	–	209.39	–	–	–
5	10	1.20	0.14	1.53	2,790.28(1)	1.23	29.11(1)	196.66(1)	2,013.34(1)
5	20	2.03	0.34	3.23	3,600.18(17)	2.05	250.53(2)	588.38	3,600.92(5)
5	50	5.16	1.16	10.25	n/a(20)	5.17	2,716.96(4)	773.89	3,604.77(4)
5	100	10.25	3.04	26.45	–	10.41	3,373.18(2)	2,341.63	–
5	200	20.77	8.70	74.45	–	21.25	n/a(20)	n/a(20)	–
5	500	54.14	45.04	356.04	–	57.02	n/a(20)	n/a(20)	–
5	1000	124.84	130.89	1,044.09	–	126.05	n/a(20)	n/a(20)	–
5	total	218.39	189.31	1,516.04	–	223.18	–	–	–
10	10	1.49	0.31	2.62	2,266.36	1.53	33.25(1)	53.09	1,753.20(1)
10	20	2.16	0.79	6.10	3,599.61(11)	2.16	650.11(3)	631.44(1)	3,600.00(8)
10	50	5.14	2.89	23.01	n/a(20)	5.28	3,177.66(9)	2,326.35(1)	3,597.48(7)
10	100	10.35	7.98	58.18	–	10.84	n/a(20)	3,600.9(19)	–
10	200	23.44	25.93	184.13	–	23.22	n/a(20)	n/a(20)	–
10	500	86.92	113.32	704.57	–	86.07	n/a(20)	n/a(20)	–
10	1000	374.20	328.10	2,268.40	–	370.22	n/a(20)	n/a(20)	–
10	total	503.70	479.32	3,247.01	–	499.32	–	–	–
total	total	932.10	716.77	5,228.41	–	931.89	–	–	–

constraints. It follows that, in any case, the feasible region is defined by nonlinear constraints, and the evolution of the branching search is unpredictable.

For all the real instances, all the algorithms (DMM, Ipopt_1, and Ipopt_10) are able to provide a feasible solution within the time limit (both for the case of linear and nonlinear weights); while, for the larger integer instances, DMM is the only heuristic with can determine a feasible solution within the time limit.

Tables 19 and 20 compare the solution values produced by DMM and Couenne, for those instances, with respect to which Couenne is able to find the global maximum. Table 19 reports the results for nonlinear weights and both similar and dissimilar capacities, while Table 20 is the analogue for linear weights. DMM computes solutions with an average error of 3.65%, with a minimum of 0% and a maximum of 18% with regards to the global optimum. We observe that Couenne optimally solved more instances with nonlinear weights (5.6%) than with linear weights (1.8%).

Finally, for 10 of the instances globally solved by Couenne, it turned out that the software behaved incorrectly, finding a solution value lower that the one produced by DMM. Hence, at the suggestion of Couenne developer, we re-execute the solver by disabling several default options, namely: `aggressive_fbbt`, `optimality_bt`, and `redcost_bt`. For instance, the solution value of the instance 6 with $m = 5$, $n = 10$, dissimilar capacities, integer variables and nonlinear weights, produced by Couenne with the default options was 286.86, while DMM provide a feasible solution of value 294.76. We re-run Couenne without the suggested options and the solution value was 305.67.

4.5 CONCLUSIONS

In this chapter we considered the MNLKP and we discussed the possible relaxations for this problem. We implemented the MWU algorithm, adapting its main steps to this class of problems. However, computational experiments illustrated that this methodology was not successful in this situation; hence, we proposed different constructive heuristic strategies followed by a local search post-processing. Extensive computational tests clearly showed that this approach outperformed the other heuristic and exact algorithms available for the MNLKP both in terms of solution quality and of CPU time elapsed.

Table 15: MNLKP, linear weights, similar capacities. Average solution values over 20 instances (# no solution).

		Real Variables				Integer Variables			
m	n	DMM	Ipopt_1	Ipopt_10	Couenne	DMM	Bonmin_1	Bonmin_10	Couenne
2	10	343.63	318.66	350.62	343.65(6)	343.95	332.91	351.35	335.53
2	20	780.54	739.70	786.05	n/a(20)	780.03	752.42	784.46	676.70
2	50	1,943.44	1,841.93	1,921.17	n/a(20)	1,941.38	1,896.66	1,939.18	1,494.77(10)
2	100	3,873.34	3,585.02	3,727.55	–	3,872.29	3,681.92	3,740.08	–
2	200	7,966.74	7,401.99	7,645.14	–	7,966.32	7,510.38(4)	7,574.34(1)	–
2	500	20,111.10	18,685.00	19,222.20	–	20,107.13	18,438.10(18)	18,438.10(18)	–
2	1000	39,562.06	36,752.90	37,448.70	–	39,563.97	n/a(20)	n/a(20)	–
2	total	74,580.85	69,325.20	71,101.43	–	74,575.07	–	–	–
5	10	374.58	364.54	392.81	243.95(1)	375.28	366.75	386.39	286.99(2)
5	20	762.51	739.22	782.22	n/a(20)	761.99	750.92	768.77	485.02(4)
5	50	2,010.44	1,990.01	2,038.93	n/a(20)	2,013.13	1,991.96	1,997.33	1,366.98(11)
5	100	3,942.02	3,808.03	3,955.77	–	3,942.90	3,978.83(15)	3,900.36(14)	–
5	200	8,135.86	7,927.23	8,125.15	–	8,141.77	n/a(20)	n/a(20)	–
5	500	20,883.19	20,123.60	20,424.60	–	20,900.04	n/a(20)	n/a(20)	–
5	1000	41,748.25	39,406.90	40,231.00	–	41,755.37	n/a(20)	n/a(20)	–
5	total	77,856.85	74,359.53	75,950.48	–	77,890.48	–	–	–
10	10	240.61	224.04	243.62	211.05(15)	235.48	229.52	238.99	170.29(4)
10	20	790.00	778.73	816.61	n/a(20)	787.61	804.24(1)	807.22	329.71(4)
10	50	2,095.67	2,042.56	2,122.72	1,572.96(18)	2,097.90	2,249.09(19)	2,249.09(19)	486.79(6)
10	100	4,040.29	3,860.72	4,008.91	–	4,043.65	n/a(20)	n/a(20)	–
10	200	8,376.55	8,063.39	8,224.59	–	8,382.21	n/a(20)	n/a(20)	–
10	500	21,309.28	20,091.20	20,535.40	–	21,321.39	n/a(20)	n/a(20)	–
10	1000	42,766.36	36,853.00	39,276.10	–	42,782.19	n/a(20)	n/a(20)	–
10	total	79,618.76	71,913.64	75,227.95	–	79,650.43	–	–	–
total	total	232,056.46	215,598.37	222,279.86	–	232,115.98	–	–	–

Table 16: MNLKP, linear weights, similar capacities. Average CPU times over 20 instances (# no solution).

		Real Variables				Integer Variables			
m	n	DMM	Ipopt_1	Ipopt_10	Couenne	DMM	Bonmin_1	Bonmin_10	Couenne
2	10	1.06	0.04	0.56	2,250.10(6)	1.07	3.67	2.91	3,592.36
2	20	2.02	0.13	1.40	n/a(20)	2.02	16.76	12.94	3,602.05
2	50	5.06	0.50	6.02	n/a(20)	5.08	145.37	195.66	3,602.23(10)
2	100	10.21	1.69	17.64	–	10.22	1,499.39	1,312.30	–
2	200	20.71	5.90	58.31	–	20.81	2,978.50(4)	2,666.15(1)	–
2	500	56.59	26.18	273.87	–	58.50	1,4671.80(18)	11,718.9(18)	–
2	1000	115.69	82.53	857.74	–	118.58	n/a(20)	n/a(20)	–
2	total	211.34	116.97	1,215.54	–	216.28	–	–	–
5	10	1.11	0.13	1.46	3,600.77(1)	1.13	350.92	330.21	3,601.28(2)
5	20	2.03	0.36	3.78	n/a(20)	2.04	550.75	675.76	3,601.41(4)
5	50	5.10	1.41	14.74	n/a(20)	5.14	3,110.13	3,182.53	3,608.20(11)
5	100	10.34	5.72	47.38	–	10.32	3,585.95(15)	3,292.52(14)	–
5	200	20.85	15.15	145.10	–	21.18	n/a(20)	n/a(20)	–
5	500	55.02	82.17	834.01	–	57.23	n/a(20)	n/a(20)	–
5	1000	119.42	229.58	2,357.48	–	128.10	n/a(20)	n/a(20)	–
5	total	213.87	334.52	3,403.95	–	225.14	–	–	–
10	10	1.21	0.34	3.36	3,602.44(15)	1.27	1,088.11	934.47	3,601.56(4)
10	20	2.04	0.81	8.00	n/a(20)	2.06	3,130.17(1)	3,097.31	3,602.08(4)
10	50	5.11	3.46	35.50	3,596.06(18)	5.18	3,583.36(19)	3,417.98(19)	3,610.75(6)
10	100	10.31	13.30	148.41	–	10.57	n/a(20)	n/a(20)	–
10	200	21.09	46.59	497.50	–	22.18	n/a(20)	n/a(20)	–
10	500	56.17	195.58	1,999.56	–	63.64	n/a(20)	n/a(20)	–
10	1000	170.69	398.56	4,219.18	–	163.09	n/a(20)	n/a(20)	–
10	total	266.62	658.64	6,911.51	–	267.99	–	–	–
total	total	691.83	1,110.13	11,531.00	–	709.41	–	–	–

Table 17: MNLKP, linear weights, dissimilar capacities. Average solution values over 20 instances (# no solution).

		Real Variables				Integer Variables			
m	n	DMM	Ipopt_1	Ipopt_10	Couenne	DMM	Bonmin_1	Bonmin_10	Couenne
2	10	333.39	296.56	342.03	342.20(3)	331.75	312.05	339.32	336.78
2	20	760.08	696.13	758.70	674.85(16)	757.24	705.37	763.36	652.22(1)
2	50	1,916.44	1,789.24	1,854.16	n/a(20)	1,920.41	1,866.97	1,887.10	1,407.08(5)
2	100	3,859.95	3,451.60	3,625.93	–	3,864.18	3,614.56	3,696.08	–
2	200	7,856.57	7,044.65	7,263.35	–	7,854.79	7,209.46(2)	7,348.76(1)	–
2	500	19,806.13	17,817.50	18,240.70	–	19,870.13	19,164.10(18)	18,296.70(14)	–
2	1000	39,193.30	35,220.40	35,900.90	–	39,183.30	31,516.00(19)	n/a(20)	–
2	total	73,725.86	66,316.08	67,985.77	–	73,781.80	–	–	–
5	10	394.60	371.40	404.20	434.82(10)	394.12	379.34	401.04	378.30(3)
5	20	749.42	699.44	757.70	600.46(18)	746.29	708.66	739.91	481.27(6)
5	50	1,985.20	1,907.99	1,965.24	n/a(20)	1,986.83	1,953.95	1,978.85	1,030.52(8)
5	100	3,906.82	3,657.64	3,776.51	–	3,899.92	3,657.99(8)	3,733.00(9)	–
5	200	7,991.03	7,535.88	7,739.56	–	8,000.10	6,768.74(19)	7,021.51(18)	–
5	500	20,598.05	19,076.70	19,540.10	–	20,597.93	n/a(20)	n/a(20)	–
5	1000	41,112.47	37,430.50	38,623.60	–	41,144.61	n/a(20)	n/a(20)	–
5	total	76,737.59	70,679.55	72,806.91	–	76,769.80	–	–	–
10	10	308.99	279.20	310.72	313.65(11)	307.06	283.12	304.86	273.38(1)
10	20	816.72	814.43	854.19	n/a(20)	817.87	824.66(1)	844.13	647.17(8)
10	50	2,060.70	1,951.85	2,040.01	n/a(20)	2,064.86	1,865.52(6)	1,921.84(6)	1,117.49(6)
10	100	3,957.53	3,640.15	3,793.31	–	3,956.12	3,426.41(13)	3,711.03(16)	–
10	200	8,287.68	7,549.40	7,859.10	–	8,299.41	n/a(20)	n/a(20)	–
10	500	20,984.56	18,858.90	19,445.60	–	20,898.03	n/a(20)	n/a(20)	–
10	1000	41,989.27	37,114.20	38,332.10	–	41,963.22	n/a(20)	n/a(20)	–
10	total	78,405.45	70,208.13	72,635.03	–	78,306.57	–	–	–
total	total	228,868.90	207,203.76	213,427.71	–	228,858.17	–	–	–

Table 18: MNLKP, linear weights, dissimilar capacities. Average CPU times over 20 instances (# no solution).

		Real Variables				Integer Variables			
m	n	DMM	Ipopt_1	Ipopt_10	Couenne	DMM	Bonmin_1	Bonmin_10	Couenne
2	10	1.06	0.04	0.58	1613.09(3)	1.07	2.24	1.85	2,017.11
2	20	2.02	0.12	1.46	3,602.01(16)	2.02	9.99	10.64	3,601.04(1)
2	50	5.06	0.48	5.78	n/a(20)	5.08	530.30	137.54	3,602.40(5)
2	100	10.21	1.55	16.23	–	10.21	1,699.74	1,571.01	–
2	200	20.69	5.13	52.37	–	20.77	2,655.79(2)	2,601.08(1)	–
2	500	56.61	22.90	237.33	–	58.30	1,493.78(18)	4,318.67(14)	–
2	1000	115.26	70.27	741.25	–	118.16	50,892.30(19)	n/a(20)	–
2	total	210.91	100.49	1,055.00	–	215.61	–	–	–
5	10	1.10	0.14	1.41	3,269.85(10)	1.13	316.61	475.27	3,600.87(3)
5	20	2.02	0.34	3.59	3,601.70(18)	2.04	477.62	793.09	3,601.03(6)
5	50	5.10	1.40	14.21	n/a(20)	5.13	2,356.92	2,872.45	3,602.52(8)
5	100	10.41	4.03	40.96	–	10.30	3,240.82(8)	2,624.47(9)	–
5	200	20.80	11.91	112.03	–	21.12	3,589.85(19)	3,411.551(18)	–
5	500	54.68	58.92	554.51	–	56.87	n/a(20)	n/a(20)	–
5	1000	118.28	192.46	1,736.15	–	126.88	n/a(20)	n/a(20)	–
5	total	212.39	269.20	2,462.86	–	223.47	–	–	–
10	10	1.24	0.28	2.90	2,546.81(11)	1.27	768.94	763.07	3,085.93(1)
10	20	2.04	0.71	7.21	n/a(20)	2.06	1,693.68(1)	2,091.58	3,601.56(8)
10	50	5.10	2.55	28.40	n/a(20)	5.17	2,634.70(6)	2,361.36(6)	3,601.53(6)
10	100	10.27	8.98	92.64	–	10.53	3,211.55(13)	2,752.26(16)	–
10	200	20.94	26.05	295.69	–	22.04	n/a(20)	n/a(20)	–
10	500	55.14	139.96	1,351.50	–	62.25	n/a(20)	n/a(20)	–
10	1000	172.99	350.02	3,629.18	–	159.23	n/a(20)	n/a(20)	–
10	total	267.72	528.55	5,407.52	–	262.55	–	–	–
total	total	691.02	898.24	8,925.38	–	701.63	–	–	–

Table 19: MNLKP, nonlinear weights. Solution values for instances globally solved by Couenne.

			Similar				Dissimilar			
			Real		Integer		Real		Integer	
m	n	instance	DMM	Couenne	DMM	Couenne	DMM	Couenne	DMM	Couenne
2	10	0	223.20	230.69	205.37	228.65	211.03	230.62	211.98	228.33
		1	462.02	470.98	457.49	471.87	–	–	457.04	471.50
		2	338.81	349.23	–	–	340.36	349.29	339.04	349.21
		3	373.14	381.21	368.56	379.98	367.99	382.72	367.83	382.82
		4	309.95	317.61	–	–	309.04	316.53	312.20	316.17
		5	553.69	584.99	553.92	583.28	553.66	587.48	554.02	585.63
		6	–	–	286.42	303.57	250.55	264.43	250.55	263.71
		7	286.71	304.76	285.50	305.06	217.68	221.78	217.83	221.64
		8	–	–	443.08	455.53	–	–	438.36	455.49
		9	–	–	440.18	454.90	–	–	407.35	455.29
		10	314.74	330.25	314.74	326.33	327.94	330.40	322.26	326.86
		11	355.74	396.79	353.70	395.00	374.36	395.14	369.83	393.58
		12	291.93	305.05	294.18	302.82	298.81	299.82	292.96	299.39
		13	328.83	353.59	328.73	352.30	329.20	349.62	329.08	349.50
		14	374.85	390.88	375.30	389.56	361.39	390.92	362.07	389.32
		15	–	–	440.29	440.79	409.60	428.26	411.79	422.74
		16	244.25	244.25	244.25	244.25	244.25	244.25	244.25	244.25
		17	388.22	401.22	391.21	401.16	350.11	352.39	351.32	352.11
		18	358.20	362.85	357.65	362.05	332.11	359.07	330.58	357.38
		19	218.75	227.50	217.83	227.50	227.50	227.50	227.50	227.50
2	20	17	–	–	–	–	525.58	530.82	524.70	530.70
5	10	0	–	–	–	–	–	–	221.56	227.47
		6	–	–	–	–	293.89	306.33	294.76	305.67
		7	–	–	–	–	–	–	293.04	303.33
		8	–	–	–	–	383.54	392.58	367.80	390.95
		10	–	–	–	–	–	–	226.83	232.73
		14	–	–	–	–	302.48	309.32	302.99	308.46
		16	–	–	–	–	238.84	255.36	239.23	254.70
		18	–	–	–	–	259.90	264.89	259.17	264.10
		19	–	–	–	–	476.79	486.37	464.06	483.98
10	10	1	–	–	–	–	449.44	453.39	448.38	452.76
		3	–	–	–	–	261.67	267.20	260.86	266.68
		4	–	–	–	–	–	–	327.73	399.69
		6	–	–	–	–	469.96	486.00	476.02	483.23
		7	–	–	–	–	263.76	266.95	263.76	265.78
		9	–	–	–	–	232.63	234.31	233.57	233.59
		12	–	–	–	–	176.41	178.04	146.23	162.55
		13	–	–	–	–	288.57	288.79	287.89	288.67
		15	–	–	–	–	267.69	274.38	259.68	274.01
		16	–	–	–	–	431.69	437.85	432.11	437.98

Table 20: MNLKP, linear weights. Solution values for instances globally solved by Couenne.

			Similar				Dissimilar			
			Real		Integer		Real		Integer	
m	n	instance	DMM	Couenne	DMM	Couenne	DMM	Couenne	DMM	Couenne
2	10	1	–	–	–	–	213.78	217.12	213.41	217.03
		2	286.74	291.88	286.94	292.11	281.86	291.32	281.86	290.34
		3	269.21	271.67	–	–	262.49	271.61	250.59	271.29
		4	–	–	–	–	–	–	287.51	296.59
		7	–	–	–	–	246.83	250.45	247.66	250.22
		10	279.41	296.13	–	–	282.88	297.53	278.65	297.07
		12	–	–	–	–	462.93	484.80	463.05	484.84
		13	–	–	–	–	452.91	484.65	454.44	485.26
		15	351.95	354.69	–	–	332.08	349.80	332.26	349.28
		17	597.89	626.07	–	–	451.64	488.11	451.23	492.90
		18	–	–	–	–	481.08	494.04	–	–
		19	250.94	252.75	–	–	–	–	–	–
5	10	17	–	–	–	–	481.44	496.10	–	–
10	10	1	–	–	–	–	477.93	516.43	479.13	517.08
		11	–	–	–	–	–	–	363.21	365.08
		13	–	–	–	–	288.66	295.05	288.69	294.78
		19	–	–	–	–	195.25	206.44	–	–

Part IV.

Conclusions

5 CONCLUSIONS

In this thesis we have described a new heuristic algorithm for Mixed Integer NonLinear Problems, based on the general Multiplicative Weights Update algorithm. The contribution of the thesis is twofold: from one side we have theoretically introduced the algorithm framework with a new class of mathematical reformulation, namely the pointwise reformulation, whose theoretical properties are inspected in detail; from the other side we have illustrated two real-world optimization problems, namely the Mean-Variance Portfolio Selection and the Multiple NonLinear Knapsack Problems, with regard to which we practically applied and implemented the algorithm.

In the theoretical part, we analyzed in depth the steps of the algorithm and the general properties of the pointwise reformulation with a particular emphasis on exactness and efficiency. A pointwise reformulation is exact if there exists a parameter such that the global optimality properties of the original formulation are preserved; while a reformulation is efficient if it can be solved in polynomial time. In case the reformulation is exact and efficient, it is sufficient to solve the (easier) reformulation in order to obtain a global solution for the original problem. The Multiplicative Weights Update for Mixed Integer NonLinear Problems is, in fact, a MultiStart type algorithm with a specific choice of the starting points, conducted according to the Multiplicative Weights Update algorithm. Moreover, we described several general building strategies for the pointwise reformulation with respect to given classes of mathematical optimization problems, such as Polynomial, Bilinear, and Quadratic Mixed Integer NonLinear Problems.

In the second part, we examined two different real-world problems and we adapted the general framework to these programs. First, we gave a sufficiently complete survey of the Mean-Variance Portfolio Selection Problems about the ways to model the behaviors of investors and the restrictions of the financial markets, robust and stochastic approaches, exact reformulations and inner approximations, and exact algorithms to solve these challenging problems. Then, we introduced a Multiplicative Weights Update for a general class of portfolio optimization problems, specifying how to generate the pointwise reformulation and how to choose the promising starting points for the original formulation. The second problem we dealt with is the Multiple NonLinear (Separable) Knapsack Problem, characterized by a set of items and a set of knapsacks and whose target consists in filling the knapsacks with units of items in order to maximize a given profit function and meeting knapsack and upper bound constraints. In the general scheme, several variables can be required to be integer. We apply the Multiple Weights Update

to this kind of problems, but the computational behavior of this algorithm was quite poor in terms of performances: in facts, it was almost always overcome by the MultiStart procedure.

Therefore, we considered a constructive heuristic based on three elements: (i) a greedy type heuristic, already proposed for the single knapsack problem, and here extended to the multiple case; (ii) a feasibility recovery strategy which, starting from the empty solution, tries to fill the knapsacks as much as possible with the solution of the surrogate relaxation; (iii) another feasibility recovery procedure which, starting from the surrogate solution, progressively removes the exceeding capacities. Then, we took the best solution produced by procedures (i)-(iii) and ran a local search post-processing. The computational tests indicate the resulting algorithm outperformed heuristic and exact solution methods available for these knapsack problems.

We hope that the Multiplicative Weights Update algorithmic methodology will be applied with success also to other Mixed Integer NonLinear Problems. However, several questions are still open. It could be interesting to consider an analysis of the goodness of a given pointwise reformulation a priori, i.e., a way to theoretically evaluate several formulations and choose between them. We felt that the general approach presented for the portfolio and the knapsack problems could be easily exploited for other problems characterized by separable non-convexities and non-concavities. Moreover, it could be an engaging target defining other automatic strategies to build the pointwise reformulation for other classes of problems and testing them over general Mixed Integer NonLinear Problems Libraries. Therefore, there could be also the possibility to employ several different reformulations at the same time, alternating from one iteration to another the set with respect to which the parameter θ is defined: this could be the case of bilinear optimization problems where the decision variables can be divided into two different disjunctive sets. Finally, we point out that the definition of the pointwise reformulation could be exploited also in more general situations than the Multiplicative Weights Update Algorithm: it generalizes the definition of relaxations to bounding reformulations since by solving them we always obtain a lower bound for the original optimization problem.

Appendix

A NOTATION FOR PORTFOLIO SELECTION

Parameters:

$r \in \mathbb{N}_+$	number of possibly risky assets
$r' \in \mathbb{N}_+$	number of number of different factors
$n \in \mathbb{N}_+$	number of financial categories
$\ell \in \mathbb{N}_+$	number of economic sectors
$R \in \mathbb{N}_+$	minimum return level for the portfolio
$p \in \mathbb{N}_+$	confidence level for the probabilistic return constraint
$B \in \mathbb{N}_+$	total investor initial budget
$\overline{s} \in \mathbb{N}_+$	prescribed minimum level per financial category
$\underline{n} \in \mathbb{N}_+$	minimum number of categories with positive positions
$\underline{K}, \overline{K} \in \mathbb{N}_+$	minimum and maximum number of assets in the portfolio
$\overline{P} \in \mathbb{R}_+$	maximum purchasing level per asset
$\overline{S} \in \mathbb{R}_+$	maximum selling level per asset
$T \in \mathbb{N}_+$	number of time period for observing the returns of quoted assets
$m \in \mathbb{N}_+$	number of factors in the multiple factor model
$\overline{\mu} \in \mathbb{R}^r$	mean return vector of the assets
$\xi \in \mathbb{R}^r$	random vector of expected returns
$\mu \in \mathbb{R}^r$	mean of the r-variate distribution of ξ
$\psi \in \mathbb{R}^r$	normalized portfolio return
$\underline{x}, \overline{x} \in \mathbb{R}^r$	lower and upper bound for the fraction of the portfolio value
$S \in \mathbb{R}^r$	size of the batches of the assets
$q \in \mathbb{R}^r_+$	market value of the quoted assets
$x^{(0)} \in \mathbb{R}^r$	fraction of the portfolio value already invested
$x^B \in \mathbb{R}^r$	benchmark (or target) portfolio
$c \in \mathbb{R}^r$	transaction costs per asset
$v \in \mathbb{R}^r$	costs per unit of asset
$f \in \mathbb{R}^m$	factor-return vector
$u \in \mathbb{R}^r$	asset-specific (non-factor) returns vector
$\Xi \in \mathbb{R}^{r \times r'}$	sensitivity-factor matrix
$\Sigma \in \mathbb{R}^{r \times r}$	covariance matrix of the r-variate distribution of ξ
$\overline{\Sigma} \in \mathbb{R}^{r \times r}$	covariance return matrix of the assets

Sets:

$L_l \subseteq \{1, \ldots, r\}$	set of assets for economic sector l ($l = 1, \ldots, \ell$)
$C_k \subseteq \{1, \ldots, r\}$	set of indexes of all risky assets connected with the category k ($k = 1, \ldots, n$)

Decision Variables:

$x \in \mathbb{R}^r$	(fraction of the) portfolio value invested per asset
$\eta \in \mathbb{Z}_+^r$	integer multiple of the lot-size S
$\delta \in \{0,1\}^r$	additional binary variables such that $\delta_j = 1$ ($j \in \{1,\dots,r\}$) iff $x_j > 0$ ($j \in \{1,\dots,r\}$)
$\gamma \in \mathbb{Z}_+^r$	additional vector of general integer variables
$\zeta \in \{0,1\}^n$	additional binary variables such that $\zeta_k = 1$ ($k \in \{1,\dots,n\}$) iff $\sum_{j\in C_k} x_j \leqslant \overline{s}$
$y \in \{0,1\}^\ell$	additional binary variables such that $y_l = 1$ ($l \in \{1,\dots,\ell\}$) iff $\sum_{j\in L_l} \delta_j = 1$
$z \in \{-1,0,1\}^r$	additional ternary variables such that $z_j = 1$ ($j \in \{1,\dots,r\}$) iff $x_j > 0$ ($j \in \{1,\dots,r\}$) and $z_j = -1$ ($j \in \{1,\dots,r\}$) iff $x_j < 0$ ($j \in \{1,\dots,r\}$)
$s \in \mathbb{R}^r$	additional continuous variables such that $s_j = \sqrt{x_j}$ ($j \in \{1,\dots,r\}$)
$b \in \mathbb{R}^T$	additional continuous variables such that $b_t = \sum_{j=1}^r (\nu_{jt} - \overline{\mu}_j)x_j$ ($t \in \{1,\dots,T\}$)

Functions:

$F_{(x)} : \mathbb{R} \to \mathbb{R}$	cumulative distribution of the normalized portfolio return
$\phi : \mathbb{R} \to \mathbb{R}$	penalty function for the minimum buy-in threshold constraint (3.9)
$\theta : \mathbb{R} \to \mathbb{R}$	penalty function for the round lot purchasing constraints (3.11)

BIBLIOGRAPHY

[1] F. Alizadeh and D. Goldfarb. "Second-Order Cone Programming". In: *Mathematical Programming, Series B* 95.1 (2003), pp. 3–51.

[2] E. Anderson et al. *LAPACK User's Guide*. Society for Industrial and Applied Mathematics (SIAM), 1999.

[3] S. Arora, E. Hazan, and S. Kale. "The Multiplicative Weights Update Method: A Meta-Algorithm and Applications". In: *Thory of Computing* 8 (2012), pp. 121–164.

[4] K.J. Arrow. *Essays in the Theory of Risk-Bearing*. North–Holland, 1970.

[5] C. Audet et al. "Links betwwn Linear Bilevel and Mixed 0-1 Programming". In: *Journal of Optimization Theory and Applications* 93.2 (1997), pp. 273–300.

[6] J. Avèrous and M. Meste. "Skewness for Multivariate Distributions: Two Approaches". In: *The Annals of Statistics* 25.5 (1997), pp. 1984–1997.

[7] L. Barone. "Bruno de Finetti. The Problem of Full-Risk Insurances". In: *Journal of Investment Management* 4.3 (2006), pp. 19–43.

[8] M.C. Bartholomew-Biggs and S.J. Kane. "A Global Optimization Problem in Portfolio Selection". In: *Computational Management Science* 6.3 (2009), pp. 329–345.

[9] J.E. Beasley. "Obtaining Test Problems via Internet". In: *Journal of Global Optimization* 8.4 (1996), pp. 429–433.

[10] J.E. Beasley. "OR-Library: Distributing Test Problems by Electronic Mail". In: *Journal of the Operational Research Society* 41.11 (1990), pp. 1069–1072.

[11] P. Belotti et al. "Mixed Integer Nonlinear Optimization". In: *Acta Numerica*. Vol. 22. Cambridge University Press, 2013, pp. 1–131.

[12] A. Ben-Tal and A.S. Nemirovskii. *Lectures on Modern Convex Optimization: Analysis, Algorithms, and Engineering Applications*. Society for Industrial and Applied Mathematics (SIAM), 2001.

[13] A. Ben-Tal and A.S. Nemirovskii. "Robust Convex Optimization". In: *Mathematics of Operations Research* 23.4 (1998), pp. 769–805.

[14] A. Ben-Tal and A.S. Nemirovskii. "Robust Solutions of Uncertain Linear Programs". In: *Operations Research Letters* 25.1 (1999), pp. 1–13.

[15] D. Bernoulli. "Specimen Theoriae Novae de Mensura Sortis". In: *Commentarii Academiae Scientiarum Imperialis Petropolitanae* 5 (1738).

[16] T. Berthold and A.M. Gleixner. "Undercover: A Primal MINLP Heuristic Exploring a Largest sub-MIP". In: *Mathematical Programming* 144.1 (2014), pp. 315–346.

[17] D. Bertsimas, C. Darnell, and R. Soucy. "Portfolio Construction Through Mixed-Integer Programming at Grantham, Mayo, Van Otterloo and Company". In: *Interfaces* 29.1 (1999), pp. 49–66.

[18] D. Bertsimas and R. Shioda. "Algorithm for Cardinality-Constrained Quadratic Optimization". In: *Computational Optimization and Applications* 43.1 (2006), pp. 1–22.

[19] M.J. Best and R.R. Grauer. "On the Sensitivity of Mean-Variance-Efficient Portfolios to Changes in Asset Means: Some Analytical and Computational Results". In: *The Review of Financial Studies* 4.2 (1991), pp. 331–342.

[20] D. Bienstock. "Computational Study of a Family of Mixed-Integer Quadratic Programming Problems". In: *Mathematical Programming* 74.2 (1996), pp. 121–140.

[21] L.S. Blackford et al. *ScaLAPACK User's Guide*. Society for Industrial and Applied Mathematics (SIAM), 1997.

[22] C.G.E. Boender and H.E. Romeijn. "Stochastic Methods". In: *Handbook of Global Optimization*. Ed. by R. Horst and H. Tuy. Springer–Verlag, 1990, pp. 829–869.

[23] I.M. Bomze. "On Standard Quadratic Optimization Problems". In: *Journal of Global Optimization* 13.4 (1998), pp. 369–387.

[24] I.M. Bomze, M. Locatelli, and F. Tardella. "New and Old Bounds for Standard Quadratic Optimization: Dominance, Equivalence and Incomparability". In: *Mathematical Programming, Series A* 115.1 (2008), pp. 31–64.

[25] P. Bonami, M. Kilinç, and J. Linderoth. "Algorithms and Software for Convex Mixed Integer Nonlinear Programs". In: *Mixed Integer Nonlinear Programming*. Ed. by J. Lee and S. Leyffer. Vol. 154. The IMA Volumes in Mathematics and its Applications. Springer–Verlag, 2012, pp. 1–39.

[26] P. Bonami and M.A. Lejeune. "An Exact Solution Approach for Portfolio Optimization Problems under Stochastic and Integer Constraints". In: *Operations Research* 57.3 (2009), pp. 650–670.

[27] P. Bonami et al. "An Algorithmic Framework for Convex Mixed Integer Nonlinear Programs". In: *Discrete Optimization* 5.2 (2008), pp. 186–204.

[28] P. Bonami et al. "More Branch-and-Bound experiments in convex nonlinear integer programming". Preprint ANL/MCS-P1949-0911, Argonne National Laboratory, Mathematics and Computer Science Division. 2011.

[29] Bonmin. URL: https://projects.coin-or.org/Bonmin.

[30] J.-F. Bonnans et al. *Numerical Optimization: Theoretical and Practical Aspects*. Second. Universitext. Springer–Verlag, 2000.

[31] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

[32] S. Boyd et al. *Linear Matrix Inequalities in System and Control Theory*. Vol. 15. Studies in Applied Mathematics. Society for Industrial and Applied Mathematics (SIAM), 1994.

[33] K.M. Bretthauer and B. Shetty. "The Nonlinear Knapsack Problem: Algorithms and Applications". In: *European Journal of Operational Research* 138.3 (2002), pp. 459–472.

[34] R. Bringhurst. *The Elements of Typographic Style: 4.0: 20th Anniversary Edition*. Hartley & Marks, 2013.

[35] M. Britten-Jones. "The Sampling Error in Estimates of Mean-Variance Efficient Portfolio Weights". In: *The Journal of Finance* 54.2 (1999), pp. 655–671.

[36] M. Broadie. "Computing Efficient Frontiers using Estimated Parameters". In: *Annals of Operations Research* 45.1 (1993), pp. 21–58.

[37] C. Buchheim et al. "A Frank-Wolfe Based Branch–and–Bound Algorithm for Mixed–Integer Portfolio Optimization". http://arxiv.org/abs/1507.05914. 2015.

[38] O.P. Burdakov, C. Kanzow, and A. Schwartz. "Mathematical Programs with Cardinality Constraints: Reformulation by Complementarity-type Conditions and a Regularization Method". In: *SIAM Journal of Optimization* 26.1 (2016), pp. 397–425.

[39] A. Caprara, H. Kellerer, and U. Pferschy. "The Multiple Subset Sum Problem". In: *SIAM Journal on Optimization* 11.2 (2000), pp. 308–319.

[40] M.F. Cardoso et al. "A Simulated Annealing Approach to the Solution of MINLP Problems". In: *Computers and Chemical Engineering* 21.12 (1997), pp. 1349–1364.

[41] G. Castellani, M. De Felice, and F. Moriconi. *Manuale di Finanza. Teoria del Portafoglio e Mercato Azionario (in Italian)*. Vol. 2. Il Mulino, 2005.

[42] S. Ceria and R.A. Stubbs. "Incorporating Estimation Errors into Portfolio Selection: Portfolio Construction". In: *Journal of Asset Management* 7.2 (2006), pp. 109–127.

[43] V. Cerny. "Thermodynamical Approach to the Travelling Salesman Problem: An Efficient Simulation Algorithm". In: *Journal of Optimization Theory and Applications* 45.1 (1985), pp. 41–51.

[44] F. Cesarone, A. Scozzari, and F. Tardella. "A New Method for Mean-Variance Portfolio Optimization with Cardinality Constraints". In: *Annals of Operations Research* 205.1 (2013), pp. 213–234.

[45] F. Cesarone, A. Scozzari, and F. Tardella. "Efficient Algorithms for Mean–Variance Portfolio Optimization with Hard Real–Word Constraints". In: *Proceedings of the 18th AFIR Colloquium: Financial Risk in a Changing World*. 2008.

[46] F. Cesarone, A. Scozzari, and F. Tardella. "Efficient Algorithms for Mean-Variance Portfolio Optimization with Hard Real-Word Constraints". In: *Giornale dell'Istituto Italiano degli Attuari* 72 (2009), pp. 37–56.

[47] F. Cesarone, A. Scozzari, and F. Tardella. "Portfolio Selection Problems in Practice: A Comparison Between Linear and Quadratic Optimization Models". 2010. URL: http://arxiv.org/abs/1105.3594.

[48] T.-J. Chang, S.C. Yang, and K.J. Chang. "Portfolio Optimization Problems in Different Risk Measures Using Genetic Algorithm". In: *Expert Systems with Applications* 36.7 (2009), pp. 10529–10537.

[49] T.-J. Chang et al. "Heuristics for cardinality constrained portfolio optimization". In: *Computers and Operations Research* 27.13 (2000), pp. 1271–1302.

[50] Z.-P. Chen and C. Zhao. "Sensitivity to Estimation Errors in Mean-Variance Models". In: *Acta Mathematicae Applicatae Sinica* 19.2 (2003), pp. 255–266.

[51] C. Chenkuri and S. Khanna. "A PTAS for the Multiple Knapsack Problem". In: *SIAM Journal on Computing* 35.3 (2006), pp. 713–728.

[52] V.K. Chopra. "Mean-Variance Revisited: Near-Optimal Portfolios and Sensitivity to Input Variations". In: *Journal of Investing* 2.1 (1993), pp. 51–59.

[53] V.K. Chopra and W.T. Ziemba. "The Effect of Errors in Means, Variances, and Covariances on Optimal Portfolio Choice". In: *Journal of Portfolio Management* 12.2 (1993), pp. 6–11.

[54] J.S. Cohen. *Computer Algebra and Symbolic Computation: Mathematical Methods*. A K Peters, Natick, Massachusetts, 2003.

[55] G.M. Constantinides and A.G. Malliaris. "Portfolio Theory". In: *Finance. Handbooks in Operations Research and Management Science*. Ed. by R.A. Jarrow, V. Maksimovic, and W.T. Ziemba. Vol. 9. North–Holland, 1995, pp. 1–30.

[56] T.E. Copeland and J.F. Weston. *Financial Theory and Corporate Policy*. Addison–Wesley, 1988.

[57] O.L.V. Costa and A.C. Paiva. "Robust Portfolio Selection Using Linear-Matrix Inequalities". In: *Journal of Economics Dynamics and Control* 26.6 (2002), pp. 889–909.

[58] R.W. Cottle, J.-S. Pang, and R.E. Stone. *The Linear Complementarity Problem*. Classics in Applied Mathematics. Society for Industrial and Applied Mathematics (SIAM), 2009.

[59] Couenne. URL: https://projects.coin-or.org/Couenne.

[60] G. Cournéjols and R.H. Tütüncü. *Optimization Methods in Finance*. Mathematics, Finance and Risk. Cambridge University Press, 2007.

[61] Y. Crama and M. Schyns. "Simulated Annealing for Complex Portfolio Selection Problems". In: *European Journal of Operational Research* 150.3 (2003), pp. 546–571.

[62] C. D'Ambrosio and A. Lodi. "Mixed Integer Nonlinear Programming Tools: An Updated Practical Overview". In: *Annals of Operations Research* 204.1 (2013), pp. 301–320.

[63] C. D'Ambrosio and S. Martello. "Heuristic Algorithms for the General Nonlinear Separable Knapsack Problem". In: *Computers & Operations Research* 38.2 (2011), pp. 505–513.

[64] P. de Fermat. "Observatio Domini Petri de Fermat". Diophantus of Alexandria, Arithmetica, Arithmeticorum Liber II. 1670.

[65] B. de Finetti. "Il Problema dei Pieni (in Italian)". In: *Giornale dell'Istituto Italiano degli Attuari* 11 (1940), pp. 1–88.

[66] B. de Finetti. "Sulla Preferibilità (in Italian)". In: *Giornale degli Economisti e Annali di Economia* 11.11–12 (1952), pp. 685–709.

[67] J.A. De Loera et al. "Integer Polynomial Optimization in Fixed Dimension". In: *Mathematics of Operations Research* 31.1 (2006), pp. 147–153.

[68] A. Dekker and E. Aarts. "Global Optimization and Simulated Annealing". In: *Mathematical Programming* 50.1–3 (1991), pp. 367–393.

[69] A. Del Pia, S.S. Dey, and M. Molinaro. "Mixed-Integer Quadratic Programming is in **NP**". In: *Mathematical Programming* 162.1 (2017), pp. 225–240.

[70] G.F. Deng and W.T. Lin. "Ant Colony Optimization for Markowitz Mean-Variance Portfolio Model". In: *First International Conference on Swarm, Evolutionary, and Memetic Computing, SEMCCO 2010*. Ed. by B.K. Panigrahi et al. Swarm, Evolutionary, and Memetic Computing. Springer–Verlag, 2010, pp. 238–245.

[71] L. Di Gaspero et al. "Hybrid Local Search for Constrained Financial Portfolio Selection Problem". In: *Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems. 4th International Conference, CPAIOR 2007, Brussels*. Ed. by P. Van Hentenryck and L. Wolsey. Vol. 4510. Lecture Notes in Computer Science. Springer–Verlag, 2007, pp. 44–58.

[72] L. Di Gaspero et al. "Hybrid Metaheuristics for Constrained Portfolio Selection Problems". In: *Quantitative Finance* 11.10 (2011), pp. 1473–1487.

[73] L. Di Gaspero et al. "Local Search for Constrained Financial Portfolio Selection Problems with Short Sellings". In: *Learning and Intelligent Optimization. 5th International Conference (LION 5), Rome, January 17–21, 2011*. Ed. by C.A. Coello Coello. Vol. 6683. Lecture Notes in Computer Science. Springer–Verlag, 2011, pp. 450–453.

[74] D. Di Lorenzo et al. "A Concave Optimization-based Approach to Sparse Portfolio Selection". In: *Optimization Methods and Software* 27.6 (2012), pp. 983–1000.

[75] J.J. Dongarra et al. *LINPACK User's Guide*. Society for Industrial and Applied Mathematics (SIAM), 1979.

[76] M. Ehrgott. *Multicriteria Optimization*. Springer, 2005.

[77] L. El Ghaoui and H. Lebret. "Robust Solutions to Least-Squares Problems with Uncertain Data". In: *SIAM Journal of Matrix Analysis and Applications* 18.4 (1997), pp. 1035–1064.

[78] L. El Ghaoui, M. Oks, and F. Oustry. "Worst-Case Value-at-Risk and Robust Portfolio Optimization: A Conic Programming Approach". In: *Operations Research* 51.4 (2003), pp. 543–556.

[79] L. El Ghaoui, F. Oustry, and H. Lebret. "Robust Solutions to Uncertain Semidefinite Programs". In: *SIAM Journal of Optimization* 9.1 (1998), pp. 33–52.

[80] E.J. Elton and M.J. Gruber. "Modern Porfolio Thory, 1950 to Date". In: *Journal of Banking and Finance* 21.11–12 (1997), pp. 1743–1759.

[81] F.J. Fabozzi, D. Huang, and G. Zhou. "Robust Portfolios: Contributions from Operations Research and Finance". In: *Annals of Operations Research* 176.1 (2010), pp. 191–220.

[82] F.J. Fabozzi et al. *Robust Portfolio Optimization and Management*. John Wiley and Sons, 2007.

[83] E.F. Fama. *Foundations of Finance: Portfolio Decisions and Securities Prices*. Basic Books, 1976.

[84] E.F. Fama. "Mandelbrot and the Stable Paretian Hypothesis". In: *The Journal of Business* 36.4 (1963), pp. 420–429.

[85] E.F. Fama. "The Behaviour of Stock-Market Prices". In: *The Journal of Business* 38.1 (1965), pp. 34–105.

[86] B. Fastrich and P. Winker. "Robust Portfolio Optimization with a Hybrid Heuristic Algorithm". In: *Computational Management Science* 9.1 (2012), pp. 63–88.

[87] A. Fernández and S. Gómez. "Portfolio Selection Using Neural Networks". In: *Computers and Operations Research* 34.4 (2007), pp. 1177–1191.

[88] A.V. Fiacco and Y. Ishizuka. "Sensitivity and Stability Analysis for Nonlinear Programming". In: *Annals of Operations Research* 27.1 (1990), pp. 215–235.

[89] T.P. Filomena and M.A. Lejeune. "Stochastic Portfolio Optimization with Proportional Transaction Costs: Convex Reformulations and Computational Experiments". In: *Operations Research Letters* 40.3 (2012), pp. 212–217.

[90] T.P. Filomena and M.A. Lejeune. "Warm–Start Heuristic for Stochastic Portfolio Optimization with Fixed and Proportional Transaction Costs". In: *Journal of Optimization Theory and Applications* (2013).

[91] D.E. Finkel. *DIRECT Optimization Algorithm User Guide*. Center for Resarch in Scientific Computation, Department of Mathematics, North Carolina State University. 2003.

[92] M. Fischetti and A. Lodi. "Local Branching". In: *Mathematical Programming, Series B* 98.1–3 (2003), pp. 23–47.

[93] P. Fishburn. *Utility Theory for Decision Making*. John Wiley and Sons, 1970.

[94] R. Fletcher. *Practical Methods of Optimization*. Second. John Wiley and Sons, 1987.

[95] C.A. Floudas and V. Visweswaran. "Quadratic Optimization". In: *Handbook of Global Optimization*. Ed. by R. Horst and P.M. Pardalos. Vol. 2. Nonconvex Optimization and Its Applications. Springer, 1995, pp. 217–269.

[96] A. Frangioni, F. Furini, and C. Gentile. "Approximated Perspective Relaxations: A Project and Lift Approach". In: *Computational Optimization and Applications* 3.63 (2016), pp. 705–735.

[97] A. Frangioni and C. Gentile. "Perspective Cuts for a Class of Convex 0-1 Mixed Integer Programs". In: *Mathematical Programming, Series A* 106.2 (2006), pp. 225–236.

[98] R.M. Freund and S. Mizuno. "Interior Point Methods: Current Status and Future Directions". In: *High Performance Optimization*. Ed. by H. Frenk et al. Vol. 33. Applied Optimization. Kluwer Academic Publishers, 2000, pp. 441–466.

[99] J.M. Gablonsky. *DIRECT Version 2.0*. Center for Resarch in Scientific Computation, Department of Mathematics, North Carolina State University. 2001.

[100] J.M. Gablonsky. "Modifications of the DIRECT Algorithm". PhD thesis. Department of Mathematics, North Carolina State University, 2001.

[101] V. Gabriel, C. Murat, and A. Thiele. "Recent Advances in Robust Optimization: An Overview". In: *European Journal of Operational Research* 235.3 (2014), pp. 471–483.

[102] M.R. Garey and D.S. Johnson. *Computers and Intractability: A Guide to the Theory of* **NP**-*Completeness*. Series of Books in the Mathematical Sciences. W.H. Freeman and Company, 1979.

[103] F. Glover. "Tabu Search, Part 1". In: *ORSA Journal of Computing* 1.3 (1989), pp. 190–205.

[104] F. Glover. "Tabu Search, Part 2". In: *ORSA Journal of Computing* 2.1 (1990), pp. 4–32.

[105] F. Glover and M. Laguna. *Tabu Search*. Kluwer Academic Publishers, 1997.

[106] D. Goldfarb and G. Iyengar. "Robust Portfolio Selection Problems". In: *Mathematics of Operations Research* 28.1 (2003), pp. 1–38.

[107] G.H. Golub and C.F. Van Loan. *Matrix Computation*. Third. Johns Hopkins Studies in the Mathematical Science. John Hopkins University Press, 1996.

[108] J. Gondzio. "Interior Point Methods 25 Years Later". In: *European Journal of Operational Research* 218.3 (2012), pp. 587–601.

[109] R.H. Green and B. Hollifield. "When Will Mean-Variance Efficient Portfolios be well Diversified?" In: *The Journal of Finance* 47.5 (1992), pp. 1785–1809.

[110] O. Günlük and J. Linderoth. "Perspective Reformulation and Applications". In: *Mixed Integer Nonlinear Programming*. Ed. by J. Lee and S. Leyffer. Vol. 154. The IMA Volumes in Mathematics and its Applications. Springer–Verlag, 2012, pp. 61–81.

[111] O.K. Gupta and A. Ravindran. "Branch-and-Bound Experiments in Convex Nonlinear Integer Programming". In: *Management Science* 31.12 (1985), pp. 1533–1546.

[112] G. Hanoch and H. Levy. "The Efficiency Analysis of Choices Involving Risk". In: *Review of Economic Studies* 36.3 (1969), pp. 335–346.

[113] G.H. Hardy, J.E. Littlewood, and G. Pòlya. *Inequalities*. Cambridge University Press, 1934.

[114] T. Heath. *Diophantus of Alexandria*. Dover, New York, 1964.

[115] R. Hemmecke et al. "Nonlinear Integer Programming". In: *50 Years of Integer Programming 1958-2008: From the Early Years to the State-of-the-Art*. Ed. by M. Jünger et al. Springer–Verlag, 2010, pp. 561–618.

[116] N.J. Higham. "Analysis of the Cholesky Decomposition of a Semi–Definite Matrix". In: *Reliable Numerical Computation*. Ed. by M.G. Cox and S. Hammarling. Oxford University Press, 1990, pp. 161–185.

[117] N.J. Higham. "Computing the Nearest Correlation Matrix: A Problem from Finance". In: *IMA Journal of Numerical Analysis* 22 (2002), pp. 329–343.

[118] J.B. Hiriart-Urruty and C. Lemaréchal. *Convex Analysis and Minimization Algorithms*. Vol. 1. Springer–Verlag, 1999.

[119] J.B. Hiriart-Urruty and C. Lemaréchal. *Convex Analysis and Minimization Algorithms*. Vol. 2. Springer–Verlag, 1999.

[120] L.W. Hoe, J.S. Hafizah, and I. Zaidi. "An Empirical Comparison of Different Risk Measures in Portfolio Optimization". In: *Business and Economic Horizons* 1.1 (2010), pp. 39–45.

[121] C.F. Huang and R.H. Litzenberger. *Foundations for Financial Economics*. North–Holland, 1988.

[122] J. Humpola, A. Fügenschuh, and T. Lehman. "A Primal Heuristic for Optimizing the Topology of Gas Networks Based on Dual Information". In: *EURO Journal on Computational Optimization* (2014). DOI: 10.1007/s13675-014-0029-0.

[123] T. Ibaraki and N. Katoh. *Resource Allocation Problems*. Cambridge, MA: MIT Press, 1998.

[124] IBM. *ILOG CPLEX 12.2 User's Manual*. IBM. 2010.

[125] R.A. Ion. "Nonparametric Statistical Process Control". PhD thesis. Korteweg-de Vries Instituut voor Wiskunde, Faculteit der Natuurwetenschappen, Wiskunde en Informatica, Universiteit van Amsterdam, 2001.

[126] Ipopt. URL: https://projects.coin-or.org/Ipopt.

[127] B.I. Jacobs, K.N. Levy, and H.M. Markowitz. "Portfolio Optimization with Factors, Scenarios, and Realistic Short Positions". In: *Operations Research* 53.4 (2005), pp. 586–599.

[128] K. Jansen. "A Fast Approximation Scheme for the Multiple Knapsack Problem". In: *SOFSEM 2012 (38th Conference on Current Treads in Theory and Practise of Computer Science, Špindlerův Mlýn, Czech Republic, January 2012)*. Ed. by M. Bieliková et al. Vol. 7147. Lecture Notes in Computer Science. 2012, pp. 313–324. DOI: 10.1007/978-3-642-27660-6_26.

[129] K. Jansen. "Parametrized Approximation Scheme for the Multiple Knapsack Problem". In: *SIAM Journal on Computing* 39.4 (2009), pp. 1392–1412.

[130] R.G. Jeroslow. "There Cannot be any Algorithm for Integer Programming with Quadratic Constraints". In: *Operations Research* 21.1 (1973), pp. 221–224.

[131] J.D. Jobson. "Confidence Regions for the Mean-Variance Efficient Set: An Alternative Approach to Estimation Risk". In: *Review of Quantitative Finance and Accounting* 1.3 (1991), pp. 235–257.

[132] J.D. Jobson and R.M. Korkie. "Putting Markowitz Theory to Work". In: *Journal of Portfolio Management* 7.4 (1981), pp. 70–74.

[133] N.J. Jobst et al. "Computational Aspects of Alternative Portfolio Selection Models in the Presence of Discrete Asset Choice Constraints". In: *Quantitative Finance* 1 (2001), pp. 1–13.

[134] D.R. Jones. "The DIRECT Global Optimization Algorithm". In: *Encyclopaedia of Optimization*. Ed. by C.A. Floudas and P.M. Pardalos. Kluwer Academic Publishers, 2001, pp. 421–440.

[135] D.R. Jones, C.D. Pettunen, and B.E. Stuckman. "Lipschitzian Optimization without the Lipschitz Constant". In: *Journal of Optimization Theory and Applications* 79.1 (1993), pp. 157–181.

[136] P. Jorion. "Portfolio Optimization in Practice". In: *Financial Analysis Journal* 48.1 (1992), pp. 68–74.

[137] J.G. Kallberg and W.T. Ziemba. "Comparision of Alternative Utility Functions in Portfolio Selection Problems". In: *Management Science* 29.11 (1983), pp. 1257–1276.

[138] J.G. Kallberg and W.T. Ziemba. "Mis-Specifications in Portfolio Selection Problems". In: *Proceedings of the 2nd Summer Workshop on Risk and Capital Held*. Ed. by G. Bamberg and K. Spremann. Vol. 227. Lecture Notes in Economics and Mathematical Systems. Springer-Verlag, 1984, pp. 74–87.

[139] R. Kannan and C.L. Monma. "On the Computational Complexity of Integer Programming Problems". In: *Optimization and Operations Research*. Ed. by R. Henn, B. Korte, and W. Oettli. Vol. 157. Lecture Notes in Economics and Mathematical Systems. Springer–Verlag, 1978, pp. 161–172.

[140] C. Kanzow and A. Schwartz. "A New Regularization Method for Mathematical Programs with Complementarity Constraints with Strong Convergence Properties". In: *SIAM Journal of Optimization* 23.2 (2013), pp. 770–798.

[141] S. Kataoka. "A Stochastic Programming Model". In: *Econometrica* 31.1–2 (1963), pp. 181–196.

[142] H. Kellerer, U. Pferschy, and D. Pisinger. *Knapsack Problems*. Berlin, Germany: Springer, 2004.

[143] A. Kielbasinski. "A Note on Rounding-Error Analysis of Cholesky Factorization". In: *Linear Algebra and its Applications* 88–89 (1987), pp. 487–494.

[144] S. Kirkpatrick. "Optimization by Simulated Annealing: Quantitative Studies". In: *Journal of Statistical Physics* 34.5–6 (1984), pp. 975–986.

[145] S. Kirkpatrick, C.D. Gelatt, and M.P. Vecchi. "Optimization by Simulated Annealing". In: *Science* 220.4598 (1983), pp. 671–680.

[146] P.N. Kolm, R.H. Tütüncü, and F.J. Fabozzi. "60 Years of Portfolio Optimization: Practical Challenges and Current Trends". In: *European Journal of Operational Research* 234.2 (2014), pp. 356–371.

[147] H. Konno and K. Suzuki. "A Fast Algorithm for Solving Large Scale Mean-Variance Models by Compact Factorization of Covariance Matrices". In: *Journal of the Operations Research Society of Japan* 35.1 (1992), pp. 93–104.

[148] H. Konno and A. Wijayanayake. "Portfolio Optimization Problem under Concave Transaction Costs and Minimal Transaction Unit Constraints". In: *Mathematical Programming, Series B* 89.2 (2001), pp. 233–250.

[149] M. Köppe. "On the Complexity of Nonlinear Mixed-Integer Optimization". In: *Mixed Integer Nonlinear Programming*. Ed. by J. Lee and S. Leyffer. Vol. 154. The IMA Volumes in Mathematics and its Applications. Springer–Verlag, 2012, pp. 533–557.

[150] A. Kulik and H. Shachnai. "There is No EPTAS for Two-dimensional Knapsack". In: *Information Processing Letters* 110.16 (2010), pp. 707–710.

[151] E.K. Lee and J.E. Mitchell. "Computational Experience of An Interior-Point SQP Algorithm in a Parallel Branch-and-Bound Framework". In: *High Performance Optimization*. Ed. by H. Frenk et al. Vol. 33. Applied Optimization. Springer–Verlag, 2000, pp. 329–347.

[152] M.A. Lejeune. "A VaR Black-Litterman Model for the Construction of Absolute Return Fund-of-Funds". In: *Quantitative Finance* 11.10 (2011), pp. 1489–1501.

[153] M.A. Lejeune. "Portfolio Optimization with Combinatorial and Downside Return Constraints". In: *Selected Contributions from the MOPTA 2012 Conference*. Ed. by L.F. Zuluaga and T. Terlaky. Modeling and Optimization: Theory and Applications. Springer–Verlag, 2014, pp. 31–50.

[154] S. Leyffer. "Deterministic Methods for Mixed Integer Nonlinear Programming". PhD thesis. University of Dundee, 1993.

[155] S. Leyffer. "User Manual for MINLP_BB". Argonne National Laboratory, Mathematics and Computer Science Division. 2003.

[156] D. Li and X. Sun. *Nonlinear Integer Programming*. Vol. 84. International Series in Operations Research & Management Science. Berlin, Germany: Springer, 2006.

[157] L. Liberti. "Reformulations in Mathematical Programming: Definitions and Systematics". In: *RAIRO–RO* 43.1 (2009), pp. 55–86.

[158] L. Liberti. "Writing Global Optimization Software". In: *Global Optimization: From Theory to Implementation*. Ed. by L. Liberti and N. Maculan. Vol. 84. Nonconvex Optimization and Its Applications. Springer, 2006, pp. 211–262.

[159] L. Liberti, S. Cafieri, and F. Tarissan. "Reformulations in Mathematical Programming: A Computational Approach". In: *Foundations of Computational Intelligence Vol. 3*. Ed. by A. Abraham et al. Studies in Computational Intelligence 203. Springer, 2009, pp. 153–234.

[160] B. Lin et al. "Using Tabu Search to Solve MINLP Problems for PSE". In: *Computer Aided Chemical Engineering. 8th International Symposium on Process Systems Engineering*. Ed. by B. Chen and A.W. Westerberg. Vol. 15. 2003, pp. 541–546.

[161] Z. Lin and Z. Bai. *Probability Inequalities*. Springer–Verlag, 2011.

[162] M.S. Lobo, M. Fazel, and S. Boyd. "Portfolio Optimization with Linear and Fixed Transaction Costs". In: *Annals of Operations Research* 152.1 (2007), pp. 341–365.

[163] M.S. Lobo et al. "Applications of Second-Order Cone Programming". In: *Linear Algebra and its Applications* 284.1–3 (1998), pp. 193–228.

[164] M. Locatelli. "Simulated Annealing Algorithms for Global Optimization". In: *Handbook of Global Optimization*. Ed. by R. Horst and P.M. Pardalos. Vol. 2. Nonconvex Optimization and Its Applications. Kluwer Academic Publishers, 2002, pp. 217–269.

[165] M. Locatelli and F. Schoen. *Global Optimization: Theory, Algorithms, and Applications*. MPS-SIAM Series on Optimization. Society for Industrial and Applied Mathematics, 2013.

[166] H.T. Loh and P.Y. Papalambros. "A Sequential Linearization Approach for Solving Mixed-Discrete Nonlinear Design Optimization Problems". In: *Journal of Mechanical Design* 113.1 (1991), pp. 325–334.

[167] Z. Lu and Y. Zhang. "Sparse Approximation via Penalty Decomposition Methods". In: *SIAM Journal of Optimization* 23.4 (2013), pp. 2448–2478.

[168] S. Lucidi. "Appunti dalle Lezioni di Ottimizzazione Globale (in Italian)". Dipartimento di Informatica e Sistemistica "A. Ruberti", Università di Roma "La Sapienza". 2015–2016.

[169] N. Madras. *Lectures on Monte Carlo Methods*. The Fields Institute for Research in Mathematical Sciences. American Mathematical Society, 2002.

[170] B. Mandelbrot. "The Variation of Certain Speculative Prices". In: *The Journal of Business* 36.4 (1963), pp. 394–419.

[171] V. Maniezzo, T. Stützle, and S. Voß, eds. *Hybridizing Metaheuristics and Mathematical Programming*. Vol. 10. Annals of Information Systems. New York: Springer, 2009.

[172] R. Mansini, W. Ogryczak, and M.G. Speranza. *Linear and Mixed Integer Programming for Portfolio Optimization*. Euro Advanced Tutorials on Operational Research. Springer–Verlag, 2015.

[173] R. Mansini, W. Ogryczak, and M.G. Speranza. "Twenty Years of Linear Programming Based Portfolio Optimization". In: *European Journal of Operational Research* 234.2 (2014), pp. 518–535.

[174] R. Mansini and M.G. Speranza. "Heuristic Algorithms for the Portfolio Selection Problem with Minimum Transaction Lots". In: *European Journal of Operational Research* 114.2 (1999), pp. 219–233.

[175] H.M. Markowitz. "de Finetti Scoops Markowitz". In: *Journal of Investment Management* 4.3 (2006), pp. 5–18.

[176] H.M. Markowitz. "Mean-Variance Approximations to Expected Utility". In: *European Journal of Operational Research* 234.2 (2014), pp. 346–355.

[177] H.M. Markowitz. "Portfolio Selection". In: *The Journal of Finance* 7.1 (1952), pp. 77–91.

[178] H.M. Markowitz. *Portfolio Selection: Efficient Diversification of Investimen*. Cowles Foundation for Research in Economics at Yale University, 1959.

[179] H.M. Markowitz. "Portfolio Theory: As I Still See It". In: *Annual Review of Financial Economics* 2 (2010), pp. 1–23.

[180] H.M. Markowitz and G.P. Todd. *Mean–Variance Analysis in Portfolio Choice and Capital Markets*. Revised. John Wiley and Sons, 2000.

[181] A.W. Marshall, I. Olkin, and B.C. Arnorld. *Inequalities: Theory of Majorization and Its Applications*. Second. Springer Series in Statistics. Springer–Verlag, 2011.

[182] S. Martello and P. Toth. *Knapsack Problems: Algorithms and Computer Implementations*. Chichester, New York: John Wiley & Sons, 1990.

[183] R. Martí. "Multi-Start Methods". In: *Handbook of Methaheuristics*. Ed. by F. Glover and G. A. Kochenberger. Vol. 57. International Series in Operations Research & Management Science. Springer, 2003.

[184] J. Meinguet. "Refined Error Analyses of Cholesky Factorization". In: *SIAM Journal of Numerical Analysis* 20.6 (1983), pp. 1243–1250.

[185] L. Mencarelli and C. D'Ambrosio. "Complex Portfolio Selection via Convex Mixed-Integer Quadratic Programming: A Survey". Laboraire d'Informatique (LIX), École Polytechnique, Palaiseau, France. 2017.

[186] L. Mencarelli, C. D'Ambrosio, and S. Martello. "Relaxations and Heuristics for the General Multiple NonLinear Knapsack". DEI, University of Bologna, Italy and LIX, École Polytechnique, Palaiseau, France. 2017.

[187] L. Mencarelli, Y. Sahraoui, and L. Liberti. "A Multiplicative Weights Update Algorithm for MINLP". In: *EURO Journal on Computational Optimization* 5.1–2 (2017), pp. 31–86.

[188] L. Mencarelli et al. "Heuristics for the General Multiple Non-linear Knapsack Problem". In: *Electronics Notes in Discrete Mathematics* 55 (2016), pp. 69–72.

[189] N. Metropolis et al. "Equation of State Calculations by Fast Computing Machines". In: *The Journal of Chemical Physics* 21.6 (1953), pp. 1087–1092.

[190] R.O. Michaud. "The Markowitz Optimization Enigma: is "Optimized" Optimal?" In: *Financial Analysis Journal* 45.1 (1989), pp. 31–42.

[191] T.C. Mills. "Stylized Facts on the Temporal and Distributional Properties of Daily FT–SE Returns". In: *Applied Financial Economics* 7.6 (1997), pp. 599–604.

[192] J.E. Mitchell and S. Braun. "Rebalancing an Investment Portfolio in the Presence of Convex Transaction Costs and Market Impact Costs". In: *Optimization Methods and Software* 28.3 (2013), pp. 523–542.

[193] C.B. Moler and G.W. Stewart. "On the Householder-Fox Algorithm for Decomposing a Projection". In: *Journal of Computational Physics* 28.1 (1978), pp. 82–91.

[194] A. Montesano. "de Finetti and the Arrow-Pratt Measure of Risk Aversion". In: *Bruno de Finetti Radical Probabilist*. Ed. by M.C. Galavotti. Texts in Philosophy (Book 8). College Pubblication, 2009, pp. 115–127.

[195] R. Moral-Escudero, R. Ruiz-Torrubiano, and A. Suarez. "Selection of Optimal Investment Portfolios with Cardinality Constraints". In: *IEEE Congress on Evolutionary Computation, CEC'06*. 2006, pp. 2382–2388.

[196] A. Munawar et al. "arGA: Adaptive Resolution Micro-Genetic Algorithm with Tabu Search to Solve MINLP Problems Using GPU". In: *Massively Parallel Evolutionary Computation on GPGPUs*. Ed. by S. Tsutsui and P. Collet. Natural Computing Series. Springer–Verlag, 2013.

[197] A. Munawar et al. "Solving Extremely Difficult MINLP Problems Using Adaptive Resolution Micro-GA with Tabu Search". In: *Learning and Intelligent Optimization. 5th International Conference (LION 5), Rome, January 17–21, 2011*. Ed. by C.A. Coello Coello. Vol. 6683. Lecture Notes in Computer Science. Springer–Verlag, 2011, pp. 203–217.

[198] K.G. Murty and S.N. Kabadi. "Some **NP**-complete Problems in Quadratic and Nonlinear Programming". In: *Mathematical Programming* 39.2 (1987), pp. 117–129.

[199] A.S. Nemirovskii and M.J. Todd. "Interior–Point Methods for Optimization". In: *Acta Numerica*. Vol. 17. Cambridge University Press, 2008, pp. 191–234.

[200] Y. Nesterov and A.S. Nemirovskii. *Interior–Point Polynomial Algorithms in Convex Programming*. Society for Industrial and Applied Mathematics (SIAM), 1994.

[201] J. Nocedal and S.J. Wright. *Numerical Optimization*. Springer Series in Operations Research. Springer–Verlag, 1999.

[202] C. Papadimitriou. *Computational Complexity*. Addison–Wesley, 1993.

[203] P.M. Pardalos and G.P. Rodgers. "Computational Aspects of a Branch and Bound Algorithm for Quadratic Zero-One Programming". In: *Computing* 45.2 (1990), pp. 131–144.

[204] P.M. Pardalos and G. Schnitger. "Checking Local Optimality in Constrained Quadratic Programming is **NP**-hard". In: *Operations Research Letters* 7.1 (1988), pp. 33–35.

[205] A.F. Perold. "Large-Scale Portfolio Optimization". In: *Management Science* 30.10 (1984), pp. 1143–1160.

[206] S.A. Plotkin, D.B. Shmoys, and E. Tardos. "Fast Approximation Algorithm for Fractional Packing and Covering Problems". In: *Mathematics of Operations Research* 20.2 (1995), pp. 257–301.

[207] F.A. Potra and S.J. Wright. "Interior-Point Methods". In: *Journal of Computational and Applied Mathematics* 124 (2000), pp. 281–302.

[208] J.W. Pratt. "Risk Adversion in the Small and in the Large". In: *Econometrica* 32.1–2 (1964), pp. 122–136.

[209] F. Pressacco and P. Serafini. "The Origins of the Mean-Variance Approach in Fnance: Revisiting de Finetti 65 Years Later". In: *Decisions in Economics and Finance* 30.1 (2007), pp. 19–49.

[210] S.T. Rachev et al. "An Empirical Examination of Daily Stock Return Distributions for U.S. Stocks". In: *Data Analysis and Decision Support*. Studies in Classification, Data Analysis, and Knowledge Organization. Springer–Verlag, 2005, pp. 269–281.

[211] R.T. Rockafellar. *Theory of Subgradients and Its Applications to Problems of Optimization: Convex and Nonconvex Functions*. Vol. 1. Research and Exposition in Mathematics. Heldermann Verlag, 1981.

[212] M. Rubinstein. *A History of the Theory of Investments: My Annotated Bibliography*. Wiley Finance 335. John Wiley and Sons, 2006.

[213] M. Rubinstein. "Bruno de Finetti and Mean-Variance Portfolio Selection". In: *Journal of Investment Management* 4.3 (2006), pp. 3–4.

[214] M. Rubinstein. "Markowitz's Portfolio Selection: A Fifty-Year Retrospective". In: *The Journal of Finance* 57.3 (2002), pp. 1041–1045.

[215] R.Y. Rubinstein and D.P. Kroese. *Simulation and the Monte Carlo Method*. John Wiley and Sons, 2008.

[216] R. Ruiz-Torrubiano and A. Suarez. "Hybrid Approaches and Dimensionality Reduction for Portfolio Selection with Cardinality Constraints". In: *IEEE Computational Intelligence Magazine* 5.2 (2010), pp. 92–107.

[217] R. Saigal, L. Vandenberghe, and H. Wolkowicz. *Handbook of Semidefinite Programming and Applications*. Kluwer Academic Publishers, 2000.

[218] A. Schaerf. "Local Search Techniques for Constrained Portfolio Selection Problems". In: *Computational Economics* 20.3 (2002), pp. 177–190.

[219] B. Scherer and D. Martin. *Introduction to Modern Portfolio Optimization*. Springer–Verlag, 2005.

[220] F. Schoen. "Stocastic Global Optimization: Stopping Rules". In: *Encyclopedia of Optimizaton*. Ed. by C.A. Floudas and P.M. Pardalos. Kluwer Academic Publishers, 2001, pp. 297–301.

[221] F. Schoen. "Stocastic Global Optimization: Two-Phase Methods". In: *Encyclopedia of Optimizaton*. Ed. by C.A. Floudas and P.M. Pardalos. Kluwer Academic Publishers, 2001, pp. 297–301.

[222] F. Schoen. "Two-Phase Methods for Global Optimization". In: *Handbook of Global Optimization*. Ed. by P.M. Pardalos and H.E. Romeijn. Vol. 2. Nonconvex Optimization and Its Applications. Springer, 2002, pp. 151–177.

[223] Scip. URL: http://scip.zib.de.

[224] A. Scozzari and F. Tardella. "A Clique Algorithm for Standard Quadratic Programming". In: *Discrete Applied Mathematics* 156.2439–2448 (2008).

[225] R.J. Serfling. "Multivariate Symmetry and Asymmetry". In: *Encyclopedia of Statistical Sciences*. Ed. by S. Kotz et al. Second. Vol. 8. John Wiley and Sons, 2006, pp. 5338–5345.

[226] D.K. Shaw, S. Liu, and L. Kopman. "Lagrangian Relaxation Procedure for Cardinality-Constrained Portfolio Optimization". In: *Optimization Methods and Software* 23.3 (2008), pp. 411–420.

[227] H.D. Sherali. "Personal Communication". 2007.

[228] M. Sipser. *Introduction to the Theory of Computation*. PWS Publishing, 1997.

[229] H. Soleimani, H.R. Golmakani, and M.H. Salimi. "Markowitz-based Portfolio Selection with Minimum Transaction Lots, Cardinality Constraints and Regarding Sector Capitalization using Genetic Algorithm". In: *Expert Systems with Applications* 36.3 (2009), pp. 5058–5063.

[230] F.J. Solis and R.J-B. Wets. "Minimization by Random Search Techniques". In: *Mathematics of Operations Research* 6.1 (1981), pp. 19–30.

[231] B.H. Solnick. "The Advantages of Domestic and International Diversification". In: *International Capital Markets*. Ed. by E.J. Elton and M.J. Gruber. North–Holland, 1975.

[232] L. Sommer. "Daniel Bernoulli. Exposition of a New Theory on the Measurement of Risk". In: *Econometrica* 22.1 (1954), pp. 23–36.

[233] J. Sun. "Rounding-Error and Perturbation Bounds for the Cholesky and LDL^T Factorizations". In: *Linear Algebra and its Applications* 173 (1992), pp. 77–97.

[234] X. Sun, X. Zheng, and D. Li. "Recent Advances in Mathematical Programming with Semi-continuous Variables and Cardinality Constraint". In: *Journal of the Operations Research Society of China* 1.1 (2013), pp. 55–77.

[235] C. Tadonki and J.-P. Vial. "Portfolio Selection with Cardinality and Bound Constraints". University of Geneva. 2003.

[236] F. Tardella. "Connections between Continuous and Combinatorial Optimization Problems through an Extension of the Fundamental Theorem of Linear Programming". In: *Electronics Notes in Discrete Mathematics* 17 (2004), pp. 257–262.

[237] R.H. Tütüncü and M. Koenig. "Robust Asset Allocation". In: *Annals of Operations Research* 132.1–4 (2004), pp. 157–187.

[238] L. Vandenberghe and S. Boyd. "Semidefinite Programming". In: *SIAM Review* 38.1 (1996), pp. 49–95.

[239] S. Vigerske and M.R. Bussieck. "MINLP Solver Software". In: *Wiley Encyclopedia of Operations Research and Management Science (EORMS)*. Ed. by J.J. Cochran et al. John Wiley and Sons, 2010.

[240] J. von Neumann and O. Morgenstern. *Theory of Games and Economic Behaviour*. Princeton University Press, 1944.

[241] B.A. Wallingford. "A Survey and Comparison of Porfolio Selection Models". In: *Journal of Financial and Quantitative Analysis* 2.2 (1967), pp. 85–106.

[242] A.J. Wiles. "Modular Elliptic Curves and Fermat's Last Theorem". In: *Annals of Mathematics* 141.3 (1995), pp. 443–551.

[243] M. Woodside-Oriakhi, C. Lucas, and J.E. Beasley. "Heuristic Algorithms for the Cardinality Constrained Efficient Frontier". In: *European Journal of Operational Research* 213.3 (2011), pp. 538–550.

[244] M.H. Wright. "The Interior-Point Revolution in Optimization: History, Recent Developments, and Lasting Consequences". In: *Bulletin of American Mathematical Society* 42.1 (2004), pp. 39–56.

[245] H.-G. Xue, G.-X. Xu, and Z.-X. Feng. "Mean-Variance Portfolio Optimal Problem under Concave Transaction Cost". In: *Applied Mathematics and Computation* 174.1 (2006), pp. 1–12.

[246] K. Ye, P. Parpas, and B. Rustem. "Robust Portfolio Optimization: A Conic Programming Approach". In: *Computational Optimization and Applications* 52.2 (2012), pp. 463–481.

[247] S. Žaković and B. Rumstem. "Semi-Infinite Programming and Applications to Minimax Problems". In: *Annals of Operations Research* 124.1–4 (2002), pp. 81–110.

[248] S.H. Zanakis and J.R. Evans. "Heuristic "Optimization": Why, When and How to Use It". In: *Interfaces* 11.5 (1981), pp. 84–91.

[249] B. Zhang and B. Chen. "Heuristic and Exact Solution Method for Convex Nonlinear Knapsack Problem". In: *Asia-Pacific Journal of Operational Research* 29.5 (2012), p. 1250031. DOI: 10.1142/S0217595912500315.

[250] B. Zhang and Z. Hua. "A Unified Method for a Class of Convex Separable Nonlinear Knapsack Problems". In: *European Journal of Operational Research* 191.1 (2008), pp. 1–6.

[251] X. Zheng et al. "Successive Convex Approximations to Cardinality-constrained Convex Programs: A Piecewise-linear DC Approach". In: *Computational Optimization and Applications* 59.1-2 (2014), pp. 379–397.

[252] X.J. Zheng, X.L. Sun, and D. Li. "Improving the Performance of MIQP Solvers for Quadratic Programs with Cardinality and Minimum Threshold Constraints: A Semidefinite Program Approach". In: *INFORMS Journal on Computing* 26.4 (2014), pp. 690–703.

université PARIS-SACLAY
ÉCOLE DOCTORALE
Sciences et technologies de l'information et de la communication (STIC)

Titre : L'Algorithme Multiplicative Weights Update pour la Programmation non linéaire en nombres entiers: Théorie, Applications et Limites

Mots clefs : Algorithme Multiplicative Weights Update, Programmation non linéaire en nombres entiers, Sélection du portefeuille moyenne-variance, Sac à dos multiple non linéaire

Résumé : L'objectif de cette thèse consiste à présenter un nouvel algorithme pour la programmation non linéaire en nombres entiers, inspirée par la méthode Multiplicative Weights Update et qui compte sur une nouvelle classe de reformulations, appelées les reformulations ponctuelles.
La programmation non linéaire en nombres entiers est un sujet très difficile et fascinant dans le domaine de l'optimisation mathématique à la fois d'un point de vue théorique et computationnel. Il est possible de formuler de nombreux problèmes dans ce schéma général et, habituellement, ils posent de réels défis en termes d'efficacité et de précision de la solution obtenue quant aux procédures de résolution.
La thèse est divisée en trois parties principales: une introduction composée par le Chapitre 1, une définition théorique du nouvel algorithme dans le Chapitre 2 et l'application de cette nouvelle méthodologie à deux problèmes concrets d'optimisation, tels que la sélection optimale du portefeuille avec le critère moyenne-variance dans le Chapitre 3 et le problème du sac à dos non linéaire dans le Chapitre 4. Conclusions et questions ouvertes sont présentées dans le Chapitre 5.

Title : The Multiplicative Weights Update Algorithm for Mixed Integer NonLinear Programming: Theory, Applications, and Limitations

Keywords : Mixed Integer NonLinear Programming, Multiplicative Weights Update Algorithm, Mean-Variance Portfolio, Multiple NonLinear Knapsack Problem

Abstract : This thesis presents a new algorithm for Mixed Integer NonLinear Programming, inspired by the Multiplicative Weights Update framework and relying on a new class of reformulations, called the pointwise reformulations.
Mixed Integer NonLinear Programming is a hard and fascinating topic in Mathematical Optimization both from a theoretical and a computational viewpoint. Many real-world problems can be cast this general scheme and, usually, are quite challenging in terms of efficiency and solution accuracy with respect to the solving procedures.
The thesis is divided in three main parts: a foreword consisting in Chapter 1, a theoretical foundation of the new algorithm in Chapter 2, and the application of this new methodology to two real-world optimization problems, namely the Mean-Variance Portfolio Selection in Chapter 3, and the Multiple NonLinear Separable Knapsack Problem in Chapter 4. Conclusions and open questions are drawn in Chapter 5.

Université Paris-Saclay
Espace Technologique / Immeuble Discovery
Route de l'Orme aux Merisiers RD 128 / 91190 Saint-Aubin, France