UNIVERSIDAD AUTÓNOMA DE MADRID

Cardinality Constraints and Dimensionality Reduction in Optimization Problems

by

Rubén Ruiz-Torrubiano

A thesis submitted in partial fulfillment for the degree of Doctor of Philosophy

in the
Escuela Politécnica Superior
Computer Science Department

under the supervision of Alberto Suárez González

June 2012

Tribunal nombrado por el Mgfco. y Excmo. Sr. Rector de la Universidad Autónoma de Madrid, el día ..... de ..... de 2012.

Presidente: D. .....

Vocal: D. .....

Vocal: D. .....

Vocal: D. .....

Secretario: D. .....

Realizado el acto de defensa y lectura de la Tesis el día ..... de ..... de 2012 en .....

Calificación: .....

EL PRESIDENTE EL SECRETARIO

LOS VOCALES

"Science is facts; just as houses are made of stone, so is science made of facts; but a pile of stones is not a house, and a collection of facts is not necessarily science"

Jules Henri Poincaré (1854-1912), French mathematician.

Abstract

The design of efficient methods to obtain sparse solutions in optimization problems has become a research area of great interest for a variety of reasons. In some applications sparse solutions are preferred because they are easier to manage and interpret. They also require less memory for storage. Sparse prediction systems have faster response times, which is a desirable feature in online applications. These benefits are especially significant in high-dimensional problems. Another reason is that sparse solutions are generally more robust and stable than solutions that involve all the original variables. In particular, they are less sensitive to variations in the inputs of the optimization. Examples of problems in which sparse solutions are important include optimal portfolio selection and index tracking in quantitative finance, the identification of sparse principal components in statistics and ensemble pruning in machine learning.

A direct procedure to generate sparse solutions is to impose cardinality constraints. The inclusion of these types of constraints generally results in a mixed integer optimization problem. Therefore, the specialized algorithms that are used to solve the unconstrained problem need to be adapted in order to handle the cardinality constraints imposed. In this thesis a hybrid framework is developed that combines metaheuristics and these specialized algorithms to address cardinality constrained optimization problems. Metaheuristics such as genetic algorithms, simulated annealing or estimation of distribution algorithms are used to address the combinatorial aspect of the problem. These metaheuristics iteratively generate candidate solutions that specify only the subset of non-zero variables. The candidate solutions are then evaluated by solving a secondary optimization problem, which is analogous to the original problem but is defined in the restricted subspace proposed by the metaheuristic. This subordinate optimization problem does not have a cardinality constraint and can therefore be efficiently solved by specialized algorithms. To reduce the computational cost of the optimization we also consider the application in a preprocessing step of heuristics that reduce the dimensionality of the problem while preserving the quality of the solutions.

Cardinality constraints are difficult to handle in metaheuristics with standard encodings of the candidate solutions. For instance, genetic algorithms with a binary encoding cannot handle these constraints in a proper manner. The reason is that standard mutation and crossover operators do not preserve these types of constraints. Therefore, one needs to include repair mechanisms or penalty terms in the evaluation of the candidate solutions. However, these ad-hoc strategies may introduce undesirable biases in the search and render it inefficient. In this thesis we propose a set encoding used in conjunction with especially designed mutation and recombination operators that are well adapted to handle cardinality constraints. Including specific domain knowledge in this manner is shown to be an efficient and effective method for addressing cardinality constrained optimization problems in a wide range of application domains.

In the second part of this thesis we apply the specialized hybrid metaheuristic approaches developed to several problems of practical interest: the identification of sparse principal components, the tracking of financial indices and the selection of optimal investment portfolios. In general, the best overall performance is obtained by genetic algorithms with a set encoding and appropriately designed mutation and crossover operators, particularly when they are used in combination with simple dimensionality reduction heuristics. The sparse solutions obtained by means of this approach are found to be stable, robust and in the pertinent cases, have better out-of-sample performance.

An application of hybrid approaches to the construction of consensus trees in phylogenetics concludes this thesis. This problem is an integer linear problem for which exact approaches such as branch-and-cut are not feasible. As a practicable alternative, pruning heuristics and metaheuristic hybridization are combined to obtain high-quality consensus trees.

Resumen

El diseño de métodos para obtener soluciones dispersas en problemas de optimización se ha convertido en un área de investigación de interés por varias razones. En algunas aplicaciones se prefieren soluciones que sean dispersas ya que son más fáciles de mantener y de interpretar. Adicionalmente se requiere menos memoria para su almacenamiento. Los sistemas de predicción dispersos presentan tiempos de respuesta más rápidos, lo cual es una ventaja para aplicaciones en línea. Estos beneficios son de especial importancia en problemas de alta dimensionalidad. Por otro lado, las soluciones dispersas son más robustas y estables que las soluciones que incluyen todas las variables originales. En concreto, son menos sensibles a pequeñas variaciones en los parámetros de entrada de la optimización. Algunos ejemplos de problemas en los que soluciones dispersas son importantes son la selección de carteras de inversión y la replicación de índices en finanzas, la identificación de componentes principales dispersas en estadística y la poda de conjuntos en aprendizaje automático.

Un procedimiento directo para obtener soluciones dispersas es imponer restricciones de cardinalidad. La inclusión de este tipo de restricciones convierte en general el problema de optimización en un problema mixto entero. Por lo tanto, los algoritmos especializados que son utilizados para resolver el problema sin restricciones necesitan ser adaptados para tener en cuenta las restricciones de cardinalidad. En esta tesis se desarrolla un marco híbrido que combina metaheurísticas y algoritmos especializados para resolver problemas con restricciones de cardinalidad. Las metaheurísticas (algoritmos genéticos, temple simulado y algoritmos de estimación de distribuciones) son utilizadas para abordar el aspecto combinatorio del problema. Estos algoritmos generan candidatos a solución de forma iterativa especificando únicamente el subconjunto de variables que tiene un valor no nulo. Los candidatos a solución son evaluados resolviendo un problema de optimización secundario que es análogo al problema original pero que está definido en el subespacio restringido propuesto por la metaheurística. Este problema de optimización subordinado no tiene restricciones de cardinalidad y puede ser resuelto eficientemente por algoritmos especializados. Con el fin de reducir el coste computacional de la optimización consideramos también la aplicación en un paso de procesamiento previo de heurísticas que reducen la dimensionalidad del problema manteniendo la calidad de las soluciones obtenidas.

Las restricciones de cardinalidad son difíciles de tratar para metaheurísticas con codificaciones estándar para los candidatos a solución. Por ejemplo, los algoritmos genéticos con codificación binaria no pueden manejar estas restricciones de manera apropiada. La razón es que, en general, los operadores más comunes para llevar a cabo los procesos de mutación y cruzamiento no mantienen este tipo de restricciones. Por ello, es necesario incluir mecanismos de reparación o términos de penalización en la evaluación de las soluciones candidatas. Sin embargo, estas estrategias ad hoc pueden introducir sesgos en la búsqueda, haciéndola más ineficiente. En esta tesis proponemos una codificación de conjuntos, así como operadores de mutación y recombinación que están especialmente diseñados para manejar restricciones de cardinalidad. Se muestra asimismo como la inclusión de este tipo de conocimiento específico del problema es un método eficaz para resolver problemas con restricciones de cardinalidad en una gran variedad de dominios de aplicación.

En la segunda parte de esta tesis aplicamos los métodos metaheurísticos híbridos diseñados a problemas de interés práctico: la identificación de componentes principales dispersas, la replicación de índices financieros y la selección de carteras de inversión. En general, los mejores resultados se obtienen mediante algoritmos genéticos con codificación de conjuntos y operadores de mutación y cruzamiento diseñados apropiadamente, especialmente si son utilizados en combinación con heurísticas simples para la reducción de dimensionalidad. Las soluciones dispersas obtenidas con este método son estables y robustas y, en los casos en los que es pertinente, presentan un mejor rendimiento fuera de muestra.

Esta tesis concluye con la aplicación de técnicas híbridas al problema de construcción de árboles filogenéticos de consenso. Se trata de un problema entero lineal, para el cual técnicas exactas como *branch-and-cut* no son aplicables en la práctica. Como alternativa viable, se usan heurísticas de poda e hibridación de metaheurísticas en combinación para obtener árboles de consenso de gran calidad.

Acknowledgements

The elaboration of this thesis has enriched me both personally and professionally during all the years since the beginning of my graduate studies. Therefore, I would like to thank in this place all the people that have made this experience possible. In the first place, I am grateful to my advisor Alberto Suárez, who guided me through this difficult journey and provided me with instructive discussions and advice. I could learn from him besides many interesting things about optimization algorithms, that science is not only a field of study, but a way of understanding life.

I would also like to thank Günther R. Raidl and the Algorithms and Data Structures Group of the Technical University of Vienna, Austria, for the 6 months I spent there. During these months, I could learn new ways of doing research and know other perspectives and problems on combinatorial optimization. Particularly, I became interested in general metaheuristic hybridization schemes and mathematical programming techniques. I would like to thank especially Sandro Pirkwieser for the interesting and valuable discussions we had when working on the consensus tree problem.

I am also very grateful to my parents, who supported me all these years, and my brother, from whom I learned many useful things. My friends in Madrid also played a very important role. Thank you too! Thanks also to my parents-in-law, who helped me feeling at home in a new country, which now is part of my life. And last, but not least, I would like to thank my wife. Without her support and her many useful comments and suggestions, I would not have been ever able to complete this thesis. Thank you for making me understand what life is really about!

CONTENTS

Abstract v

Resumen vi

Acknowledgements vii

Abbreviations xiii

1 Introduction 1
- 1.1 Hybrid Algorithms for Cardinality Constrained Problems . . . 5
- 1.2 Dimensionality Reduction . . . 7
- 1.3 Applications . . . 8
- 1.4 Contributions . . . 9
- 1.5 Publications . . . 10
  - 1.5.1 Direct Work . . . 10
  - 1.5.2 Related Work . . . 11
  - 1.5.3 Submitted Work . . . 11
- 1.6 Summary by Chapter . . . 11

I Optimization Algorithms 15

2 Metaheuristics and Cardinality Constrained Problems 17
- 2.1 Introduction . . . 17
- 2.2 Metaheuristic Hybridization for Cardinality Constrained Problems . . . 20
  - 2.2.1 Metaheuristic Hybridization . . . 20
  - 2.2.2 Hybrid Approach for Cardinality Constraints . . . 22
- 2.3 Metaheuristics for optimal subset selection . . . 25
  - 2.3.1 Simulated annealing . . . 25
  - 2.3.2 Genetic Algorithms . . . 27
  - 2.3.3 Estimation of Distribution Algorithms . . . 30
- 2.4 Dimensionality Reduction for Cardinality Constrained Optimization Problems . . . 33

2.4.1 Block pruning . . . 33
2.4.2 Greedy backward selection . . . 33
2.4.3 Greedy forward selection . . . 34
2.5 Summary and Discussion . . . 35

3 Design of Genetic Representations and Operators 37
3.1 Introduction . . . 37
3.2 Forma Theory . . . 39
3.3 Crossover operators . . . 42
3.3.1 Random Respectful Recombination (RRR) . . . 44
3.3.2 Random Equivalence Recombination (RER) . . . 44
3.3.3 Random Assortment Recombination (RAR) . . . 44
3.3.4 Transmitting RAR (TransRAR) . . . 45
3.4 Operators for cardinality constrained sets . . . 46
3.4.1 Complexity analysis . . . 50
3.5 Summary and Discussion . . . 51

II Applications 53

4 The Knapsack Problem 57
4.1 Introduction . . . 57
4.2 Optimization model . . . 58
4.3 Hybrid approaches . . . 59
4.4 Results . . . 60
4.5 Summary and Conclusions . . . 62

5 Sparse Principal Component Analysis 65
5.1 Introduction . . . 65
5.2 Optimization Model . . . 67
5.3 Hybrid approach . . . 69
5.4 Results . . . 70
5.5 Summary and Conclusions . . . 74

6 Index Tracking by Partial Replication 75
6.1 Introduction . . . 75
6.2 Index Tracking with Cardinality Constraints . . . 77
6.3 Hybrid approaches . . . 79
6.3.1 Genetic algorithms . . . 79
6.3.2 Simulated Annealing . . . 80
6.3.3 Estimation of Distribution Algorithms . . . 80
6.3.4 Block pruning . . . 80
6.4 Empirical evaluation . . . 80
6.5 Summary and Discussion . . . 84

7 Optimal Portfolio Selection 87
7.1 Introduction . . . 88

7.2 Previous Work . . . 90
7.3 Portfolio Selection with Cardinality Constraints . . . 92
7.3.1 The Markowitz Mean-variance Model . . . 92
7.3.2 Constraints in Portfolio Selection Problems . . . 94
7.3.3 Hybrid approaches to optimal portfolio selection . . . 95
7.3.3.1 Genetic algorithms . . . 96
7.3.3.2 Simulated Annealing . . . 96
7.3.3.3 Estimation of Distribution Algorithms . . . 96
7.3.4 Empirical evaluation . . . 96
7.3.5 Discussion . . . 106
7.4 Portfolio Selection with Transaction costs . . . 106
7.4.1 Lasso penalties . . . 110
7.4.2 Hybrid Approach to Portfolio Selection under Transaction Costs . 111
7.4.3 Empirical evaluation . . . 115
7.4.3.1 In-sample evaluation . . . 116
7.4.3.2 Out-of-sample evaluation . . . 122
7.4.3.3 Discussion . . . 130
7.5 Summary and Discussion . . . 131

8 The Consensus Tree Problem 133
8.1 Introduction . . . 133
8.2 Optimization model . . . 136
8.2.1 The UpDown Matrix Model . . . 137
8.2.2 The Triplet Model . . . 139
8.3 Solution methods . . . 139
8.3.1 Preprocessing step . . . 139
8.3.2 Using lazy constraints . . . 140
8.3.3 Reduction of dimensionality . . . 140
8.3.4 Refinement of incumbents . . . 140
8.3.4.1 SWAP . . . 141
8.3.4.2 STEP . . . 141
8.3.4.3 NNI or ROTATE . . . 141
8.3.4.4 SPR . . . 141
8.3.5 Generating new variables . . . 144
8.4 Results . . . 145
8.5 Summary and Conclusions . . . 149

9 Conclusions and Future Work 151
9.1 Future Work . . . 153

10 Conclusiones 155

A Appendix for Chapter 3 159
A.1 Proof of Lemma 3.11 . . . 159

B Appendix for Chapter 7 **161**
B.1 Tables for Greedy Backward and Forward Selection 161

C Appendix for Chapter 8 **165**
C.1 Proof of Theorem 8.1 165

Bibliography **167**

ABBREVIATIONS

ARX	**A**uto**R**egressive **E**xogenous
BCP	**B**ranch and **C**ut and **P**rice
BnB	**B**ranch and **B**ound
BnC	**B**ranch and **C**ut
CCQP	**C**ardinality **C**onstrained **Q**uadratic **P**rogram
CTP	**C**onsensus **T**ree **P**roblem
EDA	**E**stimation of **D**istribution **A**lgorithm
EMNA	**E**stimation of **M**ultivariate **N**ormal **A**lgorithm
eRAR	**e**xtended **R**andom **A**ssortment **R**ecombination
GA	**G**enetic **A**lgorithm
ILP	**I**nteger **L**inear **P**rogram
IP	**I**nteger **P**rogram
LASSO	**L**east **A**bsolute **S**hrinkage and **S**election **O**perator
LCA	**L**east **C**ommon **A**ncestor
LP	**L**inear **P**rogram
MA	**M**emetic **A**lgorithm
MCCP	**M**aster **C**ardinality **C**onstrained **P**roblem
MIP	**M**ixed **I**nteger **P**rogram
MIQP	**M**ixed **I**nteger **Q**uadratic **P**rogram
MOEA	**M**ulti-**O**bjective **E**volutionary **A**lgorithm
MSE	**M**ean **S**quared **E**rror
NFL	**N**o **F**ree **L**unch
NNI	**N**earest **N**eighbor **I**nterchange
NP	**N**on-deterministic **P**olynomial-time
PBIL	**P**opulation **B**ased **I**ncremental **L**earning
PCA	**P**rincipal **C**omponents **A**nalysis
QP	**Q**uadratic **P**rogramming
RAR	**R**andom **A**ssortment **R**ecombination
RER	**R**andom **E**quivalence **R**ecombination
RRR	**R**andom **R**espectful **R**ecombination
SA	**S**imulated **A**nnealing
SCP	**S**ubordinate **C**ontinuous **P**roblem
SDP	**S**emi**D**efinite **P**rogramming
SPCA	**S**parse **P**rincipal **C**omponents **A**nalysis

SPR	**S**ubtree **P**rune and **R**egraft
TA	**T**hreshold **A**ccepting
TM	**T**riplet **M**odel
TR	**T**ree **R**ank
TransRAR	**Trans**mitting **R**andom **A**ssortment **R**ecombination
TS	**T**abu **S**earch
UDMM	**U**p**D**own **M**atrix **M**odel
UMDA	**U**nivariate **M**arginal **D**istribution **A**lgorithm
VND	**V**ariable **N**eighborhood **D**escent
VNS	**V**ariable **N**eighborhood **S**earch
WT	**W**eighted **T**riplet

To my family

CHAPTER 1

INTRODUCTION

Optimization is the branch of mathematics concerned with finding the optimal points and values of arbitrary functions $f : \mathcal{D} \to \mathcal{I}$, where $\mathcal{D}$ is the domain of the function to optimize and $\mathcal{I}$ its image. The problem can be stated in general form as follows

$$\begin{aligned} \min_{\mathbf{x}\in\mathcal{D}} \quad & f(\mathbf{x}) \\ \text{subject to} \quad & g_j(\mathbf{x}) \leq 0 \; j = 1, \ldots, m \\ & h_k(\mathbf{x}) = 0 \; k = 1, \ldots, p. \end{aligned} \tag{1.1}$$

In this case the function f is *minimized*. That is, we seek the optimal point $\mathbf{x}^*$, such that $f(\mathbf{x}^*) \leq f(\mathbf{x}) \; \forall \mathbf{x} \in \mathcal{D}$. In minimization problems the function f represents some kind of cost that must be minimized in the subregion of $\mathcal{D}$ implicitly defined by constraints expressed by the functions g_j and h_k. If the function f represents some measure of profit or utility, the goal is to *maximize* the function and we write $\max_{\mathbf{x}\in\mathcal{D}}$ instead. Without loss of generality we will assume throughout this thesis that our general optimization problem is of the form 1.1. A maximization problem can be solved by writing $\hat{f}(\mathbf{x}) = -f(\mathbf{x})$ and minimizing $\hat{f}$ instead of f.

For particular forms of the objective function and the constraints, standard algorithms exist that guarantee finding the optimal solution with a reasonable computational effort. For instance, if both f and the g_j's and h_k's are linear functions, the *simplex* method (Dantzig (1998)) is an efficient procedure to find the optimal solution. More generally, if the optimization problem is convex (that is, f is a convex function with a convex domain $\mathcal{D}$, the functions that appear in the inequality constraints are convex, and the equality constraint functions are affine (Boyd and Vandenberghe (2004))) there are efficient algorithms to perform the optimization that guarantee finding the global optimum (e.g., interior point methods (Adler et al. (1989))). These algorithms can be used to address many optimization problems of practical interest that arise in different

areas of application, such as network and circuit design (Hershenson et al. (2001)), logistics (Yu and Li (2000)), quantitative finance (Markowitz (1987)) and bioinformatics (Han et al. (2007)). However, some optimization problems are not of these special types and cannot be solved exactly in the general case. The inclusion of real-world constraints may also preclude the use of standard techniques to obtain the exact solution. An example is the introduction of decision variables with a discrete domain. These variables can be used to determine additional properties of the system being optimized or to provide some specific information about the solution represented by the real-valued variables. For instance, binary variables can be introduced to represent a subset of objects, which are then characterized by the values of the real variables. Such systems can be described using *mixed-integer real-valued* functions. These are functions of the form $f : \mathbb{R}^N \times \mathcal{B}^M \to \mathbb{R}$, where $\mathcal{B}$ is some subset of the integers $\mathcal{B} \subseteq \mathbb{Z}$. Thus, the domain of the function includes two types of variables: a first block of N real variables, and a second block of M integer variables. The candidate solutions of such an optimization problem can be expressed as a tuple $(\mathbf{w}, \mathbf{z})$, where $\mathbf{w} \in \mathbb{R}^N$ and $\mathbf{z} \in \mathcal{B}^M$. If $N = 0$, the optimization task is a *pure integer problem.*

An important kind of mixed-integer problems is the class of *cardinality constrained problems.* These are problems in which the number of real-valued variables that are different from zero is constrained to be lower than a given upper bound. Consider the objective function $f : \mathbb{R}^N \to \mathbb{R}$. To introduce a cardinality constraint we replace the original objective function f by an extended function $\hat{f} : \mathbb{R}^N \times \{0,1\}^N \to \mathbb{R}$. In the domain of this extended function, each of the binary variables z_i encodes whether variable w_i has a value different from zero ($z_i = 1$) or not ($z_i = 0$). In terms of these auxiliary variables the cardinality constraint is

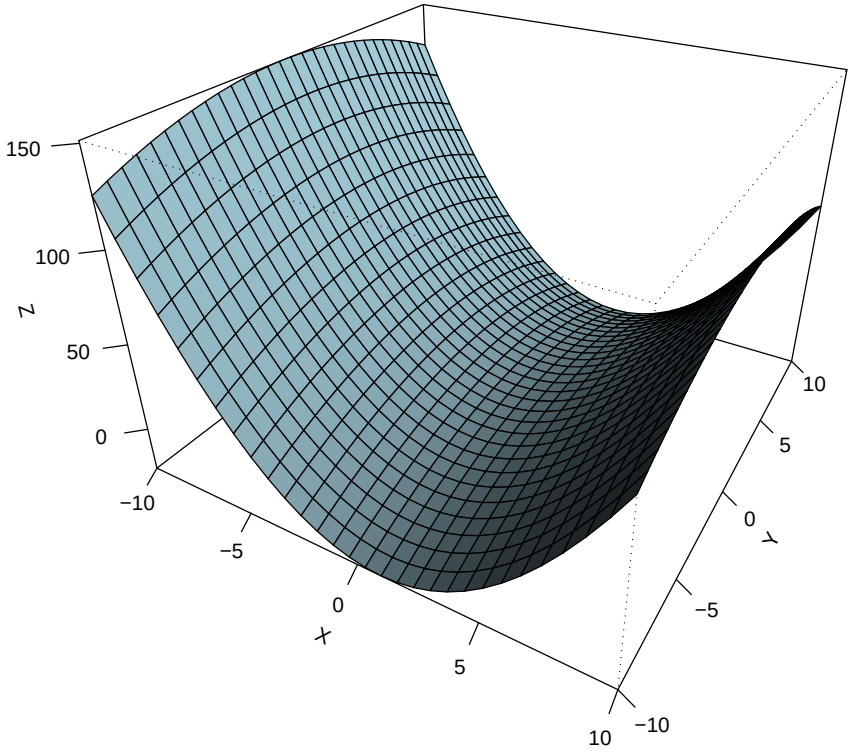
$$\sum_{i=1}^{N} z_i \leq K, \tag{1.2}$$

where $K < N$ is a specified upper bound. To ensure that the corresponding variables are effectively 0, one can use inequality constraints of the form $l_i z_i \leq w_i \leq u_i z_i$, where l_i and u_i are the lower and upper bounds on the value of w_i, respectively. If z_i is set to zero, this constraint forces $w_i = 0$. Otherwise, if $z_i = 1$, the value of w_i is constrained to lie in the interval $[l_i, u_i]$.

One of the important properties of cardinality-constrained problems is that it is possible to separately address the combinatorial optimization problem of determining the values of the indicator variables $\{z_i\}_{i=1}^N$ that satisfy the cardinality constraint and the optimization problem in $\mathbb{R}^N$ of determining the value of the components of $\{w_i\}_{i=1}^N$ for which $z_i = 1$. This *separation property* will be illustrated by means of a simple example: Consider the minimization of the function

$$f(x, y) = 5\Big(\frac{x-1}{2}\Big)^2 - \Big(\frac{y}{2}\Big)^2 \tag{1.3}$$

Figure 1.1: Hyperbolic paraboloid as sample objective function.



in $\mathbb{R}^2$. Let the space of feasible solutions be

$$\begin{aligned} -10 \leq x \leq 10, \\ -10 \leq y \leq 10. \end{aligned}$$

A plot of f in the feasible region is given in Figure 1.1. There are two minima: (1,-10) and (1,10). Assume that a cardinality constraint is included so that only one of the variables, either x or y, can be different from zero. This constraint can be enforced by introducing two auxiliary binary variables $z_1, z_2 \in \{0, 1\}$ and defining the extended function

$$\hat{f}(x, y, z_1, z_2) = 5\Big(\frac{x-1}{2}\Big)^2 - \Big(\frac{y}{2}\Big)^2. \tag{1.4}$$

The space of feasible solutions is

$$\begin{aligned} -10z_1 \leq x \leq 10z_1 \\ -10z_2 \leq y \leq 10z_2 \\ z_1 + z_2 = 1. \end{aligned}$$

This problem is a mixed-integer quadratic program (MIQP). It is possible to transform the problem into a pure integer program

$$\min_{z_1,z_2} \quad F(z_1, z_2) \tag{1.5}$$

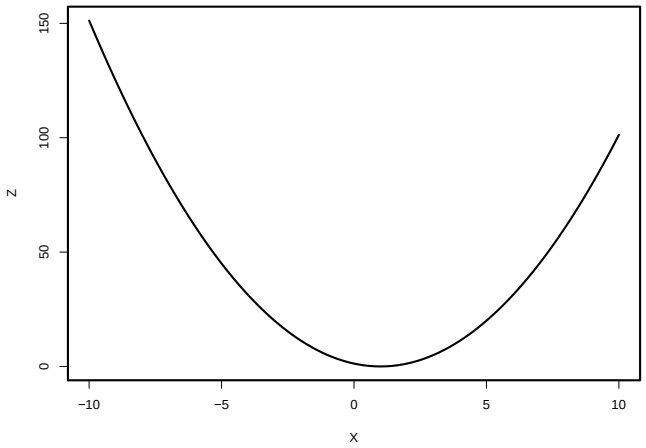
$$\text{subject to} \quad z_1 + z_2 = 1 \tag{1.6}$$

where F is defined as

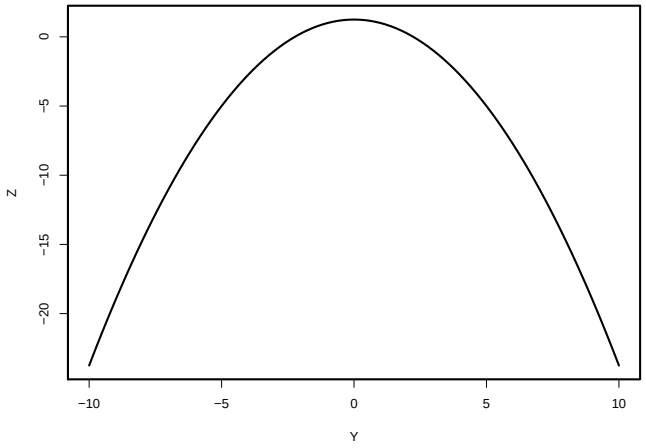
$$F(z_1, z_2) = \begin{cases} \min_x 5\left(\frac{x-1}{2}\right)^2 & -10 \leq x \leq 10, \text{ if } z_1 = 1, z_2 = 0 \\ \min_y \frac{5}{4} - \left(\frac{y}{2}\right)^2 & -10 \leq y \leq 10, \text{ if } z_1 = 0, z_2 = 1. \end{cases} \tag{1.7}$$

Figure 1.2 shows the objective functions for these two sub-problems. There are only two feasible solutions to problem (1.5)-(1.6): (1,0) and (0,1). The first one results in the one-dimensional optimization problem represented in Figure 1.2(a). The optimum of this problem is reached at the value $x = 1$. The second solution results in the problem depicted in Figure 1.2(b). The optimum of this sub-problem is achieved at $y = -10$ and $y = 10$. The optimal value in the first case is 0, whereas in the second one we have -95/4. We therefore choose the tuple $(z_1, z_2) = (0, 1)$ as optimal solution of the integer problem. As a result, the optimal solutions of the original cardinality constrained optimization problem are $(x, y) = (0, -10)$ and $(x, y) = (0, 10)$.

In summary, instead of tackling this mixed-integer problem as a whole, it has been divided into a combinatorial problem and continuous optimization subproblems. For every candidate solution in the binary search space, a continuous subproblem is defined in the reduced space determined by the binary variables. The continuous optimization subproblem is of the same type as the original problem, except that it is defined in a space of lower dimension and does not have any cardinality constraints. In this small example, it was possible to generate all possible candidate solutions for the combinatorial optimization problem and then solve the corresponding continuous optimization subproblems. However, this direct method cannot be implemented in practice. The combinatorial search space of the binary variables is too large for exhaustive search to be a feasible strategy. For instance, for 100 variables and a strict cardinality constraint of 30, the number of possible solutions in the binary space is approximately $3 \cdot 10^{25}$. If it were possible to solve each associated continuous subproblem in 1 millisecond, one would need more than 10^{13} years to obtain the optimal solution. Therefore, we propose to use an approximate method to search in this combinatorial optimization space. This approximate method can then be combined with an exact or approximate continuous optimization technique to obtain near optimal solutions efficiently and reliably.



(a) Projection of the search space when $y = 0$



(b) Projection of the search space when $x = 0$

Figure 1.2: Projections of the search space for all possible solutions with cardinality one.

1.1 Hybrid Algorithms for Cardinality Constrained Problems

In this thesis a general framework for the solution of optimization problems with cardinality constraints is developed. To this end we define a *master cardinality-constrained problem* (MCCP), in which the goal is to find the optimal values of the indicator variables that satisfy the specified cardinality constraint. The objective function of the MCCP is defined as the solution of the optimization problem in $\mathbb{R}^N$ for the remaining parameters. The MCCP is a combinatorial optimization problem that can be addressed using metaheuristic approaches. However, standard formulations of these approaches are in general ill-suited to handle cardinality constraints. For instance, in genetic algorithms the use of a binary encoding with standard crossover and mutation operators frequently

results in individuals that violate the cardinality constraints. It is possible to reduce the presence of these individuals in the population by penalizing their fitness. Alternatively heuristic repair mechanisms can be used to derive feasible individuals from them. One of the findings of this thesis is that such procedures tend to misguide the search process and obtain poor results in practice. Moreover, black-box metaheuristic approaches, in which specific domain knowledge is not incorporated in the algorithm, cannot perform better than random search when averaged over all problem instances, as established by the *no-free-lunch theorems* in optimization (Wolpert and Macready (1997)).

To design genetic algorithms that can handle cardinality constraints effectively, we resort to *forma theory* (Radcliffe (1994)). This framework is a generalization of schema theory (Holland (1975)) in which the building blocks used are *formae* instead of schemata. A forma is defined as an equivalence class resulting from the definition of equivalence relations that make the structure of the search space explicit. Individuals in the same equivalence class should share some characteristic that is relevant for the solution of the problem considered. The main idea is to incorporate specific problem knowledge to define appropriate equivalence relations, which in turn yield a useful genetic representation. The definition of an appropriate basis of equivalence relations for cardinality constrained problems results in a *set representation* for genetic algorithms that will be used throughout this thesis. The resulting set genetic algorithms are compared to other metaheuristic approaches such as simulated annealing (Kirkpatrick et al. (1983)) and estimation of distribution algorithms (Larrañaga and Lozano (2002)), which are also adapted to address problems with cardinality constraints.

Additionally, genetic operators that are independent of the particular representation used can be defined on the basis of properties such as assortment, transmission and respect that are useful to guide the search: Assortment is the capability of an operator to produce all possible meaningful combinations of the parents. Transmission requires that the resulting children share all their characteristics with some parent. Respect ensures that the relevant information that is present in the parents is inherited by the children. Another contribution of this thesis is the design of a new representation-independent crossover operator called *transmitting random assortment recombination* (TransRAR). This operator combines the concepts of forma theory in a novel way. Specifically, it gives more importance to the property of transmission, instead of to assortment or respect. As for other crossover operators defined in the framework of forma theory, a general design of TransRAR operators is given. This general design can then be instantiated in the particular genetic representation used to encode the candidate solutions.

A second building block for the general framework proposed to address cardinality constrained problems is the solution of the subordinate optimization problem that defines the objective function for the MCCP. This problem will be referred to as the *subordinate continuous problem* (SCP). In this thesis, we focus on applications in which this subordinate problem is convex. A large number of convex optimization methods exist to obtain global optimal solutions to the SCP. Among these methods we find the simplex algorithm for linear programming (Dantzig (1998)), interior point methods (Adler et al. (1989)) and quadratic programming (Gill et al. (1991)). In this thesis, we mostly use quadratic programming for the SCP.

In summary, the approach developed in this thesis to solve optimization problems with cardinality constraints consists in dividing the original problem into a discrete and a continuous part. The discrete problem involves the search for the optimal subset of the original variables that satisfies the cardinality constraint. A combinatorial optimization metaheuristic is used to carry out this search. The objective function of this combinatorial optimization problem is the solution of the original problem in the variables specified by each candidate subset. This subordinate problem is solved using standard continuous optimization techniques. The optimal value is then used as a measure of the quality of the candidate solutions generated by the optimization metaheuristic.

1.2 Dimensionality Reduction

Many optimization problems of practical interest involve searching in spaces of large dimensionality. However, because of redundancies, the intrinsic dimensionality of the search space is often much lower than the space in which the original problem is formulated. In problems with cardinality constraints, the elimination of variables whose value in the optimal solution is zero does not affect the quality of the solutions obtained. Therefore, in these types of problems, one can use heuristics to identify variables that are not likely to be included in the final solution and eliminate them from the problem without a deterioration of the quality of the solution. If effective heuristics can be designed, one can take advantage of these dimensionality reduction techniques to improve the efficiency of the search.

The increase in the dimensionality of the problem affects optimization methods differently. For instance, estimation of distribution algorithms seem to have greater difficulties with high-dimensional search spaces than genetic algorithms (Ruiz-Torrubiano and Suarez (2010)). Therefore, for these methods, it is important to identify dimensionality reduction techniques that do not significantly deteriorate the quality of the final solutions. In this thesis three pruning heuristics are proposed to address this issue. The first one, *block pruning*, uses the optimal solution of a relaxed version of the problem to decide which variables to eliminate. In this case, the variables whose absolute value is lower than a given threshold are eliminated from the problem. A more sophisticated method, *greedy backward selection*, proceeds by eliminating in each iteration the variable that has the lowest absolute value in the solution of a relaxed version of the problem. The algorithm stops after a given number of iterations. In contrast, the *greedy forward selection* heuristic incorporates in each iteration the variable that reduces the optimal objective value the most. This method also terminates after a number of iterations determined by an user-specified parameter. The heuristics proposed are executed as a preprocessing step before applying the combinatorial optimization metaheuristic.

Dimensionality reduction techniques are also investigated in the context of integer programming for the Consensus Tree Problem (CTP). In this case, ad-hoc pruning heuristics are proposed to reduce the amount of input variables while preserving the quality of the solutions obtained by an exact branch-and-cut algorithm. It is shown that by carefully designing the pruning heuristic, optimal solutions can be obtained as well.

Additionally, the possibility of adding new variables as they are needed is investigated using a *heuristic column generation* scheme. Exact column generation has widely been used in applications of linear programming techniques in high-dimensionality problems (Lübbecke and Desrosiers (2005)). However, an exact column generation scheme for the CTP is difficult to formulate and to solve. Therefore, a heuristic is proposed that generates new variables dynamically. This heuristic is embedded in the branch-and-cut solver and incorporates new variables as soon as they appear in rounded solutions of LP relaxations of the problem.

1.3 Applications

The second part of this thesis reviews applications of the techniques described in the previous section in problems of practical interest. In particular, the general hybrid framework for optimization problems with cardinality constraints and dimensionality reduction techniques are applied to the problems of finding sparse principal components in statistics, index tracking and portfolio selection in quantitative finance.

The goal of sparse PCA is to find principal components with only a few non-zero loadings (Zou et al. (2006)). The task can be formulated as a quadratic program with a cardinality constraint. The cardinality constraint limits the number of non-zero coefficients that can appear in each principal component. For this problem, we analyze the performance of several metaheuristic approaches and quadratic programming and perform a comparison with a recent method based on semidefinite programming (d'Aspremont et al. (2007)). Results with real and synthetic data show that it is possible to identify sparse principal components that explain more variance than state-of-the-art methods in the problems analyzed.

The index tracking problem in quantitative finance consists in building an investment portfolio that tracks the behavior of a particular financial index used as reference (Markowitz (1987)). It is possible to track a financial index using *full replication* (Beasley et al. (2003)); i.e., building a portfolio that invests on the same assets and with equal proportions as the reference index. However, the resulting portfolio is typically large and difficult to manage. To avoid this problem, we wish to track the evolution of the index as closely as possible but investing in only a subset of the assets included in the index. To limit the number of products used to track the index, one can introduce a cardinality constraint in the problem formulation. Various metaheuristics and dimensionality reduction techniques are applied to obtain high-quality solutions in an efficient manner.

In the problem of optimal portfolio selection (Markowitz (1952)), cardinality constraints appear as a result of established management practices (Chang et al. (2000)). For instance, an investor may want to limit the number of assets in her or his portfolio to make portfolio rebalancing easier and to minimize transaction costs. This raises the question of how to apply the algorithms developed for the one-period portfolio selection problem in a multi-period scenario. In multi-period portfolio optimization, a sequence of investment decisions is considered. In this context it is important to take into account

the transaction costs incurred when the composition of the portfolio is modified. In this thesis we compare various investment strategies. The comparison is made in terms of both the in-sample and the out-of-sample preformance of the portfolios selected. As observed in previous studies, portfolios whose composition is optimal in a particular period need not have a good performance in subsequent periods (DeMiguel et al. (2009b); DeMiguel and Nogales (2009)). Therefore, in-sample performance is often not a good predictor of the out-of-sample performance. As shown by our results, both cardinality constraints and transaction costs play an important role as regularization terms in the optimization problem. This regularization allows the identification of portfolios that are robust with respect to small variations in the input data (namely, the expected returns of the individual assets and correlations among the asset returns) and generally have good out-of-sample performance.

Finally, the thesis investigates the problem of how to summarize the information provided by a collection of phylogenetic trees obtained by different methods in a single consensus tree. The problem can be formulated as an integer linear optimization problem using different quality metrics. The purpose of our study is to apply general integer programming techniques and develop hybrid approaches based on these techniques to solve this problem. As a result of our work, a number of improvements on the classical branch-and-bound solution technique are presented. Among these improvements, the design and application of heuristics to reduce the space of input variables and the use of a heuristic column generation scheme are the most important contributions of this thesis in this area of research.

1.4 Contributions

The original contributions of this thesis can be summarized as follows

- A general framework for solving problems with cardinality constraints by dividing the problem into a pure combinatorial and a continuous optimization part. Similar decomposition schemes have been applied in previous studies to the index tracking problem (see Shapcott (1992) and Jeurissen and van den Berg (2008)) and, in recent independent work for constrained portfolio selection (DiGaspero et al. (2007)). The formulation presented in this thesis is general and can be applied in a wide range of domains. To address the effectiveness of this approach, we have compared strategies that use simulated annealing, genetic algorithms or estimation of distribution algorithms to address the combinatorial aspect of the problem and quadratic programming for the continuous part.

- A comparison of different encodings for genetic algorithms especially regarding the way in which the cardinality constraints are handled. These include a binary encoding with penalty and repair mechanisms and a set encoding. The most effective genetic algorithms use set encoding in combination with specially designed mutation and crossover operators that preserve the cardinality constraint.

- A new crossover operator (TransRAR), which has been designed using guidelines derived from the theory of formae. General specifications are given for the implementation of this operator, which is then instantiated for the set representation proposed. In the problems investigated, the genetic algorithms that employ this operator outperform genetic algorithms with other set crossover operators (e.g., the RAR operator), which have been shown to be very effective in previous studies.

- The design and application of dimensionality reduction techniques (block pruning, greedy backward and forward selection) for optimization problems with cardinality constraints. In the problems investigated these techniques improve the efficiency of the search without a significant deterioration of the quality of the solutions obtained.

- Application and extensive empirical evaluation of the proposed techniques in the problems of sparse principal components, index tracking, and portfolio selection.

- In the portfolio selection problem we have provided empirical evidence that both transaction costs and cardinality constraints have a regularization effect, which is useful to build portfolios that are robust, stable and have good out-of-sample performance.

- New methods in the context of integer linear programming for solving the consensus tree problem in phylogenetics. These contributions include the use of lazy constraints, pruning of input variables, metaheuristic incumbent solution improving and heuristic column generation.

1.5 Publications

The following work was published as a result of the investigations performed in the course of this thesis. The list of publications is presented in antichronological order. It is divided in three blocks: *(i)* direct work, which includes articles directly related to the contents of this thesis, *(ii)* related work, which considers results connected to the topics presented in this thesis, and *(iii)* submitted work, which includes articles currently under review for publication.

1.5.1 Direct Work

- Ruiz-Torrubiano, R., and Suárez, A. (2011). The TransRAR crossover operator for genetic algorithms with set encoding. *Proceedings of the 13th annual conference on Genetic and Evolutionary Computation GECCO 2011, Dublin, Ireland*, pp. 489-496. ACM New York.

- Ruiz-Torrubiano, R., García-Moratilla, S., and Suárez, A. (2010). Optimization problems with cardinality constraints. In Tenne, Y., and Goh C. K. (editors) *Computational Intelligence in Optimization: Implementations and Applications* pp. 105-130. Springer.

- Ruiz-Torrubiano, R., and Suárez, A. (2010). Hybrid approaches and dimensionality reduction for portfolio selection with cardinality constraints. *IEEE Computational Intelligence Magazine* 5(2):92-107.

- Ruiz-Torrubiano, R., and Suárez, A. (2009). A Hybrid Optimization Approach to Index Tracking. *Annals of Operations Research* 166(1):57-71.

- Pirkwieser S., Ruiz-Torrubiano, R., and Raidl, G. R. (2008). Exact methods and metaheuristic approaches for deriving high quality fully resolved consensus trees. *Proceedings of the 2nd International Conference on Bioinformatics Research and Development BIRD 2008, Poster Presentations, Vienna, Austria. Schriftreihe Informatik 26* pp. 115-124. Trauner Verlag.

- Ruiz-Torrubiano, R., and Suárez, A. (2007). Use of heuristic rules in evolutionary methods for the selection of optimal investment portfolios. *Proceedings of the IEEE World Congress on Evolutionary Computation CEC 2007, Singapore* pp. 212-219. IEEE.

- Moral-Escudero, R., Ruiz-Torrubiano, R., and Suárez, A. (2006). Selection of optimal investment portfolios with cardinality constraints. *Proceedings of the IEEE World Congress on Evolutionary Computation CEC 2006, Vancouver, Canada* pp. 2382-2388. IEEE.

1.5.2 Related Work

- Hernández-Lobato, D., Hernández-Lobato, J. M., Ruiz-Torrubiano, R., and Valle, Á. (2006). Pruning adaptive boosting ensembles by means of a genetic algorithm. In Corchado, E., Yin, H., Botti, V. J., and Fyfe, C. (editors) *Proceedings of the 7th International Conference on Intelligent Data Engineering and Automated Learning*, Vol. 4224 *Lecture Notes in Computer Science*, pp. 322-329. Springer.

1.5.3 Submitted Work

- Ruiz-Torrubiano, R., and Suárez, A. A memetic algorithm for cardinality-constrained portfolio optimization under transaction costs.

1.6 Summary by Chapter

This thesis is organized in two parts. Part I introduces the adaptation and improvements of the optimization algorithms developed to address problems with cardinality constrains. The second part presents the application of these techniques to problems of practical interest. By chapters, the contents of the thesis are as follows:

Chapter 2 presents a review of hybridization methods, metaheuristics and memetic algorithms. A general decomposition of optimization problems with cardinality constraints in two components is introduced: The original optimization problem is formulated as a combinatorial search guided by the solution of a subordinate continuous optimization problem. An optional third component, which involves the use of preprocessing heuristics that reduce the dimensionality of the search space, is also described in this chapter. We also discuss the advantages and disadvantages of the different metaheuristic techniques that can be used to address the combinatorial part of the optimization problem. In particular simulated annealing, genetic algorithms and estimation of distribution algorithms are reviewed in detail.

The focus of **Chapter 3** is on the design of genetic representations and crossover operators for problems with cardinality constraints within the framework of forma theory. Specifically, the concepts of respect, assortment and transmission are introduced as desirable properties that crossover operators should have. These properties are used to guide the design of representation independent crossover operators, such as Random Equivalence Recombination (RER), Random Respectful Recombination (RRR) and Random Assortment Recombination (RAR). Additionally, a new crossover operator based on these design principles is introduced: Transmitting Random Assortment Recombination (TransRAR). Finally, we introduce a set representation and particularize the aforementioned operators for a genetic algorithm that is adapted to handle problems with cardinality constraints.

Chapter 4 presents an analysis of the 0/1 knapsack problem using the ideas presented in the previous chapters. A general knapsack problem is expressed as a collection of cardinality constrained problems. Both the efficiency and the quality of the solutions obtained by the different metaheuristic approaches analyzed are compared with an exact branch-and-bound optimization method.

Chapter 5 deals with the application of the methods developed to the problem of constructing sparse principal components (SPCA). One of the disadvantages of PCA is that, in general, principal components are linear combinations of all the original variables. This makes the interpretation of these components difficult. Sparse principal components improve the intelligibility of the results, because only a few of the problem variables have a non-zero loading. The problem can be formulated as a mixed integer quadratic program with a cardinality constraint. Therefore, the general combinatorial optimization scheme introduced in this thesis can be used in conjunction with a quadratic solver to address the subordinate continuous optimization problem. Experiments using synthetic data and data from real-world applications show that this hybrid approach can achieve results that are comparable or better than state-of-the-art approaches, such as semidefinite programming.

Chapter 6 is devoted to the index tracking problem. This is a problem in quantitative finance whose goal is to construct an investment portfolio that tracks a financial index of reference over a period of time. A cardinality constraint is introduced in the model to limit the number of assets in the tracking portfolio. The objective function of the problem is chosen as the mean squared deviation between the returns of the index and those of the tracking portfolio. This measure results in the formulation of the

problem as a mixed integer quadratic program. The approach presented in this chapter obtains high-quality results both in-sample and out-of-sample for tracking several financial indices. Additionally, a pruning technique is used to improve the performance of the algorithm without significantly affecting the tracking accuracy.

In **Chapter 7** the results of applying these hybrid techniques to the portfolio selection problem are presented. The framework used is the classical Markowitz model with constraints limiting the amount that is invested on each asset or on groups of assets. A cardinality constraint on the maximum number of assets that can be included in the portfolio is introduced. This constraint converts the problem into a mixed integer convex optimization problem. In the first part of this chapter, the single period selection problem with cardinality constraints is investigated using hybrid optimization algorithms. Various pruning heuristics are also applied to obtain solutions of comparable quality with a lower computational cost. In the second part, the multi-period problem with cardinality constraints and piecewise linear transaction costs is solved using a hybrid genetic algorithm. The performance of this optimization technique is compared to other investment strategies, including an equally-weighted $(1/N)$ portfolio and a portfolio obtained by optimization with a lasso term that penalizes changes in the composition of the portfolio. A conclusion that can be derived from the analysis of the empirical results is that ignoring transaction costs leads to the selection of inefficient investment strategies that involve large changes in the composition of the portfolio. Introducing cardinality constraints also has a regularization effect in the optimization problem. As a result of this regularization, the investment strategies select portfolios that are more stable and exhibit good out-of-sample performance.

In **Chapter 8** several hybrid optimization methods are proposed for the consensus tree problem in phylogenetics. The goal in this problem is to build a phylogenetic tree that summarizes as well as possible the information contained in a collection of input trees. The trees in the input collection may be obtained by several methods based on molecular information, or by several runs of the same non-deterministic procedure. The similarity metrics used result in the formulation of the problem as an integer linear program. The methods presented in this chapter are based on the use of a branch-and-cut approach. Since this exact technique may require an exponential number of steps to find the optimal solution, several methods are presented to improve its performance in a heuristic manner. Among these techniques, we propose to use lazy constraints, metaheuristic incumbent solution improving and heuristic variable addition. Moreover, a pruning technique is presented that reduces the size of the search space. Improvements in performance due to the pruning algorithm can be seen from the analysis of the results of experiments on natural and synthetic trees.

Chapter 9 presents a summary of the conclusions of this thesis and outlines some topics for future research.

Part I

Optimization Algorithms

CHAPTER 2

METAHEURISTICS AND CARDINALITY CONSTRAINED PROBLEMS

Finding sparse solutions to complex optimization problems has attracted much attention in different research areas (Zou et al. (2006), Brodie et al. (2009), Wu et al. (2006)). Sparse solutions can be desirable either to satisfy user preferences, objectives and restrictions (e.g. sparse solutions are easier to understand and to manage) or to improve their stability, robustness and out-of-sample performance. A way of enforcing sparsity is to impose cardinality constraints that limit the number of non-zero variables in the optimal solution. However, these types of restrictions introduce a combinatorial component in the optimization, which generally precludes the direct use of standard optimization techniques. General combinatorial metaheuristics such as simulated annealing, genetic algorithms and estimation of distribution algorithms can handle optimization problems that involve discrete components. Nonetheless, the continuous nature of the unconstrained problem or the interaction with other constraints can diminish the efficiency of such metaheuristics. A possible approach is to use the metaheuristic to handle the discrete optimization only. The metaheuristic produces candidate solutions that define a subspace of the original search space. These candidate solutions are then evaluated by the solution of the original optimization problem in the restricted subspace, without cardinality constraints. The subordinate optimization problem defined in this manner can then be solved using standard optimization techniques.

2.1 Introduction

A cardinality constraint in an optimization problem imposes an upper bound on the number of variables that have non-zero values in the optimal solution. We consider the case in which this constraint is hard. Therefore, solutions that violate the constraint, even if it is by a small amount, are not acceptable. Cardinality constraints introduce a combinatorial element in the optimization problem that generally increases the difficulty of the problem.

Including a cardinality constraint in an optimization problem is a direct method to obtain sparse solutions. Numerous recent research efforts have been directed to the design of methods to obtain sparse solutions, especially in the machine learning community (Zou et al. (2006), Moghaddam et al. (2005), Argyriou et al. (2007), Wu et al. (2006), Jacob et al. (2009)). Sparse solutions exhibit desirable statistical properties, such as stability, robustness and good generalization performance. In a Bayesian framework, sparsity can be favored by assuming special kinds of priors that assign a large probability to zero values and simultaneously have large probability mass in a wide range of non-zero values. Some examples are Laplace (Seeger (2008)), Student's t (Tipping (2001)), horseshoe (Carvalho et al. (2009)) and spike-and-slab (George and McCulloch (1997)) priors. One can also obtain sparsity by including special types of penalties in the objective function. An example of this approach is the *lasso* method ("least absolute shrinkage and selection operator", Tibshirani (1996)), originally introduced in the context of least-squares regression. The lasso is implemented by introducing a constraint on the value of the L_1 norm of the solution vector $\|\mathbf{x}\|_1$. Alternatively, the constraint can be enforced by including a lasso penalty term proportional to $\|\mathbf{x}\|_1$ in the objective function. The advantage of such a formulation is that, if the objective function and the constranits satisfy some regularity conditions, convex optimization methods can be directly applied to solve the problem (Efron et al. (2004), Turlach (2005)). A drawback of these approaches to the selection of sparse solutions is that it is not possible to directly control the number of non-zero variables in the final solution. The degree of sparsity can only be controlled indirectly through the parameters of the sparsifying prior in Bayesian approaches or through the strength of the sparsity-inducing penalty in the lasso. By contrast, the cardinality constraint directly specifies the sparsity of the solution vector.

Optimization problems with cardinality constraints are common in different areas of practical interest. Some examples of applications in which cardinality constraints are relevant are:

- **Pruning of learning ensembles**: Consider a collection of instances $\{\mathbf{x}_i, y_i\}_{i=1}^{M}$, in which $\mathbf{x}_i$ is the vector of attributes that characterizes the ith instance and y_i is the corresponding label (a categorical variable in classification problems, a real variable in regression). The goal of supervised learning is to automatically induce from these labeled instances (the *training* data) a system that accurately predicts the label of a previously unseen test instance on the basis of $\mathbf{x}_{test}$ alone. Learning ensembles are composed of a collection of such predictors. The ensemble prediction is a combination of the individual member outputs. Ensembles have been shown to be accurate and robust prediction systems. However, they generally have large storage requirements. Furthermore the time needed to obtain a prediction increases linearly with the size of the ensemble, which can be a disadvantage for online applications. To alleviate these drawbacks one can select a subset of the classifiers of a specified maximum size while preserving (and in some cases improving) the prediction accuracy of the system. This process is referred to as ensemble pruning in literature and can be formulated as a subset selection problem.

- **Sparse Principal Component Analysis**: In the field of statistical data analysis, principal components are used to reduce the dimensionality of a given dataset. The objective is to find a few linear combinations of the input variables that are uncorrelated and account for as much of the variance in the data as possible. Typically, these principal components have loadings in all the input variables, which makes their interpretation difficult. Sparse principal components analysis seeks to obtain linear combinations of the original variables with as few non-zero entries as possible, which still explain most of the variance. To achieve this goal, a cardinality constraint on the maximum number of non-zero entries for each principal component can be used. Common applications of this technique are data compression and visualization.

- **Portfolio selection in quantitative finance**: A portfolio is a collection of financial products (assets, bonds, cash) held by an investor. The goal of optimal portfolio selection is to maximize the expected wealth while minimizing the risk of the investment, subject to various constraints resulting from investment preferences, market restrictions and other requirements. The standard framework for portfolio optimization was originally proposed by H. Markowitz in 1952. In this framework, the returns of the assets considered for investment are modeled as random variables. The return of the portfolio is simply a convex combination of the returns of the individual assets. The expected return of the portfolio is the mean of this random variable. The risk of the portfolio is quantified in terms of the variance of the portfolio returns. A common requirement in practice is to limit the number of products included in the portfolio. This constraint facilitates portfolio rebalancing and, as will be shown in this thesis, improves the robustness, stability and out-of-sample performance of the portfolio.

- **Financial index tracking**: The goal of index tracking by partial replication is to design an investment portfolio that tracks as closely as possible the evolution of a specified financial index using only a subset of the products that are considered in the construction of the index. The problem is formulated as a minimization problem in which the cost function is the mean squared error (MSE) between the returns of the portfolio and those of the index. It may also be desirable to set lower and upper bounds on the weights of assets or groups of assets in the replicating portfolio. The number of products in which the final portfolio invests can be directly limited including a cardinality constraint.

- **Subset selection in multiple linear regression**: The goal of multiple linear regression is to describe the relationship between the explanatory variables $\mathbf{x}$ and the real-valued response variable y by fitting a linear model $\boldsymbol{\beta}^T \cdot \mathbf{x}$ to some observed data $\{\mathbf{x}_i, y_i\}_{i=1}^M$. The vector of regression coefficients $\boldsymbol{\beta}$ can be obtained by minimizing the mean-squared error $MSE = \frac{1}{M}\sum_{i=1}^M (y_i - \boldsymbol{\beta}^T \mathbf{x}_i)^2$. A cardinality constraint can be introduced to limit the number of non-zero components in $\boldsymbol{\beta}$. This constraint generally reduces overfitting and allows the selection of models that are stable, robust and have good generalization properties. The sparsity of the solution also improves the interpretability of the resulting model.

Cardinality constraints are therefore important to improve the quality and facilitate the interpretation of the solutions of numerous problems of practical interest. Therefore it is important to design optimization techniques that can handle these constraints efficiently and effectively. The research presented in this thesis focuses on optimization problems that are convex in the absence of the cardinality constraint. To address the combinatorial complexity introduced by the cardinality constraints we introduce a hybrid strategy that combines general metaheuristic approaches with standard convex optimization algorithms.

The organization of this chapter is as follows: Section 2.2 presents an overview on how metaheuristics can be hybridized with standard optimization techniques to handle optimization problems with cardinality constraints. The metaheuristics simulated annealing, genetic algorithms and estimation of distribution algorithms, which are used in different parts of this thesis, are described in Section 2.3. Section 2.4 introduces several dimensionality reduction heuristics that can be used as a preprocessing step to improve the efficiency of the metaheuristics. Section 2.5 summarizes the key ideas introduced in this chapter.

2.2 Metaheuristic Hybridization for Cardinality Constrained Problems

In this section we first give a general overview of metaheuristic algorithms and introduce the concept of metaheuristic hybridization. Next we describe the hybrid approach that is used in this thesis to address optimization problems with cardinality constraints. Particular attention is given to the benefits of such an approach over metaheuristics alone.

2.2.1 Metaheuristic Hybridization

A metaheuristic is a procedure that iteratively generates candidate solutions by intelligently combining exploration and exploitation of the search space (Osman and Laporte (1996)). The exploration and exploitation components of the metaheuristic are guided by a learning strategy whose aim is to generate good candidate solutions. Different learning strategies define different metaheuristics. The strategy implemented by genetic algorithms (GA) (Holland (1975)) is inspired on the natural evolution of species in biology. In GAs candidate solutions are represented by individuals of a population that undergoes an evolutionary process. Learning is performed by establishing mechanisms that generate diversity while preserving the building blocks of the solution (mutation and recombination). Finally, the selection of the best candidate solutions introduces a directionality in the search.

Other biologically inspired metaheuristics are ant colony and particle swarm optimization. In ant colony optimization (Dorigo and Gambardella (1997)), the source of inspiration is the foraging behavior of ants. Ant colonies can be seen as a collection of simple agents endowed with some communication mechanism (the trace of pheromone

left by an individual ant in a given trajectory) that can solve complex optimization problems through the emergence of effective strategies at the level of the collective. In particle swarm optimization (Eberhart and Kennedy (1995)) the strategy is inspired in the global coordination of large flocks of birds through local interactions among the individuals in the flock. These algorithms consider a collection of particles that explore the search space. The motion of the particle is influenced by both the best local solution it has found and by better solutions found by other particles. In this manner, the swarm is expected to move towards the best solution while maintaining some diversity that prevents it from becoming trapped in a local solution.

As established by the No-Free-Lunch Theorems for optimization (Wolpert and Macready (1997)), metaheuristic approaches alone cannot perform better than random search when averaged over all possible problems. It is therefore necessary to include specific domain knowledge in order to find algorithms that are effective for the problem at hand. To achieve this goal, the concept of *metaheuristic hybridization* can be used. The main idea is to combine a metaheuristic with specialized heuristics or other metaheuristics to design hybrid algorithms that take advantage of the synergies among the individual pieces. Different taxonomies of hybrid metaheuristics can be elaborated: see, for instance Talbi (2002), El-Abd and Kamel (2005), Blum and Roli (2003). Here we summarize the classification given in Raidl (2006) and use it to frame the hybrid approach developed in this thesis to address optimization problems with cardinality constraints. Metaheuristics can be classified according to the following characteristics:

1. Type of algorithms being hybridized: Metaheuristics can be combined with other metaheuristics, with problem specific heuristics or with exact or approximate techniques of the field of mathematical programming (for instance, integer or convex programming).

2. The level of hybridization depends on how much information of the other algorithms is incorporated in the design of the hybrid metaheuristic. Low-level hybridization corresponds to a highly integrated design in which all pieces take advantage of the internal workings of the other algorithms. By contrast, in high-level hybridizations every algorithm retains its identity. The level of abstraction is higher and the design of the the pieces does not require to know in detail how the other algorithms work.

3. Execution order: The hybridized algorithms can be executed either sequentially or in an interleaved manner. In an interleaved execution, one algorithm executes the others as sub-procedures or they can interact in more complicated ways. If the problem has the appropriate structure, a parallel approach may be used.

4. Control strategy: We can distinguish between integrative or collaborative approaches. In integrative approaches there is a master algorithm that invokes the execution of the other algorithms. An example of integrative approach is the use of *decoders*, where the master algorithm operates on an incomplete representation of the solutions and obtains the complete solution by executing a subordinate algorithm. For instance, the subordinate algorithm can be an optimization technique

that addresses a subproblem resulting from the decomposition of the original problem. In purely collaborative approaches there is no hierarchy in the execution of the algorithms. Collaboration is implemented through the exchange of information between the algorithms. For instance, genetic algorithms with populations evolved in parallel that occasionally exchange individuals are examples of implementation of a collaborative strategy (Shapcott (1992)).

Memetic algorithms (MA) represent an important example of metaheuristic hybridization (Moscato and Cotta (2003)). These algorithms traditionally make use of population metaheuristics, such as genetic algorithms, but instead of applying them as black-box optimization techniques, they seek to combine them with problem knowledge in a meaningful way. Usually, this results in using a mathematical programming technique to improve or decode a candidate solution in the population metaheuristic. The denomination 'memetic' corresponds to consider *memes* instead of genes as the building blocks of the evolutionary process. A meme, as originally introduced in Dawkins (1976), can be understood as a cultural analogue of a gene. It can represent an idea, a word or a concept which, in the words of Dawkins, leaps "from brain to brain via a process which, in the broad sense, can be called imitation". Memetic algorithms have found a wide area of application in the field of NP-Hard (Cormen et al. (1990)) optimization problems, because they can often obtain high quality solutions with a moderate computational effort. Some examples of the application of MAs include multidimensional knapsack problems (Puchinger et al. (2010)), the traveling salesman problem (Krasnogor and Smith (2000)), graph partitioning (Kim et al. (2011)), course timetabling (Qaurooni (2011)), clustering (Speer et al. (2004)), and supply chain network (Yeh (2006)). As we will shortly see, the approach described in this chapter can also be seen as a memetic algorithm.

2.2.2 Hybrid Approach for Cardinality Constraints

For problems with a cardinality constraint, the size of the search space grows exponentially with the maximum size of the feasible subsets. Let $\mathcal{S}$ be the set of feasible solutions. Let N represent the number of elements available and K the upper bound on the cardinality of the subsets ($K \leq N$). The size of the combinatorial search space is

$$|\mathcal{S}| = \sum_{k=1}^{K} \binom{N}{k}. \tag{2.1}$$

For a fixed K, this number grows exponentially with N and is quite large even for moderate values of N and K.

Consider the cardinality constrained problem

$$\begin{aligned} \min_{\mathbf{x}} \quad & f(\mathbf{x}) \\ \text{subject to} \quad & g_j(\mathbf{x}) \leq 0 \; j = 1, \ldots, m \\ & h_k(\mathbf{x}) = 0 \; k = 1, \ldots, p \\ & |\{x_i | x_i \neq 0, \, i = 1, \ldots, N\}| \leq K. \end{aligned} \tag{2.2}$$

We now introduce the vector of indicator variables $\mathbf{z} \in \{0,1\}^N$. In this vector, the i-th component takes the value $z_i = 1$ if $x_i \neq 0$, otherwise $z_i = 0$. Incorporating these new variables, the optimization problem (2.2) can be written as

$$\begin{aligned} \min_{\mathbf{x},\mathbf{z}} \quad & f(\mathbf{x}, \mathbf{z}) \\ \text{subject to} \quad & g_j(\mathbf{x}, \mathbf{z}) \leq 0 \; j = 1, \ldots, m \\ & h_k(\mathbf{x}, \mathbf{z}) = 0 \; k = 1, \ldots, p \\ & \sum_{i=1}^{N} z_i \leq K. \end{aligned} \tag{2.3}$$

An alternative formulation of the problem is

$$\min_{\mathbf{z}} \quad \hat{f}(\mathbf{z}) \tag{2.4}$$

$$\text{subject to} \quad \sum_{i=1}^{N} z_i \leq K \tag{2.5}$$

where, for a given $\mathbf{z}$, $\hat{f}$ is defined as the solution of the subordinate optimization problem

$$\begin{aligned} \hat{f}(\mathbf{z}) = \min_{\mathbf{x}^{[\mathbf{z}]}} \quad & f(\mathbf{x}^{[\mathbf{z}]}) \\ \text{subject to} \quad & g_j(\mathbf{x}^{[\mathbf{z}]}) \leq 0 \; j = 1, \ldots, m \\ & h_k(\mathbf{x}^{[\mathbf{z}]}) = 0 \; k = 1, \ldots, p. \end{aligned} \tag{2.6}$$

The subordinate optimization problem does not have a cardinality constraint. The search now takes place in a subspace defined by the subset of components of $\mathbf{x}$ for which $z_i = 1$. The vector $\mathbf{x}^{[\mathbf{z}]}$ is a k-dimensional vector ($k \leq K$) obtained by removing from $\mathbf{x}$ the components for which $z_i = 0$. Problem (2.4) will be referred to as the *master cardinality constrained problem* (MCCP). We use an optimization metaheuristic to address the combinatorial MCCP, including the cardinality constraint. The metahuristic iteratively generates candidate solutions encoded by the particular values of $\mathbf{z}$. For a fixed $\mathbf{z}$, subproblem (2.6), which will be called the *subordinate continuous problem* (SCP), can be solved by an appropriate heuristic or exact method.

Let us classify this approach according to the criteria presented in subsection 2.2.1:

1. Types of algorithms hybridized: The metaheuristic is used to address the combinatorial optimization problem. It is hybridized with an exact or a heuristic method,

depending on the form of the subordinate optimization problem. If this problem is linear, the simplex algorithm can be used to solve it. If it is quadratic or convex, general interior point methods can be applied.

2. The level of hybridization is relatively high: The evaluation function of the metaheuristic is the solution of the subordinate optimization problem. The algorithm for the subproblem retains its own identity and does not have to be adapted to the metaheuristic.

3. The execution of the algorithms is interleaved: The metaheuristic proceeds by searching in the combinatorial space. For each candidate solution generated by the metaheuristic a new instance of the subordinate problem has to be solved.

4. The control structure is clearly integrative, and represents a particular case of the use of decoders: The metaheuristic operates on an incomplete representation of a candidate solution, which is the particular subset of attributes that defines the search space for the subordinate problem. To fully determine the candidate solution of the original problem one needs to solve the subordinate problem as well.

The hybrid approach proposed has several advantages. First, there is no need to handle the constraints g_j and h_k in the metaheuristic. This keeps the successor operators simple and avoids computationally expensive repair procedures. Second, the metaheuristic is kept general so that it can also be applied to other problems with cardinality constraints. The only adaptation needed is the particular algorithm used to solve the SCP. Third, specialized optimization techniques can be used to solve the subordinate problem. If the particular form of the objective function and the constraints permits it, this subordinate problem can be solved to proven optimality.

In summary, our objective is to design metaheuristic approaches to efficiently search in spaces of subsets of a specified cardinality. In this context, the question arises of how to handle possible violations of the cardinality constraint in the metaheuristic. As will be shown in Section 2.3.2, standard crossover operators in genetic algorithms generate candidate solutions that need not have the same cardinality as their parents. There are different strategies to address this limitation:

(i) Candidate solutions that violate the cardinality constraint can be generated by the algorithm. Whenever a violation occurs, a repair algorithm is applied that transforms the infeasible solution into a solution of the desired cardinality. Typically, a local search is used to obtain the closest feasible solution, but random repair mechanisms can be used as well.

(ii) Solutions that violate the cardinality constraint can be generated by the successor operators. In contrast to the previous approach, infeasible solutions are not repaired. Instead, a penalty term is introduced in the evaluation function so that infeasible candidate solutions have worse scores than feasible ones.

(iii) No candidate solutions that violate the cardinality constraint are generated at any time by the algorithm. Therefore, one needs to design mechanisms that generate successors of the appropriate cardinality.

In the next section we describe several metaheuristics that can be used to address the combinatorial part of the optimization in cardinality constrained problems. In Part II of this thesis the performance of the metaheuristics are compared in several problems of practical interest.

Even with the hybrid scheme proposed in this section the combinatorial complexity of the search could still be too large for the metaheuristic to be effective. To address this difficulty we introduce in Section 2.4 some heuristics that can be used, when applied in a preprocessing step, to reduce the dimensionality of the combinatorial search space. The objective is to eliminate variables from the original problem without reducing the quality of the obtained solutions. To this end one attempts to identify variables that are not likely to be included in the final solution.

2.3 Metaheuristics for optimal subset selection

In this section we describe three types of metaheuristics that are used to address the combinatorial aspect of the cardinality constrained optimization problems investigated in Part II of this thesis. In the following descriptions we assume that an equality cardinality constraint $\sum_{i=1}^{N} z_i = K$ is considered. A problem with inequality cardinality constraint can be approached by solving a series of problems with an equality cardinality constraint $\sum_{i=1}^{N} z_i = k$, $k = 1, \ldots, N$ and taking the best value found.

2.3.1 Simulated annealing

Simulated annealing (SA) is an optimization technique inspired by the field of thermodynamics (Kirkpatrick et al. (1983)). The main idea is to mimic the physical process of melting a solid and then cooling it to allow the formation of a regular crystalline structure that attains a minimum of the system's free energy. Convergence to the global minimum is guaranteed if the cooling (annealing) is sufficiently slow, so that the system remains close to equilibrium at all stages in the process. If the molten solid is cooled too fast, the system reaches a state that does not correspond to the regular lattice characteristic of a crystal, but to the amorphous configuration of a glass, which is a metastable local minimum of the free energy. Random thermal fluctuations provide a source of variability in the exploration of the space of physical configurations. In a fluid, thermal motion allows the molecules to access all the physical space available. At a given temperature, most realized transitions lead to lower energy configurations. However, higher energy states can be accessed with non-zero probability as well. These uphill transitions provide a mechanism for escaping local minima in the energy landscape. As the temperature is lowered, the probability of these fluctuations that bring the system away from its equilibrium configuration is also reduced. In simulated annealing the function to be minimized $F(\mathbf{z})$ (objective or cost function) takes the role of the free energy in

the physical system. The physical configuration space is replaced by the space of candidate solutions, which are connected by transitions defined by a neighborhood operator $\mathcal{N}(\cdot)$. The stochastic search proceeds by considering transitions from the current state $\mathbf{z}^{(cur)}$ to a neighboring configuration $\mathbf{z}_l \in \mathcal{N}(\mathbf{z}^{(cur)})$ generated at random. The proposed transition is accepted with probability 1 if the value of the objective function decreases. Otherwise, if the candidate configuration is of higher cost, the transition is accepted only with a certain probability $p \leq 1$. This probability is expressed with the following function

$$P_{\text{accept}}(\mathbf{z}_l, \mathbf{z}^{(cur)}; T_k) = \exp\left(-\frac{F(\mathbf{z}_l) - F(\mathbf{z}^{(cur)})}{T_k}\right), \tag{2.7}$$

where the parameter T_k plays the role of the overall temperature of the system. The probability of accepting an uphill transition becomes thus smaller as the temperature is lowered. Greedy search, which only accepts cost-lowering transitions, is in fact recovered when the temperature is set to zero. On the other extreme, the infinite temperature limit corresponds to blind stochastic search (all transitions are accepted). At the beginning of the cooling process, the exploration is performed using higher temperatures so that a large amount of the search space can be effectively explored. The search then proceeds by epochs. In the lapse that corresponds to an epoch the temperature is held constant. The length of an epoch should be sufficient to allow the system to approach equilibrium at the corresponding temperature. Between consecutive epochs the temperature is lowered according to the annealing schedule. As the temperature is reduced, the search focuses on regions of the configuration space characterized by lower values of the cost function. If the annealing schedule is sufficiently slow, this procedure guarantees convergence to the global minimum. In practice, faster annealing schedules are used, so that a solution can be found within a reasonable amount of time. An example is the geometric annealing schedule, in which the temperature is updated using the formula $T_k = \gamma T_{k-1}$. The value of γ should be smaller than, but close to 1. Even though this schedule cannot guarantee convergence to the global optimum, the solutions identified are near-optimal in many cases of practical interest, especially if due to the structure of the search space there exist low-cost paths connecting the initial state and the target optimum. A general version of this technique is given as Algorithm 1. In this pseudocode, the function annealingSchedule returns the temperature T_k for the following epoch.

Cardinality constraints can be handled in SA by selecting an appropriate encoding for $\mathbf{z}$ and a corresponding neighborhood $\mathcal{N}(\mathbf{z})$. In particular, the candidate solutions can be encoded as bit-strings of length N, representing a subset of the given cardinality K. The components of the binary vector $\mathbf{z}$ are then interpreted as indicating membership to the set: If $z_i = 1$, the ith element is included in the solution. Otherwise, if $z_i = 0$, it is excluded from the selection. It is also convenient to design a neighborhood operator that preserves the cardinality constraints, in order that no penalty or repair mechanisms are needed. A simple design is to exchange an element included in the current candidate solution with an element excluded from it.

Algorithm 1 Simulated annealing

- Generate an initial configuration $\mathbf{z}^{(\mathbf{0})}$ and set the initial temperature T_0
- $\mathbf{z}^{(cur)} \leftarrow \mathbf{z}^{(0)}$
- $i \leftarrow 0$
- While convergence criteria are not met [Annealing loop]
  - $i \leftarrow i + 1$
  - Set the temperature for epoch i: $T_i = \text{annealingSchedule}(T_{i-1})$
  - Set the length of epoch i: L_i
  - For $l = 1, \ldots, L_i$ [Epoch loop]
    1. Select an element from the neighborhood of the current search state $\mathbf{z}_l \in \mathcal{N}(\mathbf{z}^{(cur)})$ at random.
    2. If $F(\mathbf{z}_l) < F(\mathbf{z}^{(cur)})$, then $\mathbf{z}^{(cur)} \leftarrow \mathbf{z}_l$ with probability 1.
    3. Else, $\mathbf{z}^{(cur)} \leftarrow \mathbf{z}_l$ with probability $P_{\text{accept}}(\mathbf{z}_l, \mathbf{z}^{(cur)}; T_i)$.
- Return the best value found.

2.3.2 Genetic Algorithms

Genetic algorithms are a class of optimization methods that imitate the process of the natural evolution of a population of a species (Goldberg (1989); Holland (1975)). Optimization is achieved by selection from a population that exhibits some random variability. The evolved population is made up of individuals characterized by a chromosome (the genotype) that encodes a problem-specific procedure to generate candidate solutions. The objective is to maximize a *fitness* function $\Phi(I)$ that measures the quality of the individual I in terms of the actual candidate solution generated (the phenotype). Each phenotype is composed of *genes*. The possible values a gene is allowed to have are called *alleles*.

The outline of a general genetic algorithm is shown in Algorithm 2. The algorithm starts with an initial population of individuals that undergoes evolution in time steps called generations. In each generation, a subset of individuals is selected from the existing population for reproduction according to their fitness. From these selected individuals a new population is generated, using mutation and crossover operations. In the crossover operation one or more individuals are engendered from a given set of parents (usually two) by exchanging parts of the parents' genetic material. These are then subject to mutation with a specified probability. The crossover operation generally functions as a guide for the search process, introducing into the population individuals that combine advantageous features of their ancestors. Mutation mostly acts as a mechanism of escaping local optima, so that unexplored regions of the search space can be accessed. Once crossover and mutation have been applied, the population is then renewed according to a generational substitution scheme that specifies how the new population is selected from the pool of old and newly generated individuals. There are

different options for composing the new population for the next time step: A *steady state scheme* replaces only the individual with the lowest fitness of the previous generation with a new individual. On the opposite extreme, in *generational schemes* the whole population is composed of new individuals. This scheme has the disadvantage of discarding solutions that could be optimal or near-optimal. In this case, it may be useful to resort to *elitism* and retain a few high-fitness individuals from generation to generation. This procedure ensures that the best solutions found so far are not lost. However, abuse of this mechanism can cause premature convergence to local optima. In each generation of the evolution process, the population is enriched with individuals that have higher fitness values than their ancestors. Therefore the evolution process is expected to eventually reach an optimum, possibly a local one.

Algorithm 2 Genetic Algorithm

- Generate an initial population $\mathcal{P}_0$ with P individuals.
- For each individual $I_j \in \mathcal{P}_0$, calculate its fitness $\Phi(I_j)$.
- Initialize the generation counter $t \leftarrow 0$.
- While convergence criteria are not met:
  - Increase the generation counter $t \leftarrow t + 1$.
  - Select a parent set $\Pi_t \subset \mathcal{P}_t$ composed of n_P individuals from the population.
  - While $\Pi_t \neq \emptyset$:
    - Extract two individuals I_1 and I_2 from Π_t.
    - Apply the crossover operator $\Theta(I_1, I_2)$ and generate n_C children (with probability p_C).
    - Apply the mutation operator to the n_C children (with probability p_M).
  - Calculate the fitness value of the new individuals.
  - Add the new individuals to the population.
  - Select P individuals that make up $\mathcal{P}_{t+1}$, the population for generation $t + 1$.

For problems with cardinality constraints, two alternative encodings for the candidate solutions are considered. One possibility is the standard binary representation, in which the chromosomes are encoded by bit-strings. The different positions in the chromosome correspond to binary variables. This scheme is a natural representation for many combinatorial search problems. Standard genetic operators employed in combination with this representation are uniform crossover, N-point crossover and bitwise flip mutation. Uniform crossover consists in randomly choosing, for each position of the chromosome, a parent (I_1 or I_2), and setting in the corresponding gene of the offspring the allele from the parent selected. In N-point crossover, the child's chromosome is the result of the recombination of the genetic material of the parents. The segments exchanged are defined by N randomly selected positions in the parent's chromosomes. The bitwise flip mutation operator selects a gene position at random and changes the allele to its complementary. It is easy to show that these prescriptions for mutation

and crossover do not preserve the cardinality of the candidate solutions. Consider, for instance, a problem with $D = 10$ variables and a cardinality constraint $K = 5$. From parents (0100110011) and (0111100010), with cardinality 5, 1-point crossover with crossover point 4 generates the offspring (0100100010) (0111110011), which do not fulfill the cardinality constraint.

The difficulty with this encoding is that standard mutation and crossover operators do not preserve the number of non-zero bits of the parents. Therefore, penalty terms in the fitness function or ad-hoc repair mechanisms are needed to enforce the cardinality constraints. One possible solution to this problem is to assign lower fitness values to individuals in the population that violate the cardinality constraint. Assuming that a problem with an equality cardinality constraint is considered, a penalized fitness function can be built by subtracting from the standard fitness function a penalty term that depends on the magnitude of the violation of the cardinality constraint of the individual with binary chromosome $\mathbf{z}$

$$\Delta_k(\mathbf{z}) = |\mathrm{Card}(\mathbf{z}) - K| \,. \tag{2.8}$$

The penalized fitness function is then

$$\Phi_p(\mathbf{z}) = \Phi(\mathbf{z}) - \beta f_p(\Delta_k(\mathbf{z})), \tag{2.9}$$

where $f_p : \mathbb{N} \to \mathbb{R}^+$ is a monotonically increasing function of $\Delta_k(\mathbf{z})$ and $\beta \geq 0$ represents the strength of the penalty.

Another option is to repair infeasible individuals as soon as they are generated. Several repair mechanisms can be defined for this purpose. For instance, an individual can be repaired by randomly setting some bits to 0 or 1, as needed, until the cardinality constraint is satisfied (*random repair*). Another alternative is to use a heuristic to determine which bits must be set to 0 or to 1 (*heuristic repair*). The results of a greedy optimization or the solutions of a relaxed version of the problem can also be used to achieve this objective (Moral-Escudero et al. (2006)).

In summary, cardinality constraints are difficult to incorporate into GAs with a binary representation using standard crossover and mutation operators. A possible solution is to penalize infeasible individuals, assigning them lower fitness values. However, the functional form of the penalty and the parameters need to be determined through careful experimentation. Chromosome repair offers a more elegant, and possibly better alternative. Nevertheless, the repair mechanism may introduce biases in the search and can be costly. All these factors can mislead the search and cause premature convergence to a local optimum in the genetic algorithm.

Chapter 3 of this thesis is devoted to the design of genetic representations and operators that generate individuals of the same cardinality as their parents. For this purpose, the algorithm encodes the candidate solutions as a subset of the specified cardinality. This representation is more appropriate to design cardinality-preserving crossover and mutation operators. In this manner, penalty functions and repair mechanisms are

avoided. It is expected that limiting the search to the space of feasible candidate solutions will lead to the design of algorithms that are more efficient and effective.

2.3.3 Estimation of Distribution Algorithms

Estimation of distribution algorithms (EDAs) are a class of evolutionary methods in which diversity is generated by a probabilistic sampling scheme (Larrañaga and Lozano (2002)). At each time step in the evolutionary process, a probability distribution is used to characterize the genotype of the generation in a statistical manner. A population of individuals is obtained by generating random samples from this multidimensional distribution. Each sample (chromosome) encodes a candidate solution. Selection involves generating a subsample in which individuals with larger fitness values are more likely to be present. This subset is subsequently used to estimate a new probability distribution that characterizes the population in the next generation. Finally, new individuals are sampled from the distribution and another generation begins. This process is repeated until the specified convergence criteria are met. The pseudocode for a generic EDA is detailed in Algorithm 3.

Algorithm 3 Estimation of distribution algorithm (EDA)

- Initialize the distribution that characterizes the population $P^{(0)}(\mathbf{z})$
- Initialize the generation counter $g \leftarrow 0$.
- While convergence criteria are not met
  - Sample a population of P individuals using $P^{(g)}(\mathbf{z})$

    $$D_g = \{\mathbf{z}^{(g1)}, \ldots, \mathbf{z}^{(gP)}\}$$

  - Sort the population by non-increasing fitness values

    $$D'_g = \{\mathbf{z}^{(gi_1))}, \mathbf{z}^{(gi_2)}, \ldots, \mathbf{z}^{(gi_P)}\},$$

    where $i_1, i_2, \ldots, i_P$ is a reordering of the indices $1, 2, \ldots, P$ such that

    $$\Phi(\mathbf{z}^{(gi_1)}) \geq \Phi(\mathbf{z}^{(gi_2)}) \geq \cdots \geq \Phi(\mathbf{z}^{(gi_P)})$$

  - Select the first $M \leq P$ individuals from the sorted population

    $$D_g^{Se} = \{\mathbf{z}^{(gi_1)}, \mathbf{z}^{(gi_2)}, \ldots, \mathbf{z}^{(gi_M)}\}$$

  - Estimate the new probability distribution $P^{(g+1)}(\mathbf{z})$ using D_g^{Se}
  - Update the generation counter $g \leftarrow g + 1$
- Return the best solution found.

Different classes of EDA are characterized by the sampling scheme used in the selection step. Population Based Incremental Learning (PBIL) is a representative algorithm

of the EDA family (Baluja (1994)). It operates on binary chromosomes of fixed length and assumes statistical independence among the genes $\{z_i; i = 1, 2, \ldots, N\}$. In generation g, the genotype of the population is characterized by the probability vector $\mathbf{p}^{(g)}$, whose i-th component is the probability of assigning the value 1 to the gene in the i-th position. The update of the probability distribution using D_g^{Se} takes in PBIL the following form

$$\mathbf{p}^{(g+1)} = \alpha \frac{1}{M} \sum_{m=1}^{M} \mathbf{z}^{(gi_m)} + (1 - \alpha)\mathbf{p}^{(g)}, \tag{2.10}$$

where $\mathbf{z}^{(gi_m)}$ represents the individual in the i_m-th position in generation g (where individuals in the population are sorted by non-increasing fitness values), and $\alpha \in (0, 1]$ is a smoothing parameter included to avoid strong fluctuations in the estimates of the probability distribution. The Univariate Marginal Distribution Algorithm (UMDA, Muehlenbein (1998)) is a special case of PBIL when $\alpha = 1$.

There is also a continuous version of PBIL (PBIL$_c$, Sebag and Ducoulombier (1998)) that operates on real-valued choromosomes $\boldsymbol{\xi}$, which eventually need to be translated into integer-valued indicator variables $\mathbf{z}$. In this algorithm the joint distribution of $\boldsymbol{\xi}$ is assumed to be a multidimensional Gaussian with a diagonal covariance matrix

$$p(\boldsymbol{\xi}) = \prod_{i=1}^{N} \frac{1}{\sqrt{2\pi}\sigma_i} \exp\left(-\frac{(\xi_i - \mu_i)^2}{2\sigma_i^2}\right). \tag{2.11}$$

The update rule from generation g to $g + 1$ is similar to the discrete version, but includes terms that bring the population mean $\boldsymbol{\mu}^{(g+1)}$ closer to the best two individuals and away from the worst one

$$\boldsymbol{\mu}^{g+1} = (1 - \alpha)\boldsymbol{\mu}^{g} + \alpha\left(\xi^{(gi_1)} + \xi^{(gi_2)} - \xi^{(gi_P)}\right). \tag{2.12}$$

Both the discrete and the continuous versions of PBIL and UMDA assume independence among the individual genes in the chromosome (i.t., the joint probability distribution of the genes factorizes as the product of univariate marginals). In the EDA family one can also design algorithms in which more complex statistical models are assumed. In particular, the EMNA (Estimation of Multivariate Normal Algorithm, Larrañaga et al. (2001)) assumes real-valued chromosomes $\boldsymbol{\xi}$ that are jointly distributed as a multivariate Gaussian $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$, with an arbitrary covariance matrix $\boldsymbol{\Sigma}$

$$p(\boldsymbol{\xi}) = \frac{1}{\sqrt{2\pi}|\boldsymbol{\Sigma}|^{1/2}} \exp\left(-\frac{1}{2}(\boldsymbol{\xi} - \boldsymbol{\mu})^T \cdot \boldsymbol{\Sigma}^{-1} \cdot (\boldsymbol{\xi} - \boldsymbol{\mu})\right). \tag{2.13}$$

In generation $g + 1$, the vector of means $\boldsymbol{\mu}^{(g+1)}$ and the covariance matrix $\boldsymbol{\Sigma}^{(g+1)}$ are estimated by maximum likelihood

$$\mu_i^{(g+1)} = \frac{1}{M} \sum_{m=1}^{M} \xi_i^{(gi_m)} \tag{2.14}$$

$$\sigma_{ij}^{(g+1)} = \frac{1}{M-1} \sum_{m=1}^{M} (\xi_i^{(gi_m)} - \mu_i^{(g+1)})(\xi_j^{(gi_m)} - \mu_j^{(g+1)}). \tag{2.15}$$

Only the M individuals of the previous generation with the largest fitness are used in the estimation.

Since PBIL$_c$ and EMNA operate on a continuous representation for $\boldsymbol{\xi}$, a mechanism for transforming the continuous variables into discrete indices is necessary. The following is a straightforward procedure that can be used to derive an indicator vector $\mathbf{z}$ with only K non-zero components from a real-valued one $\boldsymbol{\xi}$: First, the N components of the chromosome are sorted in non-increasing order

$$\xi_{j_1} \geq \xi_{j_2} \geq \cdots \geq \xi_{j_N} \tag{2.16}$$

The binary vector $\mathbf{z} \equiv \mathbf{0}$ is initialized to zero. Then, for each $i = 1, \ldots, K$, we set $z_{j_i} = 1$.

When the encoding is binary the cardinality constraints can be enforced in the sampling procedure. Algorithm 4 describes a sampling method that generates individuals of a specified cardinality K from a distribution of bits characterized by the probability vector $\mathbf{p}$. The application of this method to sample new individuals guarantees that the algorithm preserves the cardinality constraint.

Algorithm 4 Sampling individuals of a specified cardinality from $\mathbf{p}$.

- Initialize $\hat{\mathbf{p}} \leftarrow \mathbf{p}$
- Initialize individual $\mathbf{z} = \mathbf{0}$.
- For $i = 1, 2, \ldots, K$
  - Generate a random number $u \sim U[0, 1]$
  - Determine the value of j such that $\sum_{i=1}^{j-1} \hat{p}_i < u \leq \sum_{i=j+1}^{N} \hat{p}_i$.
  - Set $\mathbf{z}_j = 1$.
  - Update the value $\hat{p}_j \leftarrow 0$
  - Renormalize
    $$\hat{p}_i \leftarrow \frac{\hat{p}_i}{\sum_{k=1}^{N} \hat{p}_k}, \; i = 1, 2, \ldots, N,$$
    so that $\hat{\mathbf{p}}$ can be interpreted as a probability vector $\sum_{i=1}^{N} \hat{p}_i = 1$.
  - Return the generated individual $\mathbf{z}$.

Numerous studies have shown that EDAs can be competitive and even outperform GAs in many different domains, especially in optimization tasks in which the dependencies among variables are complex or unknown (Larrañaga and Lozano (2002), Baluja (1994), Baluja and Caruana (1995)). We therefore expect EDAs to be a useful tool in the solution of combinatorial optimization problems with cardinality constraints.

2.4 Dimensionality Reduction for Cardinality Constrained Optimization Problems

In this section we present three pruning techniques that can be applied to reduce the dimensionality of the search space in cardinality constrained optimization problems. These heuristics attempt to identify variables that are not likely to be included in the optimal solution. Because of the cardinality constraints, eliminating variables that do not appear in the optimal solution has no effect on the quality of the solution. Therefore, by designing appropriate heuristics we expect to improve the efficiency of the search without a deterioration of the quality of the final solution. The performance improvements for the metaheuristics used to address the combinatorial optimization problem can be large. For instance, the efficiency of some types of EDAs tends to rapidly deteriorate with the dimensionality of the search space (Ruiz-Torrubiano and Suarez (2010)). The reason is that the probability estimations become more difficult in high dimensional spaces. The benefits of reducing the number of variables that need to be considered are also significant for GAs and SA.

In this section we introduce some heuristics that can be used to guide the dimensionality reduction process. The heuristics described assume that it is possible to efficiently solve the original problem in the absence of cardinality constraints. In this relaxed problem, it is important to also remove the lower bound constraints because otherwise the problem may not be feasible. For instance, consider a problem where at most 10 variables are allowed to take a value different from zero. Assume that each variable must lie within the interval $[0, 1]$ and add up to 1. Assume also that there is a lower bound constraint of $1/10$ for each variable. The problem with cardinality constraint is clearly feasible, but the relaxed problem, without the constraint, is not.

2.4.1 Block pruning

In block pruning a relaxation of the original problem without cardinality and lower bound constraints is solved. A new problem is defined by eliminating from the original problem those variables whose absolute value in the solution of the relaxed problem is lower than a specified threshold $\epsilon > 0$. This threshold is determined in such a way that the number of remaining variables is small and the solution in the reduced space is similar to the solution of the original problem. If the values of the variables included in the solution are required to be above a lower bound l, a reasonable choice for the threshold that determines whether a variable is eliminated or not is $\epsilon = l/2$. The pseudocode is given in Algorithm 5.

2.4.2 Greedy backward selection

In this pruning procedure one variable is discarded at each iteration. To choose the variable that is discarded, one solves a relaxation of the current problem in which cardinality and lower bounds constraints are eliminated. A new optimization problem is then defined by removing the variable with the smallest absolute value in the optimal solution

Algorithm 5 Block pruning

- Let P be an optimization problem on $\mathbf{x} \in \mathbb{R}^N$ with cardinality constraints, such that the problem P^* without the cardinality constraints can be solved efficiently.
- Let P^{**} be the problem P^* without lower bound constraints.
- Use EfficientOptimization(P^{**}) to obtain the optimal solution $\mathbf{x}^{**}$
- For each x_i^{**}, if $|x_i^{**}| < \epsilon$, discard the variable i from P.
- Return the new problem P.

of the relaxed problem. The process is repeated until $K + T$ variables remain, where K is the value of the upper bound in the cardinality constraint and T is a parameter to allow for some slack in the solution. The pseudocode of the greedy backward selection method is given in Algorithm 6.

Algorithm 6 Greedy backward selection

- Let P be an optimization problem on $\mathbf{x} \in \mathbb{R}^N$ with cardinality constraints, such that the problem P^* without the cardinality constraints can be efficiently solved.
- Let P^{**} be the problem P^* without lower bound constraints.
- While P has more than $K + T$ variables:
  - Use EfficientOptimization(P^{**}) to obtain the optimal solution $\mathbf{x}^{**}$
  - Discard the variable i such that $i = \arg\min_i |x_i^{**}|$.
- Return the new problem P.

Greedy backward selection requires more computational effort than block pruning, since $N - (K + T)$ relaxed optimizations need to be performed.

2.4.3 Greedy forward selection

This heuristic proceeds by adding in each iteration the variable that improves the value of the objective function the most. We take as a starting point a problem P_0 with only two variables. These two variables are the optimal solution for $K = 2$, which is computed by exhaustive search. Let the optimal solution to P_0 be $\mathbf{x}_0$ and the corresponding value of the objective function $f(\mathbf{x}_0)$. One then optimizes $N - 2$ relaxed versions of the problem P_{k+}^{**} without cardinality and lower bound constraints with the new variable k, where k is not in P_0. Let $\mathbf{x}_{k+}^{**}$ be the optimal solution of P_{k+}^{**} including variable k. Then the variable k^* is chosen, such that $|f(\mathbf{x}_0) - f(\mathbf{x}_{k^*+}^{**})|$ is maximal. The pseudocode is given in Algorithm 7.

This pruning method is computationally more expensive than block pruning and than greedy backward selection because it requires solving $N(N-1)/2 + \sum_{i=2}^{K+T+1}(N-i)$ auxiliary optimization problems.

Algorithm 7 Greedy forward selection

- Let P_0 be the optimization problem with two variables i, j such that $\{i, j\}$ would be the optimal solution for $K = 2$.
- Let $\mathbf{x}_0$ be the optimal solution to P_0 and $f(\mathbf{x}_0)$ its objective value.
- Initialize $P \leftarrow P_0$.
- While P has fewer than $K + T$ variables:
  - For each k not yet in P, run EfficientOptimization(P^{**}_{k+}) to obtain the optimal solution $\mathbf{x}^{**}_{k+}$
  - Include the variable k^* such that $k^* = \arg\max_k |f(\mathbf{x}_0) - f(\mathbf{x}^{**}_{k+})|$.
- Return the new problem P.

2.5 Summary and Discussion

In this chapter, the importance of designing efficient and effective methods for the solution of cardinality constrained optimization problems has been illustrated with examples of problems of practical interest in various areas of application. In this thesis we propose to combine metaheuristics that are well suited to the solution of the combinatorial part of the optimization task with specialized optimization algorithms that incorporate specific knowledge of the problem considered. A general description of the metaheuristics used in the thesis has been presented. Additionally, various pruning heuristics, which effectively reduce the size of the search space, have been proposed as a preprocessing step. These heuristics attempt to identify and eliminate variables that are not likely to be included in the optimal solution of the cardinality constrained problem.

Cardinality contraints can be incorporated in a genetic algorithm with 0-1 encoding and standard crossover mechanisms by introducing chromosome repair procedures or including penalty terms in the evaluation of the fitness of candidate solutions that do not fulfill the constraint. However, as will be illustrated in Part II of this thesis, both repair and penalization mechanisms tend to introduce a bias and mislead the search process. Therefore, we will emphasize the need for designing algorithms that generate only feasible individuals. In the particular case of genetic algorithms, the framework of forma theory, which will be discussed in detail in Chapter 3, provides a principled way of designing appropriate genetic representations and operators. This leads to a formulation of genetic algorithms that uses a set representation, in which each candidate solution is encoded as a subset of the specified cardinality.

CHAPTER 3

DESIGN OF GENETIC REPRESENTATIONS AND OPERATORS

The choice of chromosome encoding is one of the key elements in the design of effective genetic algorithms. This representation must be chosen to take advantage of the particular structure of the problem at hand. The framework of forma theory introduced in Radcliffe (1994) can be used in the design of genetic representations that incorporate specific domain knowledge. Formae are a generalization of schemata based on the analysis of equivalence classes induced by equivalence relations over the space of chromosomes. The theory of formae introduces some important properties that genetic operators should have. These properties are formulated in terms of the effect these operators have on the formae. The goal is to provide mechanisms to generate diversity that properly balance exploration and exploitation. This theory can be used as a guidance to design encodings for specific problems, in which operators with these desirable properties can be defined and implemented in a simple and efficient manner. The introduction of specific problem knowledge in the design of the algorithm is necessary in order to perform better than random search, as expressed by the no-free-lunch theorems for optimization (Wolpert and Macready (1997)). In this chapter we analyze different encodings that can be used to represent candidate solutions in optimization problems with cardinality constraints. In particular, to address these types of problems, we introduce a set-based encoding with appropriately defined mutation and recombination operators (Random Assortment Recombination (RAR), and Transmitting Random Assortment Recombination (TransRAR)) designed according to the principles of forma theory.

3.1 Introduction

The No-Free-Lunch (NFL) Theorems for optimization state that general optimization techniques are equivalent in terms of their performance, when averaged over all problem classes (Wolpert and Macready (1997)). In other words, above-average performance in

a given class of problems is compensated by below-average performance in other problem classes. Following Wolpert and Macready (1997), we consider only combinatorial optimization problems in which both the search space, $\mathcal{Z}$ (size $|\mathcal{Z}|$), and the space of possible values of the objective function, $\mathcal{Y}$ (size $|\mathcal{Y}|$), are finite. The derivations can be extended to spaces that are infinite and also to continuous ones. An optimization problem is represented by the objective function $f : \mathcal{Z} \to \mathcal{Y}$ where $\mathcal{F} = \mathcal{Y}^{\mathcal{Z}}$ is the space of all possible problems. If an algorithm A performs better than random search on a given problem, characterized by the objective function $f_i \in \mathcal{F}$, then there is another problem, characterized by the cost function $f_j \in \mathcal{F}$, in which A performs worse than random search. This is formally expressed by the following NFL theorem: For any pair of algorithms A_1 and A_2

$$\sum_{f \in \mathcal{F}} P(d_m^y | m, A_1, f) = \sum_{f \in \mathcal{F}} P(d_m^y | m, A_2, f) \tag{3.1}$$

where d_m^y is the sequence of size m of successive values of the objective function produced by the optimization algorithm, which is assumed to be stochastic. Note that any performance measure for a particular sequence d_m^y can be expressed as a function of the values of the objective function in the sequence $\Phi(d_m^y)$. One of the implications of (3.1) is that, if the prior distribution $P(f)$ over all possible objective functions is uniform (which should be the case if no problem knowledge is incorporated in the designed algorithm), then the average performance of any such algorithm (independently of the particular measure used) is equal. This can be seen by defining the average performance for algorithm A as

$$\hat{\Phi}(A) = \sum_{d_m^y} \Phi(d_m^y) P(d_m^y | m, A) = \sum_{d_m^y} \Phi(d_m^y) \sum_f P(d_m^y | A, m, f) P(f). \tag{3.2}$$

Let $|\mathcal{F}|$ be the size of the space of all possible objective functions. If we assume a uniform prior $P(f) = 1/|\mathcal{F}|$,

$$\hat{\Phi}(A_1) = \frac{1}{|\mathcal{F}|} \sum_{d_m^y} \Phi(d_m^y) \sum_f P(d_m^y | m, A_1, f), \tag{3.3}$$

$$\hat{\Phi}(A_2) = \frac{1}{|\mathcal{F}|} \sum_{d_m^y} \Phi(d_m^y) \sum_f P(d_m^y | m, A_2, f), \tag{3.4}$$

for any A_1, A_2.

From Eq. (3.1), it follows that $\hat{\Phi}(A_1) = \hat{\Phi}(A_2)$ for any performance measure Φ.

Introducing problem knowledge in a given algorithm A can therefore be seen as assuming a non-uniform prior $P(f)$ (i.e. some objective functions are weighted differently than others). The quantity of interest is in this case

$$\sum_f P(d_m^y | m, A, f) P(f). \tag{3.5}$$

Assuming a non-uniform prior for the objective function means that it is possible to find algorithms that are consistently better than others in a restricted class of problems. In conclusion, to improve over random search it is necessary to incorporate in the design of the optimization algorithm specific knowledge of the class of problems that we want to solve.

Forma theory is a framework that provides guidance on how to introduce this problem-specific knowledge in the design of representations and operators for genetic algorithms. Formae are a generalization of schemata, in which one attempts to capture the equivalence relations in the search space that are implied by the objective function and the constraints. In particular, a good representation should have minimal redundancy. The chromosomes that are instances of the same formae should have similar performance. The genetic operators should be designed in such a way that the disruption of these units is unlikely and that improvements are easy to obtain by recombination.

In this chapter we apply these design principles to problems with cardinality constraints. A set representation for the chromosomes is well-suited for these types of problems. Different types of genetic operators that can be used to solve these problems reliably and efficiently are investigated. The chapter is organized as follows: Section 3.2 introduces the general framework of forma theory presented in Radcliffe (1994). Crossover operators defined within this framework are introduced in Section 3.3. These operators are first defined in a general way. Then the concrete implementation for a set representation is given in 3.4. Finally, Section 3.5 summarizes the conclusions of this chapter.

3.2 Forma Theory

Genetic algorithms are general optimization methods in which a population of individuals (chromosomes), of which everyone represents a candidate solution to the optimization problem, evolves through the mechanisms of diversity generation and selection (Holland (1975)). A possible representation for these candidate solutions is a k-ary string of a fixed size n. For instance, let $k = 3$, $n = 5$ and the alphabet $\Delta = \{0, 1, 2\}$. In this case, valid chromosomes would be: 01200, 12101, 11120, ... Consider the extended alphabet $\Delta^* = \Delta \cup \{\square\}$, where $\square$ stands for any of the symbols in the original alphabet. We define a *schema* as a string of length n in the extended alphabet $\{0, 1, 2, \square\}$. Schemata represent groups of chromosomes that have a common structure. For instance, the schema $1\square\square\square\square$ represents all chromosomes that have the value 1 in the first position. Similarly, the schema $\square 2\square\square 0$ represents all chromosomes that have allele 2 in the second position and 0 in the last position.

Assuming that the likelihood to be selected in the next generation is proportional to the fitness of the individual it is possible to establish a lower bound on the expected number of chromosomes that correspond to a given schema in the population. Let $X_\xi(t)$ be the number of individuals that belong to the schema ξ in generation t. Let $\phi_\xi(t)$ be the average fitness of all chromosomes which are members of ξ in the population at time t. Finally, let $\phi(t)$ be the average fitness of all the individuals in the population at time

t. With these definitions the *schema theorem* can be stated as follows:

$$\mathbb{E}\left[X_\xi(t+1)\right] \geq X_\xi(t)\frac{\phi_\xi(t)}{\phi(t)}\left(1-\sum_{\omega\in\Omega} p_\omega p_\omega^\xi\right), \tag{3.6}$$

where $\mathbb{E}[\cdot]$ denotes the expected value over the randomized operations in the algorithm, p_ω represents the probability of applying the crossover operator $\omega \in \Omega$ and p_ω^ξ is the probability that the crossover operator ω disrupts schema ξ.

Implicitly, the schema theorem assumes that a given schema ξ represents a set of candidate solutions with a similar fitness. Otherwise, the quantity $\phi_\xi(t)$ would no longer be an useful measure of the average performance of schema ξ. Therefore, the assumption is that the variance of the fitness values for the individuals in the population that correspond to schema ξ is sufficiently low (Radcliffe and Surry (1995)). In contrast if this variance is high (i.e. the chromosomes that are instances of ξ can have both low and high values of the fitness function), the predictions of the schema theorem are no longer useful. Therefore, it is necessary to use higher level entities, other than schemata, to describe the evolution of the population.

Forma theory is a generalization of schema theory, in which *formae* instead of schemata are used. A forma is an equivalence class that results from the definition of genes as equivalence relations between candidate solutions for the problem at hand. Therefore, we can identify a forma with an allele. Consider, for instance, the equivalence relation "x is related to y if and only if x was born in the same country as y". This equivalence relation splits the search space into disjoint partitions, each one representing a country. Therefore, the equivalence relation can be seen as a gene. Correspondingly, the equivalence classes represent the different alleles of the gene. In general, a set of N equivalence relations $\{\psi_1, \ldots, \psi_N\}$ must be constructed in order to properly define a genetic representation. Each equivalence relation defines a representative gene for the problem at hand. Therefore, a meaningful set of equivalence relations gives rise to a meaningful genetic representation. This is the key observation of forma theory.

In order to define what is "meaningful" in this context, some auxiliary concepts need to be introduced. The notation and definitions are adapted from those in Radcliffe (1994). While remaining faithful to the original, we occasionally introduce some modifications with respect to Radcliffe (1994) for the sake of clarity. If the changes are significant for a given lemma, a proof of the lemma with the modified definitions is given in the Appendix.

Let $\mathcal{S}$ represent the space of all candidate solutions for a given problem, and let $\mathbb{E}(\mathcal{S})$ denote the set of all possible equivalence relations ψ that can be defined over $\mathcal{S}$. Given an equivalence relation ψ, Ξ_ψ (or simply Ξ if there is no ambiguity) is defined as the set of all equivalence classes induced by ψ. An equivalence relation is a reflexive, symmetric and transitive relation that defines a partition of the original set

$$\mathcal{S} = \bigcup_{\xi_i\in\Xi} \xi_i \quad \text{and} \quad \xi_i \cap \xi_j = 0 \ \text{ for all } \xi_i, \xi_j \in \Xi \ , \ i \neq j. \tag{3.7}$$

In other words, ψ partitions the search space into disjoint equivalence classes ξ_i, which we identify with formae.

The basic operation for combining equivalence relations is defined as the intersection or logical conjunction. Let ψ and ϕ be two equivalence relations. Their intersection $\psi \cap \phi$ has value true if and only if both ψ and ϕ have value true on any given argument pair

$$x(\psi \cap \phi)y \text{ if and only if } x\psi y \wedge x\phi y \quad \forall x, y \in \mathcal{S}. \tag{3.8}$$

The equivalence classes generated by the intersection $\psi \cap \phi$ are the intersections of those generated by ψ and ϕ.

A meaningful genetic representation is characterized in this framework by the existence of an *orthogonal basis* of equivalence relations in $\mathcal{S}$. The following definitions are useful in understanding what is meant by this property:

Definition 3.1. Let Ψ be a set of equivalence relations over $\mathcal{S}$. We say that Ψ *covers* $\mathcal{S}$ if and only if for every candidate solution $x \in \mathcal{S}$ there is at least one $y \neq x$ such that x and y are not equivalent under a relation in Ψ; that is, $\forall x \in \mathcal{S}, \exists \psi \in \Psi$, $y \in \mathcal{S}$ and $x \neq y$ such that $x\psi y$ is false.

Coverage is a first condition for a set of equivalence relations Ψ to be meaningful. This notion reflects the fact that it is possible to discriminate among the candidate solutions in $\mathcal{S}$ using the equivalence relations in Ψ.

Definition 3.2. A set Ψ of equivalence relations is said to *span* another set Φ if and only if every equivalence relation $\phi \in \Phi$ can be constructed by intersections of the equivalence relations in Ψ; that is, $\forall \phi \in \Phi$, $\exists \{\psi_1, \ldots, \psi_m\} \subseteq \Psi$ such that $\cap_{i=1}^{m} \psi_i = \phi$.

Definition 3.3. A set Ψ of equivalence relations is said to be *independent* if and only if no equivalence relation $\psi \in \Psi$ can be constructed by intersections of other equivalence relations in Ψ.

Definition 3.4. A set Ψ of equivalence relations is said to be *orthogonal* if and only if, given any subset of equivalence classes generated by different equivalence relations in Ψ, their intersection is non-empty, i.e. given any subset $\{\xi_1, \ldots, \xi_k\}$ where ξ_i is an equivalence class of relation $\psi_i \in \Psi$, it holds that $\cap_{i=1}^{k} \xi_i \neq \emptyset$.

Orthogonality implies that we can combine alleles in different genes independently. That is, given any two genes, every combination of alleles always represents a feasible candidate solution. Therefore, the assignment of a given allele to a gene is independent of any other assignment in another gene.

Definition 3.5. A set Ψ of equivalence relations is said to form a *basis* for $\mathcal{S}$ if and only if Ψ is independent and Ψ spans $\mathbb{E}(\mathcal{S})$.

An orthogonal basis of equivalence relations that covers $\mathcal{S}$ induces a meaningful genetic representation. Moreover, this basis is unique, as shown in Radcliffe (1994). We identify each equivalence relation in the basis with a *gene*, and each equivalence class

induced by an equivalence relation with an *allele*. An equivalence relation in the basis will be called a *basic* equivalence relation and each equivalence class induced by a basic equivalence relation will be called a *basic* forma.

In summary, the design of a meaningful genetic representation for a given problem consists in defining a set of genes (equivalence relations) $\Psi = \{\psi_1, \ldots, \psi_N\}$ with the following properties:

1. **Independence**: No gene in Ψ can be constructed from other genes in the set.
2. **Span**: Every possible equivalence relation can be obtained by combinations of genes in Ψ.
3. **Orthogonality**: Every candidate solution represented by any combination of alleles exists and is meaningful.
4. **Coverage**: For every two distinct candidate solutions $x \neq y$, there is at least one allele which is different in the genetic representation of x and y.

As noted in Radcliffe (1990), the schema theorem (3.6) also applies if ξ represents a general equivalence class (forma) instead of a schema. Therefore, using formae instead of schemata, the theorem remains valid and convergence of the genetic algorithm is guaranteed if formae represent sets of candidate solutions with similar performance.

3.3 Crossover operators

We have established the properties that a genetic representation should have in order to be meaningful within the framework of forma theory. The question now is how to design useful crossover operators for a given genetic representation. Assuming that a meaningful genetic representation (i.e. an orthogonal covering basis Ψ) has been defined, we identify a set of properties that are desirable for the corresponding crossover operators. To describe these properties it is useful to introduce some definitions first:

A crossover operator is a function $\omega : \mathbb{P}(\mathcal{S}) \times \mathcal{K}_\omega \to \mathcal{S}$, where $\mathbb{P}(\mathcal{S})$ denotes the power set of $\mathcal{S}$ and $\mathcal{K}_\omega$ the set of parameters of the operator. For instance, for one-point crossover this set would have a single element that specifies the crossover point. The following notions play an important role in the design of representation-independent crossover operators:

Definition 3.6. Let P be a set of candidate solutions in a given search space $\mathcal{S}$. The *dynastic potential* of P, written $\Gamma(P)$, is defined as the set of solutions whose alleles are present in the set of alleles of all solutions in P, i.e., $\Gamma(P) = \{y \in \mathcal{S} | \forall \xi \ni y, \exists x \in P \text{ such that } x \in \xi\}$, where ξ is an allele.

The dynastic potential of a set of "parents" can be seen as the set of all possible solutions that can be generated by taking alleles from the union set of all the alleles present in the parents.

The following definition captures the notion of "intersection" of a given set of solutions.

Definition 3.7. Let P be a set of candidate solutions in a given search space $\mathcal{S}$, and let Ψ be a basis of equivalence relations. The *similarity set* of P, written $\Sigma(P)$, is defined as the set of the candidate solutions that are equivalent to all candidate solutions in P under a maximal subset of basic equivalence relations, i.e., let $\Psi^* \subseteq \Psi$ be the maximal subset of Ψ such that $\forall \psi^* \in \Psi^*$ and $\forall x_1, x_2 \in P$, $x_1 \psi^* x_2$. If $\Psi^* \neq \emptyset$, then $\Sigma(P) = \{y \in \mathcal{S} | \forall x \in P$ and $\forall \psi^* \in \Psi^*$ it holds that $x \psi^* y\}$. Otherwise, if $\Psi^* = \emptyset$ we define $\Sigma(P) = \mathcal{S}$ (i.e., the parents are not equivalent under any equivalence relation).

We now list the properties that crossover operators should have so that the combination of parent solutions leads to an effective exploration of the search space

Definition 3.8. Respect. We say that a crossover operator ω *respects* a genetic representation if and only if all the children produced by ω share all common relevant information with the parents, i.e. $\forall P \subseteq \mathcal{S}$, $\forall \theta \in \mathcal{K}_\omega$, $\omega(P, \theta) \in \Sigma(P)$.

Definition 3.9. Assortment. We say that a crossover operator ω *assorts* a genetic representation if and only if for every forma ξ to which a given parent belongs, ω can produce a child belonging to ξ, i.e., $\forall P \subseteq \mathcal{S}$, $\forall x \in P$ there is a $\theta \in \mathcal{K}_\omega$ such that $\forall \xi \ni x$, $\omega(P, \theta) \in \xi$.

Definition 3.10. Transmission. A crossover operator ω is said to be *transmitting* if and only if for every child produced by ω, every allele contained in the child is present in some of the parents, i.e. $\forall P \subseteq \mathcal{S}$, $\forall \theta \in \mathcal{K}_\omega$, $\omega(P, \theta) \in \Gamma(P)$.

Respect is the property responsible for exploitation: the common information of the parent set should be inherited by the children. Therefore, the search focuses on regions of the solution space $\mathcal{S}$ containing formae to which all parents belong. A respectful crossover operator generates offspring that belong to the similarity set $\Sigma(P)$ of the parents (lemma 42 in Radcliffe (1994)).

Assortment reflects the goal that the crossover operator should be able to produce all potential combinations of formae to which the parents belong. It therefore corresponds to the capacity of exploration of the genetic algorithm.

Transmission requires that the children produced by ω have all their alleles present in at least one parent, i.e., the children lie in $\Gamma(P)$. The following lemma states that transmission implies respect.

Lemma 3.11. *Let $P \subseteq \mathcal{S}$ be a given set of candidate solutions. Then their dynastic potential is contained in their similarity set* $\Gamma(P) \subseteq \Sigma(P)$.

For instance, assume a genetic representation with three genes A, B and C: gene A can take the values (alleles) A_1 and A_2, B can take the values B_1, B_2 and B_3 and gene C can take the values C_1 and C_2. Let $P = \{x, y\}$ be the set of parents, with $x = (A_1, B_1, C_1)$ and $y = (A_1, B_2, C_2)$. According to our definitions, $\Gamma(P) = \{(A_1, B_1, C_1), (A_1, B_1, C_2), (A_1, B_2, C_1), (A_1, B_2, C_2)\}$.

The equivalence relation 'The value of gene A is the same' is fulfilled by both parents. The similarity set is therefore $\Sigma(P) = \{(A_1, B_1, C_1), (A_1, B_2, C_1), (A_1, B_3, C_1), (A_1, B_1, C_2), (A_1, B_2, C_2), (A_1, B_3, C_2)\}$.
Clearly, $\Gamma(P) \subseteq \Sigma(P)$.

Lemma 3.11 is a slightly weaker version of lemma 48 in Radcliffe (1994) but the proof we provide is simpler. This new proof is included in Appendix A for completeness.

If crossover operators can be built that successfully assort and respect formae, this set of formae is said to be *separable*. Otherwise, we say that the formae are *non-separable*. If respect and assortment cannot be simultaneously achieved, one should try to build operators that randomly sacrifice respect for assortment, or the other way round.

In summary, forma theory introduces the notions of respect, assortment and transmission, which can be used to guide the design of representation-independent crossover operators. Given a representation that is well-suited to the problem considered, the crossover operators designed can be instantiated and define a specialized genetic algorithm. In the following subsections, we provide some examples of operators designed according to these principles.

3.3.1 Random Respectful Recombination (RRR)

The RRR operator achieves respect and ignores assortment. It simply samples a candidate solution in the similarity set of the parents at random with uniform probability.

Algorithm 8 Random Respectful Recombination (RRR) crossover.
Input: A set of parents P
Output: A child chromosome Θ.

- Compute $\Sigma(P)$, the similarity set of the parents.
- Randomly select $\Theta \in \Sigma(P)$

3.3.2 Random Equivalence Recombination (RER)

The goal in the design of the Random Equivalence Recombination (RER) operator is to select a random child from the set of all candidate solutions which, for each basic equivalence relation, are equivalent to some parent. Let $\Lambda(P) = \{x \in \mathcal{S} | \forall \psi \in \Psi \ \exists y \in P,$ such that $x \psi y$ holds$\}$. The definition of the RER operator is similar to that of RRR, except that the random uniform selection is made from $\Lambda(P)$ instead of from $\Sigma(P)$.

3.3.3 Random Assortment Recombination (RAR)

This crossover operation provides a good balance between respect and assortment for cases in which the genetic representation is non-separable. RAR crossover is described in Algorithm 16. The integer parameter $w \geq 0$ (which stands for *weight*) determines the

Algorithm 9 Random Equivalence Recombination (RER) crossover.
Input: A set of parents P
Output: A child chromosome Θ.

- Compute $\Lambda(P) = \{x \in \mathcal{S} | \forall \psi \in \Psi \; \exists y \in P, \text{ such that } \psi(x, y) \text{ holds}\}$.
- Randomly select $\Theta \in \Lambda(P)$

amount of relevant common information of all parents that is retained by the offspring. For $w = 0$, elements that are present in the chromosomes of all parents are not allowed in the child. Higher values of w assign more importance to the elements in the parents' similarity set. In the limit $w \to \infty$, only elements in $\Sigma(P)$ are selected and the operator converges to RRR.

Algorithm 10 Random Assortment Recombination (RAR) algorithm
Input: A set of parents P
Output: A child chromosome Θ.

- Construct a bag (a multiset, in which there can be repeated elements) G with the following formae $\forall \xi \in \Xi_P$:
  - If $\forall x \in P$, $x \in \xi$, introduce w copies of ξ.
  - else introduce 1 copy of ξ.
- Initialize $\Xi_\Theta = \mathcal{S}$
- While $G \neq \emptyset$:
  - Choose randomly $\xi \in G$.
  - If $\xi \cap \Xi_\Theta \neq \emptyset$, $\Xi_\Theta = \Xi_\Theta \cap \{\xi\}$
- Return a random element in Ξ_Θ.

3.3.4 Transmitting RAR (TransRAR)

The Transmitting RAR operator was originally introduced in Ruiz-Torrubiano and Suárez (2011). The principle used in the design of the RAR operator is to achieve an appropriate balance between respect and assortment. The key idea of the TransRAR operator is to guarantee gene transmission instead of respect. Transmission is preferable to strict respect because it guarantees that the genetic material of the parents' chromosomes will be transmitted to their offspring. The importance of transmitting genetic recombination has been emphasized in previous studies. For instance, Cotta and Troya (2003) establish the sufficient conditions to ensure that the pieces of information being transmitted are independent of the actual ordering in which formae are included the child. However, the condition they formulate relies on the separability of the genetic

representation. TransRAR recombination is effective also when the representation used is non-separable.

The operator is described in Algorithm 11. The TransRAR operator assorts formae because every combination of alleles in the parents can be obtained with a non-zero probability. It also transmits genes: if they are selected, alleles that are present in all parents are accepted with probability one. Formae that are present in only one of the parents are accepted with probability p. The value of p controls the degree of respectfulness of the operator. For lower values of p the operator favors respect. In the limit $p = 0$ all formae to which all parents belong are selected. For higher values of the parameter p, more formae not present in every parent are selected on average. In this manner transmission is favored.

Algorithm 11 The TransRAR crossover operator.

INPUT: A set P of parents.
OUTPUT: A child chromosome Φ.

1. Create the multiset U as the multiset-union of the formae to which some parent $x \in P$ belongs: $U = \biguplus\{\xi | \exists x \in P \text{ such that } x \in \xi\}$.
2. Initialize $\Xi_\Theta = \mathcal{S}$
3. While $U \neq \emptyset$:
   - Extract ξ_k from U at random with uniform probability. $U = U \setminus \{\xi_k\}$
   - If $\forall x \in P$, $x \in \xi_k$ then $\Xi_\Theta = \Xi_\Theta \cap \{\xi_k\}$ with probability 1.
   - else, $\Xi_\Theta = \Xi_\Theta \cap \{\xi_k\}$ with probability p.

3.4 Operators for cardinality constrained sets

In this section we instantiate the crossover operators defined in the previous section for the particular case of genetic representations based on sets of fixed cardinality. In this case, the relevant basis of equivalence relations is the following:

Let N be the number of available elements. For the sake of simplicity, we represent each element by an integer in the set $\mathcal{N} = \{1, \ldots, N\}$. We define the search space $\mathcal{S} = \{s \in \mathbb{P}(\mathcal{N}) \text{ such that } |s| = K\}$, where $|\cdot|$ denotes cardinality of a set and K, with $0 < K < N$, is the fixed size of the sets.

Let Ψ be a set of N equivalence relations $\Psi = \{\psi_1, \ldots, \psi_N\}$ such that $x\psi_i y$ is true if and only if $i \in x \cap y$. Then Ψ is a covering orthogonal basis of equivalence relations for $\mathcal{S}$. Since any two distinct elements $x \neq y$ in $\mathcal{S}$ are two subsets of $\mathcal{N}$ with cardinality $K < N$, then it is clear that there is at least one $\psi_i \in \Psi$ such that x and y are not equivalent under ψ_i (i.e., the equivalence relation corresponding to an element $i \in \mathcal{N}$ they do not have in common). Therefore, Ψ covers $\mathcal{S}$.

The two requirements for Ψ to form a basis are: *(i)* Ψ is independent and *(ii)* Ψ spans $\mathbb{E}(\mathcal{S})$, the set of all equivalence relations which can be defined over $\mathcal{S}$. The independence

of Ψ is directly established by the observation that no equivalence relation ψ_i can be constructed by intersections of other equivalence relations in Ψ. It is also apparent that every equivalence relation in $\mathbb{E}(\mathcal{S})$ can be expressed as the intersection of equivalence relations in Ψ. For instance, the equivalence relation "x and y are subsets of $\mathcal{N}$ containing the elements j and k" can be expressed as $\psi_j \cap \psi_k$. Thus Ψ spans $\mathbb{E}(\mathcal{S})$. In general, every equivalence relation in $\mathbb{E}(\mathcal{S})$ has the form "x and y are subsets of $\mathcal{N}$ containing all the elements in a given subset $s \subseteq \mathcal{N}$". Such equivalence relations can be expressed as $\cap_{i \in s} \psi_i$.

It can be easily seen that Ψ is an orthogonal set of equivalence relations, because any intersection of the formae induced by equivalence relations in Ψ is non-empty. For instance, let $s \subseteq \mathcal{N}$ and $\cap_{i \in s} \psi_i$ be an arbitrary intersection of equivalence relations in Ψ. This intersection is the set of all subsets that contain all the elements in s. This set is always non-empty.

A simple example illustrates the notions of transmission, respect and assortment for sets of fixed cardinality. Suppose we have to recombine the sets $\{1, 2, 3\}$ and $\{1, 2, 4\}$, where the set of all elements is {1,2,3,4,5}. The first individual contains the alleles 1, 2, 3, $\bar{4}$ and $\bar{5}$, where $\bar{i}$ denotes that element i is not part of the subset. Similarly, the second individual is determined by the alleles 1, 2, $\bar{3}$, 4 and $\bar{5}$. Respect requires that the children produced by the crossover operation are equivalent to all parents under the relations ψ_1 and ψ_2. Therefore, the possible children would be $\{1, 2, 3\}$, $\{1, 2, 4\}$ and $\{1, 2, 5\}$. Transmission, which is stronger than respect, requires the children to have every allele in common with some parent. The alleles present in the parents are 1, 2, 3, $\bar{3}$, 4, $\bar{4}$ and $\bar{5}$. Therefore, the possible children would be $\{1, 2, 3\}$ and $\{1, 2, 4\}$. Assortment requires every combination of the alleles to be a possible outcome of the recombination. For instance, $\{2, 3, 4\}$ should be possible with some probability. As can be seen, respect and assortment cannot be simultaneously achieved in this representation.

Therefore we have shown that Ψ is a meaningful genetic representation for $\mathcal{S}$. We can now write down specific versions of the representation-independent operators introduced in the previous section. In Algorithm 12 the pseudocode of RRR for fixed-size sets is given. It begins by calculating the intersection of the parents' chromosomes. Then the elements in this intersection are selected at random until the child is complete (i.e., it contains exactly K elements) or until the intersection is empty. In this last case, the child chromosome is completed with elements selected at random from $\mathcal{N}$.

The pseudocode in Algorithm 13 shows the corresponding set version for RER. Instead of taking the intersection of the parents' chromosomes, the union set is calculated. Then elements are selected at random until the desired cardinality is reached. Note that the union set of two sets of cardinality K has cardinality at least K. Therefore, the set from which elements are selected (originally the union set) cannot be empty before the child is complete.

The RAR crossover operator for sets is detailed in Algorithm 14. As mentioned earlier, a positive integer w is used to control the amount of common information of the parents that is retained by the offspring. The RAR operator makes use of six sets: Set A is the intersection set, which consists of elements that appear in both parents. Set

Algorithm 12 The Random Respectful Recombination operator for sets of fixed cardinality.

Input: Two parents I_1 and I_2, and a fixed cardinality K.
Output: A child chromosome Θ.

1. $\Theta = I_1 \cap I_2$.
2. While $|\Theta| < K$:
   - Extract randomly an item $s \in \mathcal{N}$.
   - $\Theta = \Theta \cup \{s\}$.

Algorithm 13 The Random Equivalence Recombination operator for sets of fixed cardinality.

Input: Two parents I_1 and I_2, and a fixed cardinality K.
Output: A child chromosome Θ.

1. $\Theta = \emptyset$.
2. $U = I_1 \cup I_2$.
3. While $|\Theta| < K$:
   - Extract randomly an item $s \in U$.
   - $\Theta = \Theta \cup \{s\}$.

B includes the elements that are absent from all parents. Sets C and D contain the elements that are present in only one parent. Set E is initially empty ($E = \emptyset$). An additional set G is built with w copies of the elements from A and B and one copy from the elements in C and D. The elements in G include a label that specifies the set from which they originate. A child chromosome is generated by extracting one element from G in each iteration. Let g be the element extracted from G: If it originates from A or C and $g \notin E$, then it is directly included in the child's chromosome. If $g \in B$ or $g \in D$, then it is included in set E. The process is terminated when the child has the required cardinality K or when $G = \emptyset$. If the latter happens, then the child is completed with elements still not included. These elements are selected at random from $\mathcal{N}$ as in RRR.

Finally, Algorithm 15 provides an instantiation of TransRAR for sets of fixed cardinality. In this pseudocode the *extension function* $E_{I_1 I_2}$ for the parent chromosomes I_1 and I_2 denotes the following function

$$\begin{aligned} E_{I_1 I_2} &: I_1 \cup I_2 \to \{0,1\} \\ E_{I_1 I_2}(u) &= \begin{cases} 1 & \text{if } u \in I_1 \cap I_2 \\ 0 & \text{if } u \in I_1 \cup I_2 - (I_1 \cap I_2). \end{cases} \end{aligned} \tag{3.9}$$

Algorithm 14 The Random Assortment Recombination operator for sets of fixed cardinality.

Input: Two parents I_1 and I_2, and a fixed cardinality K.
Output: A child chromosome Θ.

1. Create auxiliary sets A, B, C, D, E:
   - $A =$ elements present in both parents.
   - $B =$ elements not present in any of the parents.
   - $C \equiv D$ elements present in only one parent.
   - $E = \emptyset$.
2. Build set $G = \{w$ copies of elements from A and B, and 1 copy of elements in C and $D\}$
3. While $|\Theta| < k$ and $G \neq \emptyset$:
   - Extract $g \in G$ without replacement.
   - If $g \in A$ or $g \in C$, and $g \notin E$, $\Theta = \Theta \cup \{g\}$.
   - If $g \in B$ or $g \in D$, $E = E \cup \{g\}$.
4. If $|\Theta| < k$, add elements chosen at random from $\mathcal{N} - \Theta$ until chromosome is complete.

Algorithm 15 The TransRAR crossover operator for sets of fixed cardinality.

Input: I_1, I_2 the parent chromosomes of cardinality K.
Output: Φ offspring of cardinality K.

1. Create multiset U as the multiset-union of the parent chromosomes: $U = I_1 \uplus I_2$.
2. Assign each element $u \in U$ the attribute $E_{I_1 I_2}(u)$.
3. While child chromosome Φ is incomplete ($|\Phi| < K$):
   - Extract an element u_k from U uniform randomly. $U = U \setminus \{u_k\}$
   - If $E_{I_1 I_2}(u_k) = 1$, then $\Phi = \Phi \cup \{u_k\}$ with probability 1.
   - else, $\Phi = \Phi \cup \{u_k\}$ with probability p.
   - If $U = \emptyset$, select $K - |\Phi|$ elements randomly to complete chromosome.

This function determines an auxiliary attribute that is assigned to each element in the union of the parents' chromosomes. This attribute has the value 1 when the element belongs to the intersection set $I_1 \cap I_2$ and 0 otherwise. We then create a multiset in which each element has multiplicity 2 if it belongs to both parents, and 1 otherwise. If the attribute $E_{I_1 I_2}$ is equal to 1, the new element is included in the child chromosome with probability 1. If not, the element is included with a probability p. An adequate balance between respect and diversity is achieved for intermediate values of p. After exploratory experiments, the value $p = 1/2$ is used in the empirical evaluations because it provides good overall results in all the problems investigated, as will be shown in Part II of this thesis.

3.4.1 Complexity analysis

Here we present an analysis of the run-time complexity of the set versions of the crossover operators RER, RRR, RAR and TransRAR.

Let K and N be defined as in the previous sections. The RER and RRR algorithms can be implemented in $O(K \log K)$ time: once the parent chromosomes are sorted, calculating the intersection and the union requires only linear time. The run-time complexity of the RAR operator is computed for a fixed value of the parameter $w > 0$: Step 1 in Algorithm 14 can be completed in $O(N) + O(K \log K)$ operations, assuming that the parent chromosomes have to be sorted first. Assume that building the multiset G requires constant time $O(1)$. In the worst case step 3 of the algorithm requires $|G|$ iterations. By construction, $|G| = O(wN)$. The determination of the set from which g originates requires constant time if each element is appropriately labeled when G is constructed. Assuming that set E is kept sorted at every step, then searching g in E requires $O(\log N)$ steps. Therefore, the total worst-case complexity is

$$f(N, K, w) = O(1) + O(N) + O(K \log K) + O(wN \log N) = O(wN \log N). \tag{3.10}$$

The TransRAR operator can be implemented very efficiently. Let I_1 and I_2 be the parent chromosomes of cardinality K, and let the extension function $E_{I_1 I_2}$ be defined as in Eq. (3.9). Steps 1 and 3 require $O(K)$ operations. Consider Step 2. The value of the function $E_{I_1 I_2}$ can be obtained by sorting the elements in U and then removing one element u for which we calculate $E_{I_1 I_2}(u)$. Note that element u is repeated if and only if it is in the intersection set of the two parent sets. Therefore binary search can be used to search for an additional copy of u. If a copy is found, then $E_{I_1 I_2}(u) = 1$. Otherwise $E_{I_1 I_2}(u) = 0$. Since sorting requires $O(K \log K)$ steps and removing and searching in the sorted multiset require $O(\log K)$ steps, the worst case complexity $f(K)$ of the algorithm is

$$f(K) = O(K) + O(K \log K) = O(K \log K). \tag{3.11}$$

Note that the worst-case complexity of TransRAR is expressed in terms of the size of the subset K, whereas in the case of RAR the complexity is a function of the size of the *total* number of elements, $N > K$. In many cases of practical interest $N \gg K$. Therefore, the worst-case complexity of RAR is larger than TransRAR. This does not

necessarily imply that TransRAR is more efficient than RAR on average. Note also that the complexity of TransRAR does not depend on the parameter p either. Nonetheless, the worst-case complexity of RAR depends explicitly on the parameter w.

3.5 Summary and Discussion

In this chapter the framework of forma theory has been reviewed. Several definitions have been introduced to characterize genetic representations that are meaningful according to this framework. The genetic encodings designed according to the principles of forma theory include problem knowledge in the form of equivalence relations over the space of feasible solutions. This approach presents several advantages over standard analysis based on schemata, which do not take into account the specific structure introduced by the cost function in the search space. By taking into account specific domain knowledge in the chromosome representation and in the design of the crossover operators we expect to obtain better solutions to the optimization problem at a lower computational cost.

It is possible to design representation-independent crossover operators in a very general way using abstract concepts of forma theory, such as respect, assortment and transmission. Examples of these types of operators that have been used in previous studies are RRR, whose goal is to achieve respect, and RAR, an operator that balances respect and assortment. In this thesis we introduce the TransRAR operator, which is designed to balance assortment with transmission instead of with respect.

In forma theory a genetic representation is defined as an orthogonal basis of equivalence relations that covers the search space. The set of equivalence relations selected should bring out the structure induced by the objective function in the search space. Once a particular representation is chosen the general crossover operators defined in an abstract level can be instantiated for that particular representation. If the chosen representation appropriately reflects the relevant features of the problem considered, one can use the instantiated versions of the abstract operators directly, without much fine-tuning of their parameters. Otherwise, if the basis of equivalence relations does not reflect all relevant aspects of the problem, more sophisticated operators need to be be designed.

Finally, this chapter proposes the use of a particular chromosome encoding based on sets with a fixed number of elements to address cardinality-constrained optimization problems. Particular implementations of the RER, RRR, RAR and TransRAR crossover operators for this type of representation are given and their complexity is analyzed. In the remainder of this thesis we investigate the improvements of performance that can be obtained using these specially designed genetic algorithms in problems of practical interest.

Part II

Applications

In the following chapters, we present applications of the general framework proposed for the solution of optimization problems with cardinality constraints. The first optimization task considered is the 0/1 knapsack problem. This problem is intended to serve as an illustration of how the metaheuristics investigated can be used to perform the combinatorial search when cardinality constraints are considered. In these problems the performance of metaheuristic approaches is compared with exact solutions.

In the other domains of application investigated the problem without the cardinality constraint can be solved using quadratic programming. In particular, we consider the problems of determining sparse principal components in statistics, and the optimal portfolio selection and index tracking problems in quantitative finance. The combination of metaheuristics and quadratic programming provides an efficient approach to the solution of these problems. The sparse solutions obtained are stable and robust with respect to the inputs of the optimization, and, in the pertinent cases, exhibit good out-of-sample performance. This finding is one important contribution of this thesis. Additionally, one can use pruning heuristics to reduce the dimensionality of the problem. This provides a practical procedure to tackle high-dimensional problems, whose solution would be unfeasible without the dimensionality reduction.

Finally, we address the consensus tree problem in phylogenetics. In this problem dimensionality reduction techniques are also used to improve the efficiency of exact methods based on branch-and-cut techniques. Additional improvements can be obtained using hybrid metaheuristics. These improvements in efficiency are illustrated using both simulated trees and phylogenetic trees obtained from actual genetic data.

CHAPTER 4

THE KNAPSACK PROBLEM

The 0/1 knapsack problem is a classical example of constrained combinatorial optimization. It consists in maximizing the total profit of the elements contained in a knapsack. Since the knapsack has only limited capacity, the problem is to select a subset of elements such than the sum of the profits of the elements in the subset is maximal and the capacity of the knapsack is not exceeded. In its original formulation, the problem does not include a cardinality constraint. However, it can be approached by solving a series of cardinality constrained problems. In this chapter, the knapsack problem is used as benchmark for testing the ideas presented in Chapters 2 and 3.

4.1 Introduction

Knapsack problems are a family of combinatorial optimization problems that involve selecting a subset from a pool of items. An extensive survey of methods to solve knapsack problems is presented in Kellerer et al. (2004). There are several variants of knapsack problems. In the *bounded* knapsack problem (Pisinger (2000)), one is allowed to take a limited number of items from each type. The *unbounded* version of the problem (Poirriez et al. (2009)) allows any number of elements to be selected from each type. *Multidimensional* knapsack problems (Puchinger et al. (2010)) introduce more than one knapsack, and consider a linear restriction per knapsack. In this case, repetitions of the elements are not allowed in the knapsack. A quadratic objective function instead of a linear one is considered in the *quadratic* knapsack problem (Pisinger (2007)). The profit is a quadratic form computed from a symmetric matrix of non-negative profits.

In this chapter, we restrict our attention to the classical 0/1 knapsack problem. Both exact and approximate methods have been used to address this problem, which is NP-complete and cannot be solved in polynomial time (Miller and Thatcher (1972)). In Balas and Zemel (1980), the authors introduce the notion of *core* of a knapsack problem and used it to construct efficient heuristics for its solution. The core can be understood as the subset of the original problem variables which remains if we eliminate the variables

for which there is certainty that that are included or that they are not included in the optimal solution. Their approach is based on solving the LP relaxation of the problem to obtain an approximation to the core. A dynamic programming approach was used in Martello et al. (1999). The authors also derived tighter bounds for the LP relaxation of the problem to improve the performance of the algorithm. Exact algorithms based on branch-and-bound approaches and dynamic programming were reviewed in Pisinger (2005). In this reference the author focuses on constructing new test instances of the problem that are difficult to solve by state-of-the-art algorithms. Genetic algorithms with a transposition mechanism were tested in Simões and Costa (2001). In their study, GAs that use transposition performed better than GAs that use standard crossover operators. In Ku and Lee (2001) a *set oriented* GA was presented. The representation and operators employed in this study exhibit some similarities with those introduced in the framework on forma theory. In Larrañaga and Lozano (2002) EDAs are used to address problems of this type. The authors of this study introduce binary and permutation based representations and propose an effective mechanism to handle unfeasible individuals.

The objective of this chapter is to illustrate the application of the ideas from Chapters 2 and 3 using the knapsack problem as benchmark. Since no subordinate optimization is necessary, one can concentrate on the cardinality constraint and compare the various methods proposed without having any influence from difficulties in the solution of subordinate optimization problems. Therefore, the focus will be on the comparison of the perfomance of the various metaturistics considered: GAs with binary and set encoding, simulated annealing and estimation of distribution algorithms.

The rest of this chapter is organized as follows: Section 4.2 introduces the 0/1 knapsack problem. In Section 4.3, the hybrid approaches used to solve the problem with cardinality constraints are detailed. Section 4.4 presents the results of computational experiments performed to compare the metaheuristic approaches with an exact solution based on a branch-and-bound approach. Finally, Section 4.5 outlines the conclusions of the chapter.

4.2 Optimization model

Consider a set of N items that can be used to fill up a knapsack. Each of these elements has a profit $p_i > 0$ and a weight $w_i > 0$ associated with it. The objective is to identify the subset of items that maximizes the accumulated profit, subject to the constraint that the overall weight does not exceed a fixed capacity $W > 0$

$$\max_{\mathbf{z}} \quad \sum_{i=1}^{N} p_i z_i \tag{4.1}$$

$$\text{s.t.} \quad \sum_{i=1}^{N} w_i z_i \leq W \tag{4.2}$$

$$z_i \in \{0,1\} \quad \text{for } i = 1, \ldots, N. \tag{4.3}$$

In this formulation of the problem, the components of the binary vector $\mathbf{z}$ encode which items are included ($z_i = 1$) or excluded ($z_i = 0$) from the knapsack. Note that the model does not have an explicit cardinality constraint. Nevertheless, the optimum can be found by solving a series of cardinality constrained knapsack problems of the form

$$\max_{\mathbf{z}} \quad \sum_{i=1}^{N} p_i z_i \tag{4.4}$$

$$\text{s.t.} \quad \sum_{i=1}^{N} w_i z_i \leq W \tag{4.5}$$

$$\sum_{i=1}^{N} z_i = k \quad \text{for } k = 1, \ldots, K. \tag{4.6}$$

The k-th element in this sequence of problems is a knapsack problem with the restriction that only k items can be included in the knapsack. Let $\mathbf{z}^k$ be the optimal solution to the problem with cardinality k. Then we take as final solution $\mathbf{z}^* = \arg\max_k \sum_{i=1}^{N} p_i z_i^k$. Note that we can also apply the following early termination criterium: if a problem with cardinality k is not feasible, the problems with cardinalities $k+1, \ldots, N$ need not be investigated.

4.3 Hybrid approaches

We now describe the optimization metaheuristics that will be applied to the standard knapsack problem. Since this problem has no continuous variables or additional constraints, one can directly evaluate the objective function of the candidate solution (4.1) without having to solve a subordinate optimization problem as is the case in most of the applications analyzed in this thesis. To make a fair comparison of all the methods investigated, the capacity constraint is handled in the same manner in all of them: if the candidate subset exceeds the knapsack capacity W, a penalty proportional to the weight surplus is applied. This penalty is not related to the term that could be used to penalize violations of the cardinality constraints as is done in some of the metaheuristics considered (see subsection 2.3.2).

In the family of genetic algorithms (GA), we consider on the one hand GAs with binary encoding with uniform crossover and a linear penalty term for the cardinality constraint, and, on the other hand, GAs with set encoding that employ the different crossover operators described in Section 3.3: RER (Algorithm 13), RRR (Algorithm 12), RAR (Algorithm 14) and TransRAR (Algorithm 15). We also apply simulated annealing with a set representation (Algorithm 1) where the neighboring states in the search are generated using the operator that is used for mutation in the GAs. This operator exchanges one element present in the candidate solution with an element not present in it.

Finally, the PBIL algorithm from the EDA family (see Algorithm 3) with update eq. (2.10) is included in the comparison. To sample individuals of a given cardinality from the distribution of gene values, we use the sampling scheme detailed in Algorithm 4.

4.4 Results

To compare the performance of the different optimization methods analyzed, we use the testing protocol proposed in Michalewicz (1996) and Simões and Costa (2001). Three types of problems, defined in terms of two parameters $v, r \in \mathbb{R}^+$, $v > 1$, are considered:

(1) *Uncorrelated*: Weights and profits are generated uniformly at random in $[1, v]$.

(2) *Weakly correlated*: Weights are generated uniformly at random in $[1, v]$. Profits are generated in the interval $[w_i - r, w_i + r]$.

(3) *Strongly correlated*: Weights are generated uniformly at randmon in $[1, v]$. The value of the profit is determined as $p_i = w_i + r$.

In general, knapsack problems with correlations between weights and profits are more difficult to solve than problems in which the weights and profits are independent (Pisinger (2005)). We use $v = 10$, $r = 5$ and a capacity $W = 2v$, which tends to result in solutions with only a small number of items.

The results reported are averages over 25 realizations of each problem, which are solved using the different approximate methods: a standard GA with linear penalty, different GAs with set encoding using the RER, RRR, RAR and TransRAR crossover operators, SA and PBIL. A geometric annealing schedule $T_k = \gamma T_{k-1}$ with $\gamma = 0.9$ is used to obtain the solutions in SA. The GAs evolve populations composed of 100 individuals. The probabilities of crossover and mutation are $p_c = 1$, $p_m = 10^{-2}$, respectively. In PBIL, a population composed of 1000 individuals is used. The probability distribution is updated using the best 10% of the individuals. The smoothing parameter α is 0.1. Exact results obtained with the solver SYMPHONY from the COIN-OR project (Ladanyi et al. (2009)) implementing a branch-and-cut (B&C) approach (Padberg and Rinaldi (1991)), are also reported for reference. In the strongly correlated problems, it has not been possible to find the exact solution with the amount of time allocated.

The value of the parameter p in the TransRAR operator (Alg. 11) is determined in exploratory experiments. Figure 4.1 presents a typical outcome of these experiments. This figure displays the best average profit obtained in a knapsack problem with 500 elements and no correlation as a function of p, the probability of accepting an element that is present in only one of the parents once it has been drawn. As can be seen from this plot, large values of p lead to a sharp deterioration of performance. The reason is that too much variability is introduced in the search and the algorithm is not able to preserve formae that perform well. Values of p close to 1/2 yield the best results. Fairly good results are also obtained for lower values of p. In this particular case, high degrees of respect (and therefore lower variability) lead to good results. This is

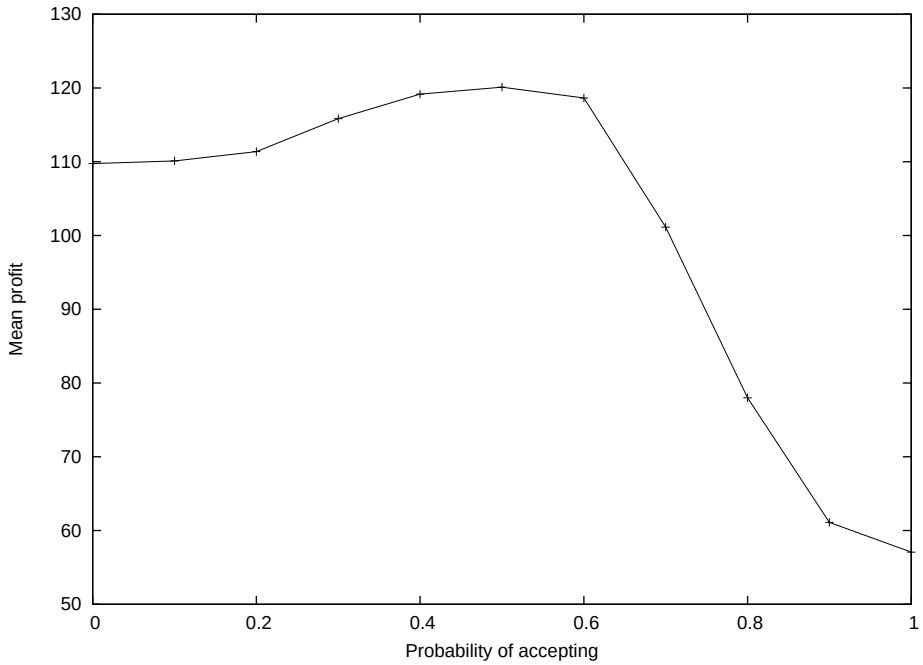


Figure 4.1: Mean profit obtained by TransRAR in the knapsack problem with 500 items and no correlation as a function of the probability of acceptance.

generally the case, but too much respect can result in premature convergence, in which case suboptimal solutions would be obtained. Experiments in other problems give similar results. Therefore, no further adjustments of this parameters are made and the value $p = 1/2$ is used in all cases.

The optimal value of the parameter w in RAR has also been determined in exploratory experiments. In view of the results of these experiments, the best performance is obtained with $w = 1$. This is different from the value $w = 2$ proposed as a natural choice and used in the original study by Radcliffe (Radcliffe and George (1993)). In our experiments, the more standard choice $w = 2$ leads to suboptimal solutions because too much common information of the parents is exploited by the crossover operator. The consequence is that the algorithm frequently converges prematurely and becomes trapped in local optima.

Tables 4.1 and 4.2 display the average profit obtained and the total execution time (in seconds) for each method. The experiments were performed on an Intel Core Duo with 3.00 Ghz processor speed and 2 Gb RAM. None of the approximate methods reaches the optimal profit, which is calculated using an exact branch-and-cut method. The highest profit obtained by an approximate optimization method is highlighted in boldface.

In most cases, the best results are obtained by TransRAR. Nevertheless, the differences in quality between the solutions obtained by RAR and by TransRAR are often very small. In terms of efficiency, the computational cost of TransRAR is much lower than RAR, RER and RRR. The smallest speed-up ratio between TransRAR and the following fastest method in the GA family is 2.1 in the strong correlation case with a universe of 100 elements. In general the algorithms that use a set encoding (set GAs and

Table 4.1: Comparison of average profits with restrictive capacity.

Algorithm	Problem	No. items		
		100	250	500
GA Linear penalty	No corr.	79.36	90.63	95.93
	weak	52.97	59.07	60.40
	strong	76.19	83.98	84.52
GA RER	No corr.	80.84	99.81	109.43
	weak	53.39	63.67	68.43
	strong	78.98	92.40	96.20
GA RRR	No corr.	80.74	96.61	104.68
	weak	52.96	61.80	65.21
	strong	78.76	89.99	94.79
GA RAR	No corr.	**82.09**	105.34	119.88
	weak	**54.38**	66.24	74.17
	strong	79.77	**94.20**	101.40
GA TransRAR	No corr.	81.97	**105.46**	**120.11**
	weak	**54.38**	**66.69**	**74.78**
	strong	**79.78**	**94.20**	101.40
SA	No corr.	80.70	102.91	118.07
	weak	53.53	65.13	73.40
	strong	79.73	94.15	**102.16**
PBIL	No corr.	81.89	104.51	117.28
	weak	54.33	65.85	72.05
	strong	78.99	92.39	96.60
B&C (exact)	No corr.	82.11	106.43	123.93
	weak	54.43	67.10	76.61
	strong	-	-	-

SA) exhibit the best performance. However, they require longer times to reach a solution as well, especially SA. PBIL obtains good results in small uncorrelated problems, being also competitive in medium size instances, but the algorithm encounters some difficulties in the larger instances. This is explained by the fact that the sampling and estimation of probability distributions becomes progressively more difficult as the dimensionality of the problem increases. Furthermore, PBIL assumes statistical independence between the variables, which makes the algorithm perform worse on problems in which correlations are present. As expected, the standard GA with linear penalty has a very poor performance in all the knapsack problems analyzed. The linear penalty function misleads the search and cannot conduct the algorithm efficiently towards promising regions in the search space.

4.5 Summary and Conclusions

In this chapter, we have used the 0/1 knapsack problem as benchmark for testing the ideas and algorithms proposed in Chapter 2 and 3. The focus has been placed on comparing the performance of the different algorithms in solving the cardinality constrained

Table 4.2: Comparison of total execution times with restrictive capacity.

Algorithm	Problem	No. items		
		100	250	500
GA Linear penalty	No corr.	26.2	38.1	57.2
	weak	26.9	38.3	56.1
	strong	26.2	37.8	55.3
GA RER	No corr.	34.8	53.5	66.1
	weak	34.4	51.2	66.8
	strong	37.1	57.3	70.8
GA RRR	No corr.	24.6	34.7	42.4
	weak	24.2	32.6	41.1
	strong	25.8	36.5	49.8
GA RAR	No corr.	46.0	106.3	199.9
	weak	45.6	102.6	200.4
	strong	48.4	109.0	204.0
GA TransRAR	No corr.	20.9	31.9	41.9
	weak	21.0	30.8	42.7
	strong	22.5	33.3	43.2
SA	No corr.	98.4	284.4	531.7
	weak	99.9	286.4	531.9
	strong	98.7	286.0	525.6
PBIL	No corr.	24.1	47.4	91.5
	weak	24.2	47.7	87.9
	strong	24.0	47.3	86.9
B&C (exact)	No corr.	62.0	178.7	568.8
	weak	81.8	180.3	560.1
	strong	-	-	-

sub-problems resulting from our problem formulation. We used a testing protocol where instances with various degrees of dependency between weights and profits were generated and compared the results with an exact approach based on branch-and-cut.

From the results obtained by the methods investigated in a sufficient number of randomly generated instances of the problem, we conclude that the GA approaches that use the set representation introduced in Chapter 3 outperform the binary GA, SA and PBIL. Moreover, there is only a small gap between the true optimal solutions found by branch-and-cut and the approximate solutions obtained by the best of the set GAs. In these types of problems the RER and RRR crossover operators obtain worse results than RAR and TransRAR. These operators either generate too much variability (in case of RER) or exploit too much common information from the parents (in RRR), and consequently fail to produce high-quality solutions. By contrast, the GA with RAR and TransRAR crossover operators achieve a good balance between assortment and either respect (RAR) or transmission (TransRAR). Among these the GA with TransRAR crossover obtains the best results both in terms of quality and speed. It is noteworthy that the best overall performing algorithm, the set GA with TransRAR, is also the fastest method. This is due to the fact that the TransRAR operator can be implemented very efficiently.

CHAPTER 5

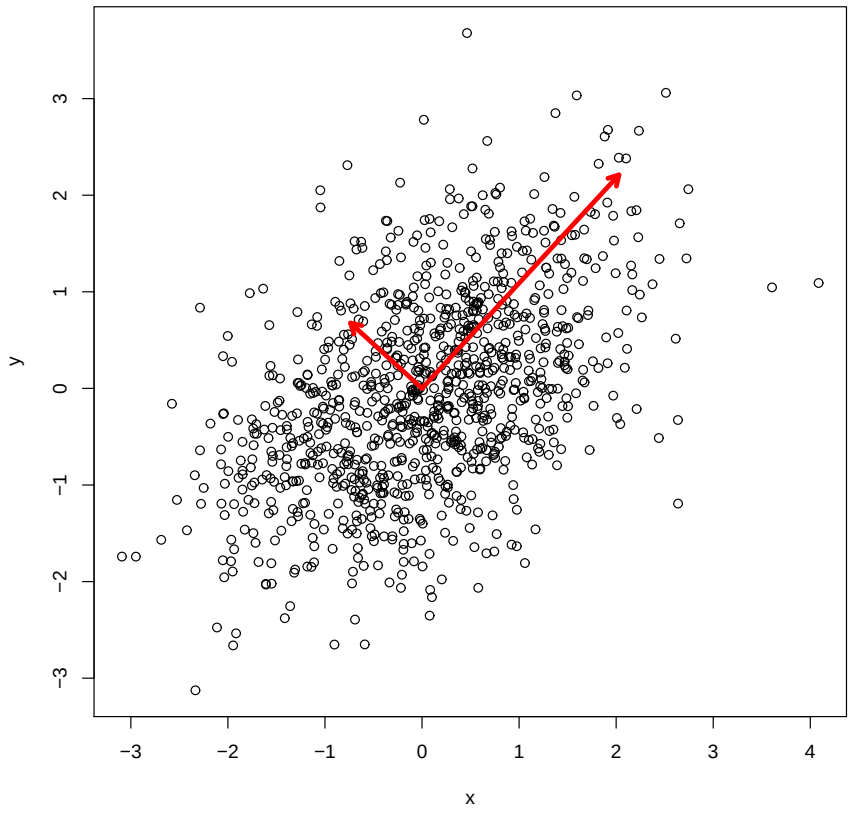
SPARSE PRINCIPAL COMPONENT ANALYSIS

Principal component analysis (PCA) is a data analysis technique that consists in finding a set of orthogonal directions along which the variance of the observed data is as large as possible. The first principal component is the direction along which the data has the largest variance. Assuming that k principal components have been identified, the $(k+1)$th principal component is the direction of largest variance that is orthogonal to the k principal components identified. This problem, without additional constraints, can be formulated as a quadratic programming problem for which efficient algorithms exist. In general, the principal components obtained are a combination of all input variables, which complicates their interpretation. The addition of a cardinality constraint to enforce sparsity makes the resulting combinatorial problem (known as Sparse PCA) difficult to solve. To address this problem, we propose to apply the hybrid algorithms introduced in the first part of this thesis: metaheuristics are used to generate candidate solutions of the combinatorial optimization. These candidate solutions are evaluated using the solution of a subordinate optimization problem that can be efficiently solved using quadratic programming. Finally, the efficacy of his approach is illustrated in a series of experiments using data from real-world applications.

5.1 Introduction

Principal Component Analysis (PCA) is a dimensionality reduction technique that is frequently used in data analysis, compression and visualization. It consists in finding the directions along which the variance of the data is maximized. These directions, which can be expressed as linear combinations of the initial variables, are known as *principal components*. The degree of participation of each variable is measured by its coefficient in the linear combination or *loading*. Figure 5.1 shows an example of PCA using a sample of two-dimensional random variables. Principal components can be obtained by computing the eigenvectors of the covariance matrix of the data. Since the covariance matrix is symmetric, its eigenvectors are real and form an orthogonal basis in the original space. The identification of principal components can also be formulated

Figure 5.1: Example of PCA on random data. The arrows indicate the principal components obtained.



as a constrained optimization problem. To find the first m principal components one needs to find the m directions that maximize the explained variance, subject to the constraint that the solution vectors are normalized and orthogonal to each other. This is a quadratic programming problem, in which the function to be maximized is the variance of the projection of the random vector X composed by the original problem variables, $X = \{x_1, x_2, \ldots, x_N\}$ onto an m-dimensional space.

A drawback of this method is that, in general, the principal components are linear combinations of *all* the original variables in the problem. This makes their analysis and interpretation difficult. Requiring sparsity of these components is a way to improve their interpretability. Sparse principal components are linear combinations of a small subset of the original variables that explain as much of the variance of the original data as possible. Since they have loadings in only a few of the problem variables they are easier to interpret. Furthermore, the variables that are more likely to appear in the first sparse principal components can be identified as the most relevant to the problem. Additionally, sparse representations for this principal components can be used to reduce storage costs. However, finding sparse principal components is an NP-hard problem (d'Aspremont et al. (2008a)). This is a consequence of the cardinality constraint limiting the number of loadings that are allowed to take non-zero value. The standard optimization techniques

that can be used to efficiently solve the original problem need therefore to be adapted to handle the cardinality constraint.

The problem of finding sparse principal components has received much attention in recent literature. A simple heuristic based on setting to zero the loadings that are smaller than a given threshold (known as "simple thresholding") is proposed in Cadima and Jolliffe (1995). The solutions can be good approximations to sparse PCs, but are no longer orthogonal. Methods based on the lasso can also be applied to SPCA (Jolliffe et al. (2003)). The technique proposed in this reference (called SCoTLASS, "simplified component technique - LASSO") employs a project gradient method. In the lasso, a L_1-norm penalty for non-zero values of the factor loadings is used. A larger weight of the penalty term in the objective function leads to sparser models. However, it is not possible to have direct control of the number of non-zero coefficients in the solution. In Zou et al. (2006) SPCA is formulated as a regression problem that can be approximately solved using convex optimization techniques. Greedy search was used in Moghaddam et al. (2005). In this work, a simple local heuristic is proposed to improve candidate solutions. Furthermore, the value of cardinality constraint is determined automatically for each problem based on the eigenvalues of the covariance matrix and a target minimum value of the variance to be explained. Since the k-th smallest eigenvalue of the covariance matrix is a lower bound on the maximum possible variance with this cardinality, one may choose the smallest k for which the corresponding eigenvalue is equal or above the target variance. A cardinality constraint that explicitly limits the number of non-zero loadings in the components is considered in d'Aspremont et al. (2007). In this reference an approximate solution is obtained by solving a relaxed version of the original problem. The relaxation consists in replacing the cardinality constraint with a convex inequality constraint. The relaxed problem can then be solved using semidefinite programming (Vandenberghe and Boyd (1996)).

This chapter is organized as follows: Section 6.2 formulates SPCA as a constrained optimization problem. In Section 6.3, the hybrid optimization approach introduced in the first part of this thesis is applied to the problem using several optimization meta-heuristics. The results of an empirical evaluation of the solution methods proposed are presented in Section 6.4. These results are obtained on randomly generated instances and the "pitprops" dataset. Finally, the conclusions of this chapter are summarized in 6.5.

5.2 Optimization Model

In this section SPCA is formulated as a constrained quadratic optimization problem. The strategy adopted is to identify one principal component at a time. To obtain m principal components, a sequence of m cardinality constrained optimization problems need to be solved. Additional constraints are included to enforce that the solutions of each problem in the sequence are normalized and are also orthogonal to the sparse principal components obtained earlier in the sequence. In this way, we guarantee that the resulting components actually form an orthonormal basis.

The first sparse principal component is obtained by solving the cardinality-constrained optimization problem

$$\max_{\mathbf{x},\mathbf{z}} \quad \mathbf{x}^{[\mathbf{z}]^{\mathrm{T}}} \cdot \mathbf{A}^{[\mathbf{z},\mathbf{z}]} \cdot \mathbf{x}^{[\mathbf{z}]} \tag{5.1}$$

$$\text{s.t.} \quad \|\mathbf{x}^{[\mathbf{z}]}\|_2 = 1 \tag{5.2}$$

$$\mathbf{z}^{\mathrm{T}} \cdot \mathbf{1} \leq K, \tag{5.3}$$

where $\mathbf{A}$ is the $n \times n$ covariance matrix of the data. The elements of the binary vector $\mathbf{z}$ encode whether the principal component has a non-zero projection along the corresponding direction ($z_i = 1$) or not ($z_i = 0$). The $\mathbf{x}^{[\mathbf{z}]}$ is obtained by removing the components for which $z_i = 0$ from $\mathbf{x}$. Similarly $\mathbf{A}^{[\mathbf{z},\mathbf{z}]}$ is obtained by eliminating from $\mathbf{A}$ the rows and colums for which $z_i = 0$.

Once the first principal component $\mathbf{x}_1$ has been found, the covariance matrix $\mathbf{A}$ is deflated as follows

$$\mathbf{A}_1 = \mathbf{A} - \left(\mathbf{x}_1^{\mathrm{T}} \cdot \mathbf{A} \cdot \mathbf{x}_1\right) \mathbf{x}_1 \, \mathbf{x}_1^{\mathrm{T}}, \tag{5.4}$$

and a new problem of the form (5.1)-(5.3) is solved in terms of the deflated covariance matrix $\mathbf{A}_1$ with the additional orthogonality constraints

$$\mathbf{x_j}^{\mathrm{T}} \cdot \mathbf{x} = 0 \quad j = 1, \ldots, m, \tag{5.5}$$

where $\mathbf{x_j}$ is the $j-$th principal component and m is the number of PCs obtained up to that moment. Note that the optimum for the first principal component is attained exactly with cardinality K because of the *interlacing property* (Moghaddam et al. (2005)) of symmetric matrices: Let $\{\lambda_i(\mathbf{A}_n)\}_{i=1}^n$ be the spectrum of the $n \times n$ symmetric matrix $\mathbf{A}_n$, where $\lambda_1(\mathbf{A}_n) \leq \lambda_2(\mathbf{A}_n) \leq \cdots \leq \lambda_n(\mathbf{A}_n)$, and let $\mathbf{A}_{n-1}$ be a submatrix of size $n-1$. The eigenvalues of $\mathbf{A}_{n-1}$ are interlaced with the eigenvalues of $\mathbf{A}_n$

$$\lambda_1(\mathbf{A}_n) \leq \lambda_1(\mathbf{A}_{n-1}) \leq \lambda_2(\mathbf{A}_n) \leq \cdots \leq \lambda_{n-1}(\mathbf{A}_n) \leq \lambda_{n-1}(\mathbf{A}_{n-1}) \leq \lambda_n(\mathbf{A}_n) \tag{5.6}$$

This implies that eliminating one variable also shrinks the spectrum of the reduced matrix by incrementing the minimum and decrementing the maximum eigenvalues. It is therefore sufficient to consider the equality cardinality constraint $\mathbf{z}^{\mathrm{T}} \cdot \mathbf{1} = K$. Note that in the subsequent problems this may not be true because of the orthogonality restriction, and therefore all cardinalities strictly below K have to be taken into consideration as well.

The number m of principal components (i.e. the number of optimization problems to solve) is normally determined beforehand or given implicitly by stopping when a given percentage of the total variance of the original dataset is explained by the principal components identified thus far.

5.3 Hybrid approach

Since the matrix **A** is positive-definite standard quadratic programming techniques can be used to solve the problem without the constraint (5.3). In the hybrid approach proposed in this thesis the metaheuristics generate candidate solutions for **z**. The non-zero components of this vector specify which variables are allowed to be used in the linear combination that defines the sparse principal component. This subordinate optimization problem is of the same form as the original problem, without the cardinality constraint, but is now defined in a subspace of the original space. Therefore, it can be efficiently solved using quadratic programming. In summary, the metaheuristics described in Chapter 2 (GAs, SA and EDA) propose candidate solutions to the combinatorial optimization problem of selecting the subset of the original variables in which the corresponding sparse principal component has loadings different from zero. The subsets of variables proposed by the metaheuristic are then evaluated using the solution of the subordinate optimization problem (5.1)-(5.2) for a fixed **z**.

Two different encodings are considered for the genetic algorithms. The first one is binary encoding. In this type of encoding the candidate set of variables is represented as a binary string of length n (the vector **z** introduced in the previous section). Standard crossover operators, such as uniform crossover, can produce individuals that violate the cardinality constraint (see Subsection 2.3.2). It is therefore necessary to determine how these unfeasible individuals should be handled. Two options are considered. The first one involves including a linear penalty, such as (2.9), in the fitness function. The parameter β, which quantifies the strength of the penalty, is determined in exploratory experiments. Another way of handling unfeasible individuals is applying the repair mechanism outlined in Subsection 2.3.2. A second type of genetic encoding considered is the set representation introduced in Chapter 3. In this representation, an individual is encoded as a subset of size $k \leq K$. As crossover operators for this representation we use RAR (Algorithm 14) and TransRAR (Algorithm 15). Mutation is implemented by interchanging a randomly chosen element in the current subset with an element not included in it.

We also apply SA (Subsection 2.3.1, Algorithm 1) to solve the problem of finding the optimal subset of variables. Every candidate solution is represented as a subset of the target cardinality in a similar way to the GA with set encoding. The neighborhood of a given candidate solution is determined by the operator that interchanges an element in the subset with an element outside of the subset (i.e. the mutation operator in the GA with set encoding).

In another set of experiments the PBIL algorithm (given by the update rule (2.10)) from the EDA family (Subsection 2.3.3) is used to address the combinatorial part of the problem. This algorithm works with a binary encoding. Therefore, individuals of the target cardinality are obtained by sampling using the Algorithm 4.

5.4 Results

To compare the performance of the different methods under consideration, we first apply them to the benchmark problem introduced in d'Aspremont et al. (2008a). Consider the sparse vector $\mathbf{v}$, whose components are

$$v_i = \begin{cases} 1, & \text{if } i \leq 50 \\ 1/(i-50), & \text{if } 50 < i \leq 100 \\ 0, & \text{otherwise} \end{cases} \tag{5.7}$$

A covariance matrix is built from this vector and $\mathbf{U}$, a square matrix of dimensions 150×150 whose elements are $U[0,1]$ random variables

$$\mathbf{A} = \sigma \mathbf{v}\mathbf{v}^T + \mathbf{U}^T\mathbf{U}, \tag{5.8}$$

where $\sigma = 10$ is the signal-to-noise ratio. In this manner the pattern of cardinality is partially masked by noise.

In our experiments the results of SA, binary GAs (with linear penalties and random repair), set GAs with RAR (with $w = 1$) and TransRAR crossover operators, PBIL and DSPCA, an approximate method based on semidefinite programming (d'Aspremont et al. (2007, 2008b)) are compared. SA uses a geometric annealing scheme with $\gamma = 0.9$. The GAs use a population of 50 invididuals. Crossover and mutation are performed with probabilities $p_c = 1$ and $p_m = 10^{-2}$, respectively. PBIL is executed with a population of 400 individuals and $\alpha = 0.1$. In this algorithm, the best 10% of the individuals are used to update the probability distribution. The first sparse principal component is then calculated. For every method involving stochastic search (all except DSPCA), the best out of 5 independent executions of the algorithm is taken.

Figure 5.2 displays the variance explained by the first sparse principal component as a function of its cardinality $K = 1, 2, \ldots, 140$, for all the methods considered. As shown in this figure, the GA using a linear penalty does not obtain high-quality solutions. This is mainly due to the high dimensionality of the problem. PBIL performs slightly better, but is clearly inferior to all the other methods except the GA with linear penalty. Table 5.1 shows the detailed results for cardinality $K = 50$, which is the cardinality of the true hidden pattern. In this table, the largest value of the variance achieved is highlighted in boldface. The success rates, the run-time of the algorithm on an Intel Core Duo machine with 3.00 GHz clock speed and 2 GB RAM and the total number of optimizations performed are shown in this table as well. Note that times for the the DSPCA algorithm are not given because the MATLAB implementation used in d'Aspremont et al. (2008a) cannot be directly compared with the C programs used to obtain the other results. From these results we conclude that the GA with set encoding and RAR ($w = 1$) or TransRAR crossover and the GA with binary encoding and random repair obtain the best results and explain more variance than the solution obtained by DSPCA. The implementation of the GA with the TransRAR operator is clearly faster than the other GAs. SA is very fast and achieves a result that is only slightly worse

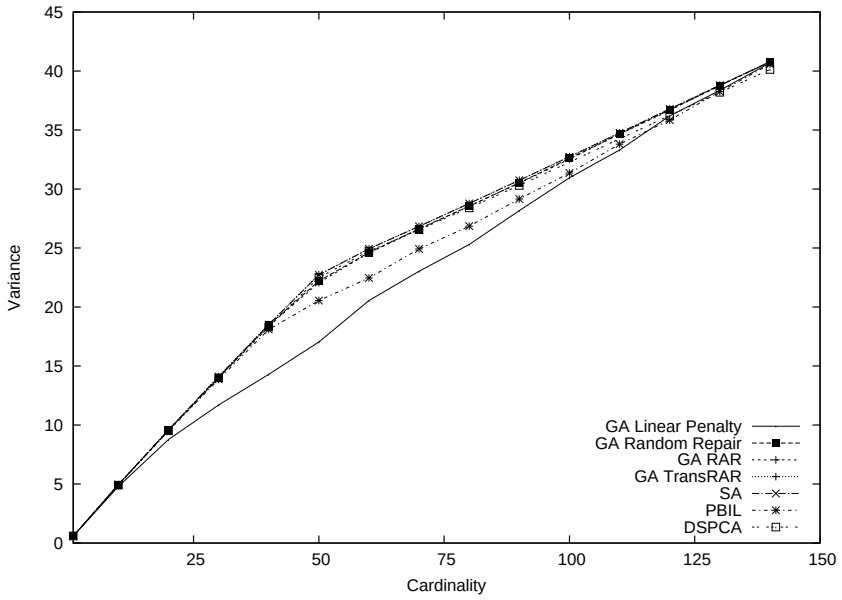


Figure 5.2: Comparison of results for the SPCA problem.

Table 5.1: Results for the GA, SA, EDA and DSPCA approaches in the synthetic problem for $K = 50$

Algorithm	Best variance	Success rate	Time (s)	Optimizations
GA + Linear Penalty	18.8899	0.20	82.11	5142
GA + Random Repair	**22.7423**	0.80	127.75	7924
GA + RAR ($w = 1$)	**22.7423**	0.80	71.11	5162
GA + TransRAR	**22.7423**	1.00	66.75	4544
PBIL	20.4326	1.00	128.73	40800
SA	22.3304	1.00	30.20	10932
DSPCA	22.5001	–	–	–

with a success rate of 100%. PBIL and the GA with binary encoding and linear penalty obtain solutions that are of clearly inferior quality. The reason for their poor performance could be the high dimensionality of the problem. The performance of algorithms of the EDA family rapidly deteriorates with increasing dimensionality (see, for instance, Ruiz-Torrubiano and Suarez (2010)). On the other hand, the sampling of individuals of a given cardinality and the use of a probability distribution in which all marginals are independent could be hindering the performance of PBIL as well. The use of a linear penalty term in the genetic algorithm with binary encoding seems to be detrimental to the efficacy of the search.

In a second experiment with synthetic data, we generate a test instance as proposed in Zou et al. (2006) by defining three hidden factors V_1, V_2 and V_3

$$V_1 \sim \mathcal{N}(0, 290)$$
$$V_2 \sim \mathcal{N}(0, 300)$$
$$V_3 = -0.3V_1 + 0.925V_2 + \epsilon,\ \epsilon \sim \mathcal{N}(0, 1)$$

Next we define the 10 observed variables

$$X_i = V_j + \epsilon_i^j,\ \epsilon \sim \mathcal{N}(0, 1)$$

where $j = 1$ for $i = 1, \ldots, 4$, $j = 2$ for $i = 5, \ldots, 8$ and $j = 3$ for $i = 9, 10$. The ϵ_i^j are independent random variables. We compute the exact covariance matrix and target two principal components for it. Since the hidden factors approximately have the same variance, and each of the first two factors appears in four variables, we expect with $K = 4$ principal components with non-zero loadings in $\{X_1, X_2, X_3, X_4\}$ and $\{X_5, X_6, X_7, X_8\}$, respectively. Table 5.2 shows the results for these types of problems. Non-zero loadings in sparse principal components are underlined. The first principal component explaining roughly 40% of the variance is obtained with equal loadings in the variables X_5, X_6, X_7 and X_8. The second principal component explains nearly the same variance with loadings in the first four variables. This is the same result as the one obtained in d'Aspremont et al. (2004) with DSPCA.

Table 5.3 shows the results obtained using a set GA with RAR crossover in the "pitprops" dataset (Jeffers (1967)). This dataset represents a typical case where the loadings obtained using standard PCA are very difficult to interpret. Following the methodology described in d'Aspremont et al. (2004), 6 sparse principal components are obtained. The 6 unconstraind principal components explain almost 87% of the total variance. The most sparse result in d'Aspremont et al. (2004) is obtained with a maximum of 6 non-zero loadings, explaining 77.3% of the total variance. The cardinality constraint is set to $K = 5$ in our experiments, seeking therefore sparser principal components. The resulting 6 sparse principal components identified capture 78.8% of the total variance. Note that the principal components obtained are less sparse than the ones obtained using SDP. In this case, the sparsity pattern is (6, 2, 3, 1, 1, 1) (a total of 14 non-zero loadings). Every principal component obtained using RAR-GA has cardinality 5. In consequence, there are 25 non-zero loadings. To provide a more direct comparison we attempt to obtain principal components with a similar sparsity pattern as SDP. We found out that using the sparsity pattern (5, 2, 2, 2, 2, 1) the RAR-GA principal components are able to explain 77.7% of the total variance, slightly better than SDP with only a maximum of 5 and a total of 14 non-zero loadings. RAR-GA could not find a feasible solution using exactly the same sparsity pattern as SDP. It is also interesting that with a sparsity pattern of (4, 3, 3, 2, 2, 2) RAR-GA is able to explain 78.3 % of the total variance. This results in a maximum of 4 non-zero loadings, which is even easier to interpret.

Table 5.2: Results for the hidden factors dataset.

	X_1	X_2	X_3	X_4	X_5	X_6	X_7	X_8	X_9	X_{10}	% Var	Var.Acc.
PCA	−0.1157	−0.1157	−0.1157	−0.1157	0.3955	0.3955	0.3955	0.3955	0.4005	0.4005	60.06	60.06
	0.4785	0.4785	0.4785	0.4785	0.1449	0.14489	0.14489	0.14489	−0.0095	−0.0095	39.67	**99.73**
RAR-GA	0.0000	0.0000	0.0000	0.0000	0.5000	0.5000	0.5000	0.5000	0.0000	0.0000	40.91	40.91
($K = 4$)	0.5000	0.5000	0.5000	0.5000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	39.55	**80.46**

Table 5.3: Results for the "pitprops" data set

	X_1	X_2	X_3	X_4	X_5	X_6	X_7	X_8	X_9	X_{10}	X_{11}	X_{12}	X_{13}	% Var	Var.Acc.
PCA	0.4052	0.4069	0.1256	0.1745	0.0571	0.2852	0.3974	0.2892	0.3576	0.3795	−0.0110	−0.1148	−0.1122	32.37	32.37
	0.2157	0.1839	0.5408	0.4553	−0.1706	−0.0157	−0.1885	−0.1868	0.0150	−0.2511	0.2061	0.3443	0.3094	18.27	50.64
	−0.2065	−0.2340	0.1398	0.3500	0.4787	0.4745	0.2591	−0.2516	−0.2074	−0.1183	−0.0716	0.0929	−0.3240	14.49	65.13
	−0.0928	−0.1049	0.0835	0.0627	0.0583	−0.0628	−0.0762	0.2931	0.0958	−0.2009	0.7951	−0.3070	−0.3074	8.55	73.68
	0.0833	0.1139	−0.3494	−0.3578	−0.1800	0.3098	0.2224	−0.1966	0.1089	−0.1591	0.3621	0.5822	−0.0788	7.03	80.71
	0.1191	0.1620	−0.2733	−0.0521	0.6250	0.0496	0.0037	−0.0548	0.0328	−0.1729	0.1744	−0.1747	0.62784	6.27	**86.98**
RAR-GA ($K = 5$)	0.4798	0.4908	0.0000	0.0000	0.0000	0.0000	0.4050	0.0000	0.4228	0.4314	0.0000	0.0000	0.0000	26.20	26.20
	0.0000	0.0000	0.6303	0.6552	0.0000	0.3117	0.0000	0.0000	0.0000	0.0000	0.1450	0.2349	0.0000	16.24	42.44
	0.0000	0.0000	0.0000	0.0000	0.4241	0.3667	0.0000	0.4235	0.0000	0.0000	0.0000	−0.4866	−0.5191	13.38	55.82
	0.0000	0.0000	0.0000	0.0000	0.0000	−0.3552	−0.3203	0.3075	0.3068	0.0000	0.7632	0.0000	0.0000	9.21	65.03
	0.0000	0.0000	−0.1453	0.0000	0.7946	0.0000	0.0000	−0.4866	0.0000	0.0000	0.1960	0.2690	0.0000	7.63	72.66
	0.0000	0.0000	0.0899	0.1304	0.2213	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	−0.6051	0.7482	6.16	**78.82**

5.5 Summary and Conclusions

In this chapter we have presented an application of the hybrid optimization method introduced in Chapter 2 for the SPCA problem. The objective is to obtain principal components with few non-zero loadings. The benefits are twofold: First, sparse principal components are easier to interpret. Second, sparse principal components can be evaluated and stored more efficiently than normal PCs. This can be relevant for problems such as data compression.

Several hybrid metaheuristics have been compared with state-of-the-art methods, such as the DSPCA technique proposed in d'Aspremont et al. (2007). Analyzing the results of this comparison the conclusions are: GAs with random repair and set GAs with RAR or TransRAR perform best in our experiments. Furhtermore, the GA with the TransRAR operator achieves the best-known solution more efficiently than GAs that employ other crossover operators. This underscores the importance of gene transmission in enhancing the effectiveness of the search as discussed in Subsection 3.3.4. Binary GAs using penalties are clearly misled by the penalty term in the fitness function. The GA with binary encoding and a repair mechanism for unfeasible individuals also obtains good results in the problems analyzed. However, the heuristic repair mechanism tends to produce duplicated genotypes, which is detrimental for the search. SA obtains high-quality solutions very quickly. Therefore, it can be used in applications where a quick initial guess is needed (for instance, as a seed for other metaheuristic approaches). The performance of PBIL is rather poor. This is due mainly to the high-dimensionality of the problem: the estimation and sampling from the probability distribution becomes more difficult as the dimensionality of the problem increases.

In the synthetic problems analyzed, the best hybrid methods identify principal components that explain more variance than DSPCA. In the "pitprops" dataset, the set GA with RAR ($w = 1$) finds very sparse PCs that explain a large portion of the variance. As future work, more comparisons with other methods (for instance, the greedy method proposed in Moghaddam et al. (2005) or lasso approaches (Efron et al. (2004))) are needed to further establish the effectiveness of the hybrid approach introduced in this thesis.

CHAPTER 6

INDEX TRACKING BY PARTIAL REPLICATION

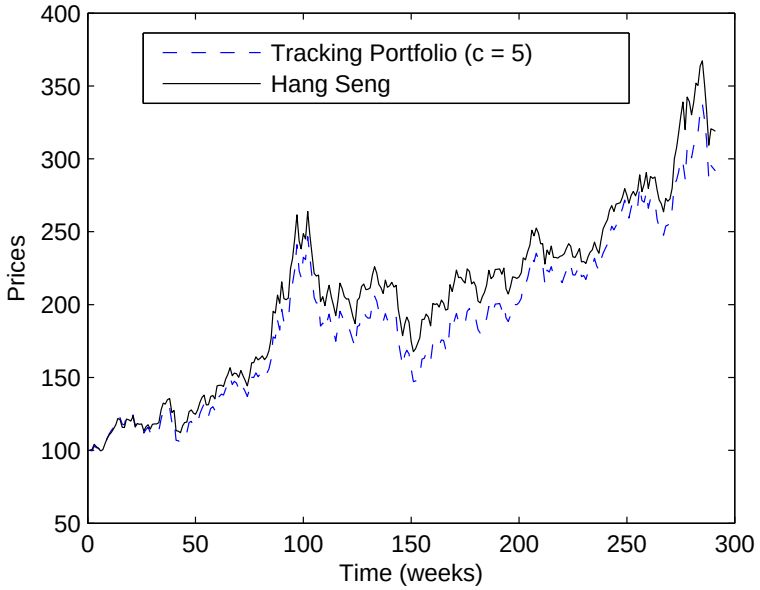
Index tracking consists in reproducing the performance of a stock market index by investing in a subset of the stocks included in the index. The definition of tracking error as the mean squared deviation between the returns of the index and the tracking portfolio leads to the formulation of the problem as a mixed-integer quadratic programming problem. Given a subset of assets, a quadratic solver can be used to find the optimal tracking portfolio that invests only in the selected assets. Several optimization metaheuristics are used to solve the combinatorial problem of identifying the appropriate assets. This hybrid approach allows the identification of quasi-optimal tracking portfolios at a reduced computational cost.

6.1 Introduction

Index tracking consists in constructing a portfolio whose evolution during a specified period is as close as possible to the index that is being tracked. Such a portfolio is called a *tracking portfolio*. The index tracking problem arises in the context of asset management. A rational investor typically wishes to obtain the highest possible performance assuming as little risk as possible. There could be additional restrictions for investment, which may either arise from conditions imposed by the market (minimum investment in a given asset), be the result of a quantitative analysis (e.g. capital concentration constraints from the Black-Litterman model), or reflect expert knowledge and preferences of the investor. This set of market constraints and investor preferences in combination with the expected risk and return of the assets determines the strategy that should be used for fund management.

The goal of the index tracking problem is to build an investment portfolio whose performance is as close as possible to a benchmark financial index, which is used as reference. The problem can be solved exactly by investing on every asset an amount of capital that is proportional to the corresponding weight in the index. In practice, this strategy has the drawback of incurring high initial transaction costs. There is

Figure 6.1: An example of a tracking portfolio for the Hang Seng index that invests in only 5 assets.



an overhead in managing a portfolio that invests in every constituent of the index. Moreover, rebalancing the portfolio can be costly if the composition of the index is revised. An alternative is to design a tracking portfolio that invests only in a reduced set of assets. Figure 6.1 shows an example of a tracking portfolio for the Hang Seng index. Although only 5 products are used, the portfolio can still track the index fairly well. Nonetheless, this partial replication strategy will in general be unable to exactly reproduce the behavior of the index.

A tracking portfolio that invests in a fixed number of assets and closely follows the evolution of the index can be obtained by minimizing a measure of tracking error. In literature several measures of this error have been proposed (Amann and Zimmermann (2001), Lobo et al. (2007), Shapcott (1992), Beasley et al. (2003), Buckley and Korn (1998), Rudolf et al. (1999)). Most of them are based either on correlations between the tracking portfolio and the index returns or on estimations of the variance of the difference between the returns of the index and the returns of the tracking portfolio (Markowitz (1987), Buckley and Korn (1998), Shapcott (1992)). However, measures based solely on the variance of the tracking deviations are insufficient to quantify the tracking quality. As noted in Beasley et al. (2003), if the difference between the returns of the index and those of the tracking portfolio is constant, then the tracking error would be zero. This measure is inaccurate because it does not take the tracking bias into account. In the current investigation the mean squared error for the returns is used as the measure of disagreement between the tracking portfolio and the index which is being tracked. This definition of the tracking error has the advantage of being quadratic. Furthermore it takes the bias of the tracking portfolio into account, so that constant differences are also penalized (Amann and Zimmermann (2001), Beasley et al. (2003), Gilli and Këllezi (2001b)).

In this chapter we report the results of applying the hybrid optimization approach described in Chapter 2 with several combinatorial optimization metaheuristics and a preprocessing pruning step. The metaheuristic is responsible for finding the optimal subset of assets in which the tracking portfolio invests. A quadratic solver (Gill et al. (1991)) is the used to find the optimal composition of the portfolio that invests in the specified subset of assets.

Most previous work on index tracking focuses on finding the portfolio that is optimal using as inputs the recent historical evolution of the assets. Since we are interested in the future tracking performance of the portfolio, it is necessary to assume that recent historic performance is a good predictor of the performance in the near future. In this investigation, the validity of this assumption is tested estimating both in-sample and out-of-sample measures of performance. Using the language of machine learning, the data is partitioned into *training* and *testing* sets. The training data is used to construct the optimal tracking portfolio investing in a subset of the index assets. The performance of this tracking portfolio is then evaluated not only with the data used in the optimization (in-sample performance), but also in an independent test set, which is not used in the optimization (out-of-sample performance). We show that optimal portfolios with respect to the training data need not be optimal in the test set. Conversely, portfolios that are suboptimal on the training data can have a better out-of-sample (generalization) performance (i.e. a lower tracking error) in the test set.

This chapter is organized as follows: In Section 6.2 we formulate the index tracking problem. We then briefly review the methods that have been proposed in literature to address this problem. Section 6.3 describes the hybrid approach and the used metaheuristics used to solve the index tracking problem in this thesis. The results of an empirical evaluation of the methods proposed are discussed in Section 6.4. The conclusions of this chapter are presented in 6.5.

6.2 Index Tracking with Cardinality Constraints

Index tracking has been extensively investigated in literature. In Markowitz (1987) the problem of index tracking was formulated as a multiobjective mean-variance optimization by making some assumptions on the statistical properties of the returns of the index assets. In this reference the cardinality constraints that are the main concern of this thesis were not considered. These types of constraints were investigated in Shapcott (1992). In this work the problem of selecting the optimal subset of assets and the quadratic optimization problem that results once this subset is selected were handled separately. The resulting hybrid evolutionary algorithm is similar to the one investigated in this chapter. In particular, Shapcott (1992) also employed a GA with set encoding and the Random Assorting Recombination (RAR) crossover operator. The main goal of Shapcott (1992) was to compare a random search algorithm to a genetic algorithm with and without migrations in a multiprocessor environment. The objective function used in that work was the variance of the difference between the index returns and the tracking portfolio returns. In the present work, the tracking error considered is the mean squared error

(MSE) between the returns of the index and of the tracking portfolio. Additionally, we take into account other realistic constraints in the model. In particular, we consider bounds on the minimum and maximum amount of investment that can be made in an asset or in a certain class of assets.

The index tracking problem was also addressed in Buckley and Korn (1998) using optimal impulse control techniques. In that reference the problem was formulated as a continuous-time control problem, in which the the random cash flows to and from the portfolio were modeled as a diffusion process. In Gilli and Këllezi (2001b) the problem was addressed by using the threshold accepting (TA) heuristic, which is a deterministic analogue of simulated annealing in which transitions are rejected only when they lead to a deterioration in performance that is above a specified threshold. Evolutionary algorithms with real-valued chromosome representations were used in Beasley et al. (2003). That investigation focused on the effects of transaction costs and portfolio rebalancing in the final performance. In Lobo et al. (2007) the portfolio optimization and index tracking problems were addressed by means of a heuristic relaxation method that consists in solving a small number of convex optimization problems with fixed transaction costs. Hybrid optimization approaches to minimizing the tracking error by partial replication were also investigated in Jeurissen and van den Berg (2005, 2008); Ruiz-Torrubiano and Suárez (2007a).

In this work, index tracking is formulated as a mixed-integer quadratic optimization problem

$$\min_{\mathbf{w},\mathbf{z}} \quad \left[\frac{1}{T}\sum_{t=1}^{T}\sum_{j=1}^{N}\left(w_j r_j(t) - r_t\right)^2\right] \tag{6.1}$$

$$\sum_{i=1}^{N} w_i = 1, \tag{6.2}$$

$$\mathbf{l} \leq \mathbf{A}\cdot\mathbf{w} \leq \mathbf{u} \tag{6.3}$$

$$z_i \in \{0,1\},\ a_i z_i \leq w_i \leq b_i z_i,\ a_i \geq 0,\ b_i \geq 0,\ i = 1,2,\ldots,N \tag{6.4}$$

$$\sum_{i=1}^{N} z_i \leq K, \tag{6.5}$$

where T is the number of timesteps considered, N is the number of constituents of the index, $r_j(t)$ is the return of asset j at time t and r_t is the return of the index at time t. Restriction (6.2) is a budget constraint that ensures that all the capital is invested in the portfolio. Investment concentration constraints are captured by (6.3). These constraints set an upper and a lower bound in a linear combination of assets or asset classes. The $N \times M$ matrix $\mathbf{A}$ contains on each row the coefficients of the M linear inequality constraints, and $M \times 1$ vectors $\mathbf{l}$ and $\mathbf{u}$ represent the lower and the upper bounds on these inequalities, respectively. Expression (6.4) reflects lower and upper bound constraints on individual assets. The binary variables $\{z_1, z_2, \ldots, z_N\}$ indicate whether an asset is included in or excluded from the tracking portfolio. Note that when

$z_i = 0$, the lower and upper bounds for the weight of asset i are both equal to zero, which effectively excludes this asset from the investment. The cardinality constraint is expressed by Eq.(6.5).

In this work, we consider the problem of finding an optimal tracking portfolio for a time period $[1, T]$, which generally corresponds to the recent past in which the evolution of the assets is known. The ultimate goal is to construct portfolios whose tracking performance in the period $[T+1, T+L]$ is also close to optimal. However, the evolution of the assets in this period is not known. Therefore, it is necessary to assume that portfolios that are optimal during the initial time period will also perform well within a future horizon L. Provided that $T + L$ is not too far in the future it is reasonable to assume that the statistical properties of the asset returns in the near future are similar to those in the recent past. As $L \to \infty$, the validity of this hypothesis becomes more questionable and the tracking performance of the portfolio typically deteriorates. Borrowing the concepts used in machine learning, we refer to the returns in the initial time period $[1, T]$ as the training data, and to the returns in time period $[T+1, T+L]$ as the test data. In the experiments carried out we measure the tracking performance of the selected portfolios both in the training set (in-sample performance) and in the test set (out-of-sample performance).

6.3 Hybrid approaches

As outlined in Chapter 2, we apply a combinatorial optimization metaheuristic to determine the optimal subset of assets that should be included in the portfolio. For each candidate solution, a quadratic solver is used to determine the optimal portfolio that invests only in these assets by minimizing (6.1) subject to (6.2)-(6.4). As metaheuristics, we apply genetic algorithms, simulated annealing and estimation of distribution algorithms. We also include a preprocessing step based on block pruning and compare the results using the reduced and the complete set of products.

6.3.1 Genetic algorithms

Genetic algorithms have been introduced in Subsection 2.3.2 and in Algorithm 2. Two different encodings are investigated in this case. A traditional binary encoding with standard crossover and mutation operators is compared to the set representation described in Chapter 3.

Using a binary encoding, a candidate solution is represented as a fixed size binary string. Each bit represents the presence or absence of a given product in the portfolio encoded by the individual. For instance, the chromosome 010011 represents the optimal portfolio investing in products 2, 5 and 6. The problem is how to handle individuals that do not satisfy the cardinality constraint. Such individuals can be generated by the application of standard crossover such as N-point and uniform crossover. A possible approach is to use penalty functions, like the linear penalty given in Eq. (2.9). In this case, the strength of the penalty (parameter β) must be estimated through exploratory

experiments. The other possibility is to use repair mechanisms to transform unfeasible individuals into individuals that satisfy the cardinality constraint. The heuristic repair applied in this cases consists in performing an optimization without the constraints (6.4) and (6.5). The products are then ranked according to their weights. We then eliminate from the replicating portfolio the products with lower weights until the cardinality constraint is satified.

We can also use genetic algorithms using a representation in which an individual encodes a subset of the specified cardinality. In this case, each chromosome represents the optimal portfolio investing in only those products present in the subset. For instance, the individual $\{1, 2, 4\}$ represents the optimal portfolio investing only in products 1, 2 and 4. To implement mutation, one simply exchanges a product in the portfolio with another product that is not in the portfolio. This operator preserves the cardinality of the individual. As crossover operator, RAR (Algorithm 14) and TransRAR (Algorithm 15) are used.

6.3.2 Simulated Annealing

The simulated annealing metaheuristic was described in Subsection 2.3.1 (Algorithm 1). Candidate solutions are represented by sets of fixed cardinality. The neighborhood operator $\mathcal{N}(\cdot)$ is defined as the interchange of one product in the portfolio with another one not included in it. The free energy function is the MSE of the optimal tracking portfolio that invests only in the products included in the set that defines the investment universe for the candidate solution.

6.3.3 Estimation of Distribution Algorithms

EDAs are introduced in Section 2.3.3. For this investigation, we used the discrete version of the PBIL algorithm. The sampling of individuals is performed with Algorithm 4.

6.3.4 Block pruning

We apply the block pruning heuristic as described in Section 2.4 and Algorithm 5. The relaxed optimization problem that is used to determine which assets should be excluded from the investment universe consists in minimizing (6.1) subject to (6.2)-(6.4). In the next section, which presents the results of the empirical evaluation of the different methods proposed, we perform a series of experiments to determine to what extent reducing the universe of assets in which the optimization takes place affects the in-sample and the out-of-sample performance of the replicating portfolio.

6.4 Empirical evaluation

In the current investigation publicly available benchmark data from the OR-Library (Beasley (1990)) is used to compare the optimization techniques described in the previous

section. The historical returns of five world market indices are used in the experiments: Hang Seng (31 assets), DAX (85 assets), FTSE (89 assets), S&P (98 assets) and Nikkei (225 assets). For every index, the time series of 290 weekly returns for the index and for its constituents are available. From these data, the first 145 values are used to build a tracking portfolio that includes a maximum of $K = 10$ assets. The last 145 values are used to measure the out-of-sample tracking error. The population sizes are 350 for the GAs and 1000 for PBIL. A steady state substitution scheme with no elitism is used in each generation. The probabilities of crossover and mutation are set to $p_C = 1.0$ and $p_M = 0.01$, respectively. For SA, a geometrical cooling scheme with $\alpha = 0.9$ is used. Table 6.1 presents a summary of the experiments performed. The best out of 5 executions of the different optimization methods is reported. The best in-sample MSE and the corresponding out-of-sample MSE are given in the next two columns. The fraction of times in which the algorithm obtains the best solution is shown in the column "Sucess rate". We also give the execution time on an Intel Core Duo machine with 3.00 GHz clock-speed and 2 GB RAM and the number of quadratic optimizations performed in the last two columns.

The set GA with the TransRAR crossover operator obtains the best overall results. The set GA with the RAR operator ($w = 1$) has comparable performance except in the replication of the Nikkei, which is the index with the largest number of constituents. Note that the execution time of TransRAR is always lower than that of RAR. PBIL has also good performance, but the computational costs are higher than for the other algorithms. In fact, the algorithm reached the maximum number of optimizations established without having converged. The results of SA and GA with binary encoding and linear penalty are suboptimal in all but the simplest problems (Hang Seng and DAX). They also exhibit low success rates. Nonetheless, the very low execution times of SA for all problems except Nikkei are noteworthy. In this last case, all the algorithms seem to have difficulties in converging. This is due to the presence of highly correlated assets which mislead the algorithms towards suboptimal solutions. An example of such correlated series is shown in Figure 6.2 in the Nikkei case.

In all the problems investigated, the out-of-sample error is typically larger than the in-sample error, but of the same order of magnitude. Note that there are cases in which a replicating portfolio that has poor in-sample performance exhibits in turn better out-of-sample performance. For instance, the in-sample MSE of the binary GA with linear penalty in the problem of replicating the FTSE index is rather large. By contrast, it has the best out-of-sample MSE. In most other cases, the best in-sample result also corresponds to the best out-of-sample result (for instace, the GA with the TransRAR crossover operator for the Nikkei problem).

We now apply block pruning as a preprocessing step for all the algorithms investigated. In block pruning, we eliminate those products that have a weight under $l_i/2$ in an unconstrained optimization. Note that this is equivalent to eliminating from the optimization universe assets whose weight in the replicating portfolio is small. This is reasonable if our goal is to obtain sparse solutions. Table 6.2 shows the resulting number of products after applying block pruning on every problem instance. The amount of pruning depends on the structure of the particular index considered: While in the S&P

Table 6.1: Results for the GA, SA and EDA approaches in the index tracking problem.

Algorithm	Index	Best MSE In-Sample	MSE Out-of-Sample	Success rate	Time (s)	Number opts.
GA Linear Penalty	Hang Seng	$\mathbf{1.3462 \cdot 10^{-5}}$	$2.0575 \cdot 10^{-5}$	0.60	4.15	51509
	DAX	$\mathbf{8.0837 \cdot 10^{-6}}$	$7.4824 \cdot 10^{-5}$	0.20	13.69	144868
	FTSE	$2.7345 \cdot 10^{-5}$	$5.3148 \cdot 10^{-5}$	0.20	17.66	158465
	S&P	$1.7974 \cdot 10^{-5}$	$5.2898 \cdot 10^{-5}$	0.20	36.89	311008
	Nikkei	$2.0061 \cdot 10^{-5}$	$1.0707 \cdot 10^{-4}$	0.20	123.28	1015774
GA Random Repair	Hang Seng	$\mathbf{1.3462 \cdot 10^{-5}}$	$2.0575 \cdot 10^{-5}$	1.00	5.92	81690
	DAX	$\mathbf{8.0837 \cdot 10^{-6}}$	$7.4824 \cdot 10^{-5}$	1.00	18.89	231840
	FTSE	$\mathbf{2.1836 \cdot 10^{-5}}$	$8.0091 \cdot 10^{-5}$	0.40	21.20	255820
	S&P	$\mathbf{1.6573 \cdot 10^{-5}}$	$5.5457 \cdot 10^{-5}$	0.20	47.02	508313
	Nikkei	$1.8255 \cdot 10^{-5}$	$6.9574 \cdot 10^{-5}$	0.20	170.62	1664696
GA RAR ($w = 1$)	Hang Seng	$\mathbf{1.3462 \cdot 10^{-5}}$	$2.0575 \cdot 10^{-5}$	1.00	4.67	51513
	DAX	$\mathbf{8.0837 \cdot 10^{-6}}$	$7.4824 \cdot 10^{-5}$	1.00	14.17	124717
	FTSE	$\mathbf{2.1836 \cdot 10^{-5}}$	$8.0091 \cdot 10^{-5}$	0.40	18.83	156456
	S&P	$\mathbf{1.6573 \cdot 10^{-5}}$	$5.5457 \cdot 10^{-5}$	0.20	42.31	311002
	Nikkei	$1.8917 \cdot 10^{-5}$	$8.1057 \cdot 10^{-5}$	0.20	175.34	1015766
GA TransRAR	Hang Seng	$\mathbf{1.3462 \cdot 10^{-5}}$	$2.0575 \cdot 10^{-5}$	1.00	3.67	50301
	DAX	$\mathbf{8.0837 \cdot 10^{-6}}$	$7.4824 \cdot 10^{-5}$	1.00	9.45	121792
	FTSE	$\mathbf{2.1836 \cdot 10^{-5}}$	$8.0091 \cdot 10^{-5}$	0.60	12.22	152348
	S&P	$\mathbf{1.6573 \cdot 10^{-5}}$	$5.5457 \cdot 10^{-5}$	0.80	26.09	302868
	Nikkei	$\mathbf{1.7969 \cdot 10^{-5}}$	$6.4711 \cdot 10^{-5}$	0.20	84.98	989744
SA	Hang Seng	$\mathbf{1.3462 \cdot 10^{-5}}$	$2.0575 \cdot 10^{-5}$	0.40	1.12	19342
	DAX	$\mathbf{8.0837 \cdot 10^{-6}}$	$7.4824 \cdot 10^{-5}$	0.40	1.73	27101
	FTSE	$2.3951 \cdot 10^{-5}$	$7.0007 \cdot 10^{-5}$	0.20	1.44	24657
	S&P	$1.6781 \cdot 10^{-5}$	$4.7347 \cdot 10^{-5}$	0.20	1.97	29764
	Nikkei	$2.1974 \cdot 10^{-5}$	$1.0719 \cdot 10^{-4}$	0.20	95.00	1476549
PBIL	Hang Seng	$\mathbf{1.3462 \cdot 10^{-5}}$	$2.0575 \cdot 10^{-5}$	1.00	167.04	2010000
	DAX	$\mathbf{8.0837 \cdot 10^{-6}}$	$7.4824 \cdot 10^{-5}$	1.00	199.28	2010000
	FTSE	$\mathbf{2.1836 \cdot 10^{-5}}$	$8.0091 \cdot 10^{-5}$	1.00	195.31	2010000
	S&P	$1.6781 \cdot 10^{-5}$	$4.7347 \cdot 10^{-5}$	0.60	314.77	2010000
	Nikkei	$1.9510 \cdot 10^{-5}$	$7.4572 \cdot 10^{-5}$	0.20	222.86	2010000

index only 21.4 % of the assets are eliminated, the heuristic prunes 68 % of the products in the Nikkei index.

The results with pruning are listed in Table 6.3. In this table, the last column shows the speed-up factors obtained by dividing the execution time without pruning by the execution time after pruning. These speed-up factors range from 1.61 (TransRAR, FTSE) to 77.87 (SA, Nikkei). The largest speed-up factors are achieved by PBIL, although in this case the detrimental effect of pruning seems to have the largest influence. As can be seen from these results, both set GAs with RAR and TransRAR achieve the best known results for all problems except for the problem of replicating the Nikkei index. This is an indication that, in this problem, the pruning heuristic eliminates some products from the optimization that are actually present in the best known solution. Nonetheless, the quality of the obtained solutions remains almost unchanged in all other

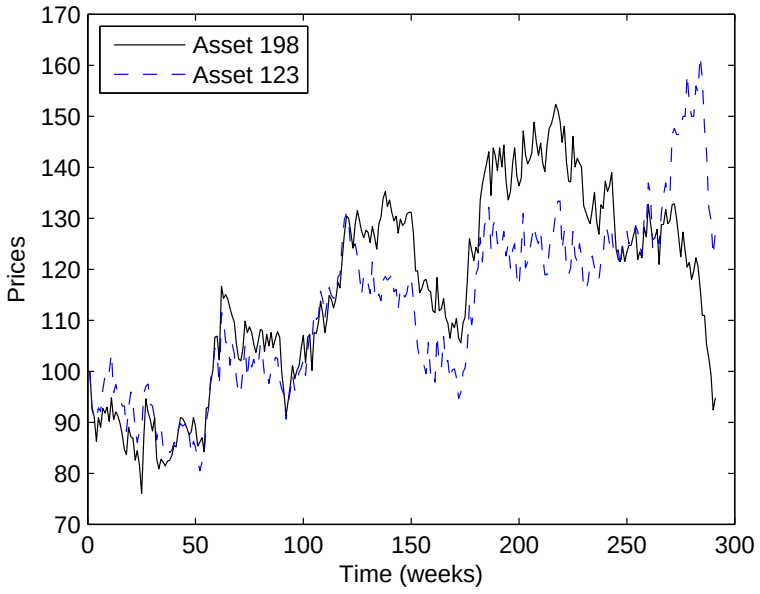
Table 6.2: Number of assets that are kept after block pruning

Index	N before CP	N after CP	% Pruning
Hang Seng	31	23	25.8 %
DAX	85	45	47.1 %
FTSE	89	68	23.6 %
S&P	98	77	21.4 %
Nikkei	225	72	68.0 %

Table 6.3: Results for the GA, SA and EDA approaches in the index tracking problem with block pruning.

Algorithm	Index	Best MSE In-Sample	MSE Out-of-Sample	Success rate	Time (s)	Speed-up Factor
GA Linear Penalty	Hang Seng	$1.5323 \cdot 10^{-5}$	$1.9840 \cdot 10^{-5}$	0.20	2.53	1.64
	DAX	$9.5296 \cdot 10^{-6}$	$7.5745 \cdot 10^{-5}$	0.20	2.91	4.70
	FTSE	$2.2745 \cdot 10^{-5}$	$7.8406 \cdot 10^{-5}$	0.20	7.48	2.36
	S&P	$1.8026 \cdot 10^{-5}$	$6.4064 \cdot 10^{-5}$	0.20	15.72	2.35
	Nikkei	$2.3845 \cdot 10^{-5}$	$9.6544 \cdot 10^{-5}$	0.20	24.11	5.11
GA Random Repair	Hang Seng	$\mathbf{1.3462 \cdot 10^{-5}}$	$2.0575 \cdot 10^{-5}$	1.00	1.52	3.89
	DAX	$\mathbf{8.0837 \cdot 10^{-6}}$	$7.4824 \cdot 10^{-5}$	1.00	4.56	4.14
	FTSE	$2.4408 \cdot 10^{-5}$	$6.4976 \cdot 10^{-5}$	0.20	10.59	2.00
	S&P	$\mathbf{1.6573 \cdot 10^{-5}}$	$5.5457 \cdot 10^{-5}$	0.60	21.48	2.19
	Nikkei	$2.3114 \cdot 10^{-5}$	$7.8567 \cdot 10^{-5}$	0.40	33.11	5.15
GA RAR ($w = 1$)	Hang Seng	$\mathbf{1.3462 \cdot 10^{-5}}$	$2.0575 \cdot 10^{-5}$	1.00	1.23	3.80
	DAX	$\mathbf{8.0837 \cdot 10^{-6}}$	$7.4824 \cdot 10^{-5}$	1.00	4.20	3.37
	FTSE	$\mathbf{2.1836 \cdot 10^{-5}}$	$8.0091 \cdot 10^{-5}$	0.20	8.84	2.13
	S&P	$\mathbf{1.6573 \cdot 10^{-5}}$	$5.5457 \cdot 10^{-5}$	0.60	18.17	2.33
	Nikkei	$2.1584 \cdot 10^{-5}$	$1.0225 \cdot 10^{-4}$	0.20	27.36	6.41
GA TransRAR	Hang Seng	$\mathbf{1.3462 \cdot 10^{-5}}$	$2.0575 \cdot 10^{-5}$	1.00	1.13	3.25
	DAX	$\mathbf{8.0837 \cdot 10^{-6}}$	$7.4824 \cdot 10^{-5}$	0.40	2.95	3.20
	FTSE	$\mathbf{2.1836 \cdot 10^{-5}}$	$8.0091 \cdot 10^{-5}$	0.20	7.57	1.61
	S&P	$\mathbf{1.6573 \cdot 10^{-5}}$	$5.5457 \cdot 10^{-5}$	0.20	15.72	1.66
	Nikkei	$2.3114 \cdot 10^{-5}$	$7.8567 \cdot 10^{-5}$	0.80	25.45	3.34
SA	Hang Seng	$\mathbf{1.3462 \cdot 10^{-5}}$	$2.0575 \cdot 10^{-5}$	0.80	0.31	3.61
	DAX	$\mathbf{8.0837 \cdot 10^{-6}}$	$7.4824 \cdot 10^{-5}$	0.20	0.39	4.43
	FTSE	$\mathbf{2.1836 \cdot 10^{-5}}$	$8.0091 \cdot 10^{-5}$	0.20	0.87	1.65
	S&P	$1.6816 \cdot 10^{-5}$	$4.7808 \cdot 10^{-5}$	0.40	0.92	2.14
	Nikkei	$2.3083 \cdot 10^{-5}$	$8.9584 \cdot 10^{-5}$	0.20	1.22	77.87
PBIL	Hang Seng	$\mathbf{1.3462 \cdot 10^{-5}}$	$2.0575 \cdot 10^{-5}$	1.00	3.39	49.27
	DAX	$8.2111 \cdot 10^{-6}$	$7.7601 \cdot 10^{-5}$	0.40	3.77	52.86
	FTSE	$2.1932 \cdot 10^{-5}$	$7.8699 \cdot 10^{-5}$	0.20	13.66	14.30
	S&P	$1.6782 \cdot 10^{-5}$	$4.7347 \cdot 10^{-5}$	0.40	13.91	22.63
	Nikkei	$2.9726 \cdot 10^{-5}$	$8.9375 \cdot 10^{-5}$	0.20	5.31	41.97

Figure 6.2: Highly correlated assets contained in the Nikkei index.



cases. In general, pruning does not improve the out-of-sample MSE, except in a few cases in which the solutions have high in-sample MSE, but low out-of-sample MSE (GA Linear, Hang Seng).

6.5 Summary and Discussion

In this chapter, the problem of financial index tracking by partial replication is solved by means of the hybrid optimization approach presented in Chapter 2. In particular, GAs, SA and PBIL are used to address the combinatorial component of the problem. The use of a quadratic measure of tracking error allows the use of a quadratic solver to evaluate the candidate solutions proposed by the metaheuristics. From the methods considered, the best results are achieved by the GAs with set encoding that employ the TransRAR crossover operator. Nonetheless using RAR crossover instead also leads to high-quality solutions. Using a binary encoding with either a penalty term in the fitness function or complemented by a repair mechanism, generally obtains poorer results than the other algorithms. SA can obtain high quality solutions very quickly. It can be therefore used to obtain a quick initial guess or a seed for other algorithms. The PBIL algorithm has difficulties to converge, and has the longest execution times among all the techniques compared.

We can also apply dimensionality reduction techniques to eliminate from the investment universe assets that are not likely to be included in the optimal solution. After applying a block pruning heuristic, both set GAs with RAR and TransRAR are still able to obtain the best known solution in all cases except in the Nikkei problem. This indicates that, in most cases, the heuristic can be used to reduce the dimensionality of the problem without causing significant deterioration of the final solution. In the Nikkei

problem the performance of the solution that is obtained after pruning is only slightly inferior to the best known solution. The speedup factors obtained by pruning are in general large, especially for the PBIL algorithm.

With some noteworthy exceptions, the in-sample and out-of-sample performances of the replicating portfolios are correlated: portfolios with low in-sample MSE also have low out-of-sample MSE. Therefore, in the problems investigated, the tracking performance of the portfolio in a given period seems to be a good predictor for the tracking performance in the subsequent period.

CHAPTER 7

OPTIMAL PORTFOLIO SELECTION

The problem of selecting an investment portfolio that is optimal has attracted much attention from researchers in finance and in optimization. Imposing constraints that limit the number of products that can be included in the portfolio is an useful strategy that facilitates the management of the portfolio. As will be shown in this chapter cardinality constrained optimal portfolios tend to be more stable, robust and have better out-of-sample performance than unconstrained portfolios. However, including cardinality constraints transforms portfolio selection into a mixed-integer quadratic problem, which can be difficult to solve. Therefore, it is necessary to develop efficient algorithms to find near-optimal solutions at a reasonable computational cost that can be used by practitioners in real-world applications. In this thesis we propose to combine metaheuristics, such as genetic algorithms, simulated annealing or estimation of distributions algorithms, and standard quadratic programming algorithms to address the problem. The metaheuristics can be used to address the combinatorial part of the problem, which consists in selecting the products in which the portfolio invests. The candidate solutions generated by the metaheuristic are evaluated using the solution of a subordinate optimization problem. This subordinate problem is a portfolio optimization in the reduced investment universe specified by the candidate solution, but now without the cardinality constraints. Because of the cardinality constraints, many of the products in the original universe considered for investment are not included in the final portfolio. Therefore, one can think of using pruning techniques that identify and eliminate such products in a preprocessing step. The dimensionality reduction obtained by applying these pruning heuristics greatly improves the computational efficiency of the optimization. Furthermore, provided that the application of these heuristics does not eliminate products whose presence in the final portfolio is crucial for the optimality of the portfolio, pruning has no or small impact on the quality of the solutions obtained. Finally, the problem of portfolio selection is extended to take into account transaction costs.

7.1 Introduction

The selection of optimal investment portfolios is a problem of great interest in the area of quantitative finance and has attracted much attention in the scientific community (Chang et al. (2000), Bienstock (1995), Crama and Schyns (1999), Gilli and Këllezi (2001a), Lobo et al. (2007), Schaerf (2002), Moral-Escudero et al. (2006), DiGaspero et al. (2011)). The problem is of practical importance for investors, whose objective is to allocate capital in an optimal manner among assets while respecting some investment restrictions. Several models and methods for solving this problem have been proposed, mostly within the classical framework developed by H. Markowitz in 1952 (Markowitz (1952)). In the work of Markowitz, which sets the foundations of modern portfolio optimization theory, the returns of the products that are considered for investment are modeled as random variables. The profit of the investment is the expected portfolio return. The risk measure is the variance of the portfolio returns. Therefore, portfolio selection is a multiobjective optimization task with two opposing goals: the maximization of profit and the minimization of risk. Since these cannot be simultaneously achieved, the problem is to identify the *efficient frontier*, which is defined as the set of solutions that are Pareto optimal. A portfolio is in the efficient frontier if for a given risk no other portfolio with a larger expected return can be built by modifying the investment weights. By duality, this same portfolio is the one that minimizes risk for that value of expected return.

Without further constraints the identification of portfolios in the efficient frontier is a quadratic optimization problem that can be readily solved by standard numerical techniques (Gill et al. (1991)). A difficulty with this approach is that the portfolios resulting from this unconstrained optimization typically invest small amounts in large numbers of products to take advantage of the benefits of diversification and reduce the overall risk. This type of investment strategy can be difficult to implement in practice: Portfolios composed of a large number of assets are difficult to manage and may incur high transaction costs. To address this shortcoming, several restrictions can be imposed on the allocation of capital among assets. One can limit the total number of assets in the final portfolio or impose lower and upper bounds to the proportion of capital invested in each product. These constraints make the problem difficult to solve by standard optimization techniques. In fact, the problem becomes NP-Complete (Moral-Escudero et al. (2006); Tabata and Takeda (1995)). Nonetheless, heuristic optimization methods can be applied to identify near-optimal solutions at a reasonable computational cost. Several general optimization techniques have been proposed to solve this problem: evolutionary algorithms (Michalewicz (1996), Goldberg (1989), Chang et al. (2000)), simulated annealing (Kirkpatrick et al. (1983), Crama and Schyns (1999)), tabu search (Glover (1986), Schaerf (2002)), local search (DiGaspero et al. (2007), DiGaspero et al. (2011)), and other. In this chapter we propose several memetic (Moscato and Cotta (2003)) techniques to solve the problem. In memetic algorithms, prior knowledge of the problem at hand is used to combine different algorithmic techniques to obtain high-quality solutions with a moderate computational effort (Jeong et al. (2009)). We apply these ideas to construct a variety of portfolio optimization algorithms by combining either a

genetic algorithm with binary and set encoding (Moral-Escudero et al. (2006), Radcliffe (1993)), simulated annealing or various estimation of distribution algorithms (Larrañaga and Lozano (2002)), quadratic programming (Gill et al. (1991)) and specially designed pruning heuristics that effectively reduce the size of the search space. The problem of finding the optimal subset of assets for investment is addressed by the metaheuristic technique. The continuous optimization problem of finding the optimal weights given this subset is solved using a standard quadratic optimization algorithm. Different variants of EDAs are investigated in this chapter, including models in which all the variables are assumed to be statistically independent (Muehlenbein (1998), Baluja (1994)), and models in which more complex interactions between the variables are allowed (Larrañaga et al. (2001)). The evaluation and comparison of the different methods are carried out on publicly available benchmark data from the OR-Library (Beasley (1990)), which includes data from assets included in major world stock indices. The experiments performed show that efficient and accurate solutions are obtained when special pruning heuristics are applied. Pruning attempts to identify and eliminate from the universe of assets available for investment those products that are not likely to appear in the optimal portfolio. The use of pruning heuristics not only improves the results of the hybrid methods based on EDA, but also enhances the efficiency of SA and of genetic algorithms with a set representation (Moral-Escudero et al. (2006)).

Finally, we present the results of a hybrid metaheuristic method designed to solve the multi-period portfolio selection problem including transaction costs and turnover constraints. Piecewise linear transaction costs are non-differentiable. Therefore standard quadratic programming algorithms are not appropriate to solve the problem. Turnover restrictions, which partition the search space into disjoint feasible regions for each investment decision (hold the actual position, purchase or sell a minimum amount), introduce additional combinatorial complexity in the problem. We adapt the set encoding proposed for the single-period formulation of the problem by including an additional attribute for each element in the subset, which reflects the trading decision made for that particular asset (buy, hold or sell). As will be shown in the empirical evaluation (using public available data compiled by Fama and French[1]), both transaction costs and cardinality constraints have have a regularization effect which results in stable, robust portfolios that have in general good out-of-sample performance.

This chapter is organized as follows: Section 7.2 summarizes previous work on the portfolio selection problem. In Section 7.3, the optimization model with cardinality constraints and hybrid methods for its solution, as well as empirical results are presented. The inclusion of transaction costs is considered in Section 7.4, where a hybrid approach with an extended set encoding is used in an extensive empirical evaluation to solve the problem. Finally, the discussion of the results and some conclusions are presented in Section 7.5.

[1]Data available from http://mba.tuck.dartmouth.edu/pages/faculty/ken.french/data_library.html

7.2 Previous Work

There is a large amount of literature on portfolio selection considering both the single period and the multi-period formulations of the problem. The single period problem consists in building a portfolio of fixed composition whose performance is optimal in a given investment period. It can be formulated as an optimization that uses as input either recent historical data or implicit market estimates of expected returns and covariances among returns. The multi-period problem consists in finding the optimal investment strategy to manage a portfolio that can be rebalanced at specified times during the life of the investment.

We first consider the single period formulation. Branch-and-Bound techniques were used in Bienstock (1995) to solve the problem exactly. Despite the fact that these techniques improved the efficiency of the search, the time needed to find the global optimum grows exponentially with the number of assets available for investment. Genetic algorithms have also been used to address this problem. In Chang et al. (2000), the performance of GAs was compared to simulated annealing (SA) (Kirkpatrick et al. (1983)) and to tabu search (Glover (1986)). In this work the best results were obtained by pooling the results of the different heuristics. In Crama and Schyns (1999) simulated annealing was used to explore the space of real-valued asset weights. Tabu search and other local search techniques were employed in Schaerf (2002). In this reference, a continuous encoding was used and the neighborhood operators were designed to obtain feasible portfolios at all times. In Huang and Jane (2009), a hybrid portfolio selection and forecasting approach was designed by combining ARX, Grey systems and rough set theory. The authors predicted stock market trends and, making use of these predictions, selected an investment portfolio based on automatically generated investment rules. Several factors reflecting company wealth information were also considered in their approach. In Streichert et al. (2004) and Streichert and Tamaka-Tamawaki (2006) Multi-Objective Evolutionary Algorithms (MOEAs) were used to address the problem. These algorithms employed a hybrid encoding instead of a pure continuous one. Heuristic operators were applied to repair unfeasible individuals generated in the course of the evolution. The impact of Lamarckism and of local search improvements were also analyzed in this work. The authors concluded that the hybrid encoding used improved the overall performance of the algorithm. A hybrid optimization approach to the problem of tracking a stock index by partial replication is also proposed in Jeurissen and van den Berg (2005, 2008); Shapcott (1992). In these articles, the discrete optimization problem of calculating the optimal subset of assets to be included in the portfolio and the problem of determining the weight of the selected assets by minimizing the tracking error were handled separately. The subset selection problem was solved using a genetic algorithm with different mutation and crossover operators that maintain the cardinality constraint. In particular, the set representation and the RAR crossover operators introduced by Shapcott (1992) are used in the current investigation as well. A similar decomposition of the problem in a continuous and a combinatorial optimization was recently used in DiGaspero et al. (2011, 2007). In these articles, the authors used a hybrid approach where the combinatorial problem of searching for the optimal subset of assets

was carried out by a local search algorithm, and the portfolio weights were calculated by quadratic programming once the subset determined by the candidate solution was fixed. The comparison of genetic algorithms that use a standard binary representation, in which the violations of the cardinality constraint are either penalized or avoided by repairing unfeasible individuals, and a genetic algorithm based on set representations of fixed cardinality, as in Shapcott (1992), was performed in Moral-Escudero et al. (2006); Ruiz-Torrubiano and Suárez (2007b). These articles showed that using penalty terms in the cost function is not an effective method for handling unfeasible individuals when a binary encoding is used. In contrast, algorithms in which the generated individuals always remain feasible either by repair techniques, or by using specially designed operators that preserve the cardinality of the candidate solutions, generally perform well. In the current approach, we extend and improve this work by considering additional metaheuristics and by exploiting the advantages of pruning techniques that effectively reduce the size of the universe of assets where the search is conducted.

For the multi-period formulation of the problem it is important to take into account the transaction costs incurred when rebalancing the portfolio. An extension of the Markowitz model that includes transaction costs was proposed in Pogue (1970). In that reference, transaction costs resulting from both brokerage fees and market illiquidity were considered. In Perold (1984) the standard complementary pivot algorithm of Markowitz (Markowitz (1987)) was extended to take into account concave piecewise linear transaction costs, turnover constraints and approximate minimum trading size constraints. A non-linear programming technique was applied by Yoshimoto (Yoshimoto (1996)) to a portfolio selection problem with V-shaped transaction costs. That work showed that ignoring the transaction costs can result in inefficient portfolios. In contrast, considering transaction costs leads to the selection of portfolios that are generally more stable. Turnover or minimum trading size constraints were not considered in that investigation. In Lobo et al. (2007) the authors addressed non-convex portfolio optimization problems with transaction costs that include a fixed fee. They proposed an iterative heuristic algorithm to approximate the optimal portfolio by solving a series of convex relaxations of the original problem. The resulting portfolio was suboptimal but had the advantage of being an upper bound of the optimal solution. They also showed that, in real-world cases, the bound is generally tight, even for large problems. The same approach can be used also for index tracking. In Mitchell and Braun (2004), the authors proposed to rescale the objective function by the amount of wealth invested after transaction costs are subtracted. The resulting model was a fractional programming problem that can be addressed by convex optimization techniques. Best and Hlouskova (Best and Hlouskova (2008)) applied a modified quadratic programming active set algorithm to solve the mean-variance problem with transaction costs in an investment universe of size N. The transaction costs can be accounted for by defining a $3N$-dimensional optimization problem with $3N$ additional constraints. To reduce the complexity associated with the increase of the dimensionality of the optimization space, they proposed an algorithm that works in N-dimensions, in which the transaction costs were accounted for implicitly rather than explicitly. No cardinality or turnover constraints were considered in that work.

Several authors have focused on obtaining portfolios with better out-of-sample performance by imposing additional constraints or modifications in the portfolio optimization model. Specifically, DeMiguel et al. (2009a) propose to constrain the norm of the vector of portfolio weights. Robust estimation techniques were used in DeMiguel and Nogales (2009) to obtain portfolios whose weights have small fluctuations over time. Santos et al. (2012) addressed the problem of finding portfolios whose capital requirements are minimized by imposing constraints on the number of VaR violations in a given period of time. In Brodie et al. (2009) optimal portfolio selection was formulated as a regularized regression problem. The objective function included a penalty term proportional to the L_1 norm of the vector of portfolio weights $||\mathbf{w}||_1$ as in *lasso regression* (Tibshirani (1996)). As a consequence of the properties of the L_1 norm, when the weight of this type of penalty term is sufficiently large, the portfolios obtained are sparse and invest in only a subset of the assets available for investment. In general, these sparse portfolios are more stable than minimal variance portfolios obtained without the L_1-norm penalty. In the current work we consider an extension of this idea and include in the objective function an L_1 term that penalizes the differences between the portfolio weights before and after rebalancing $\left|\left|\mathbf{w} - \mathbf{w}^{(0)}\right|\right|_1$, instead. This form of the penalty is similar to the term that appears in the objective function when transaction costs are considered. The inclusion of this type of penalties leads to the selection of portfolios whose composition is more stable over time. As a result, they are easier to manage, have lower rebalancing costs and generally exhibit good out-of-sample performance.

7.3 Portfolio Selection with Cardinality Constraints

7.3.1 The Markowitz Mean-variance Model

The work published by H. Markowitz in 1952 (Markowitz (1952)) is considered as the foundation of modern portfolio theory. It provides a mathematical framework for optimal portfolio selection. The problem consists in how to allocate a fixed amount of capital among different assets, whose evolution is subject to uncertainty, in such a way that the expected return of the investment is maximized and the risk associated with it is minimized. The approach adopted by Markowitz is to model the asset returns as a random vector whose distribution can be fully characterized by a vector of expected returns and a covariance matrix. If the returns of the assets were deterministic (zero variance) the optimal solution would be to invest only in the asset with the highest expected return. In the presence of uncertainty this solution need not be optimal. The investor may prefer to settle for a lower expected return provided that the uncertainty in the investment is reduced as well. This establishes a rational basis for understanding the potential benefits of diversification.

Let N be the number of assets in $\mathcal{U}$, the universe of products available for investment. Consider the prices of each market product at time instant t,

$$\{S_i(t)\}_{i=1}^{N}$$

The Markowitz framework considers the evolution of the portfolio during the period $[t, t+1)$. The composition of the portfolio $\{x_i\}_{i=1}^N$, where x_i is the units of product i, is determined at the beginning of this period and held constant during the interval considered. The evolution of the value of the portfolio during this period is

$$P(\tau) = \sum_{i=1}^{N} x_i S_i(\tau), \tau \in [t, t+1). \tag{7.1}$$

Since the composition of the portfolio is fixed, the changes in portfolio value are due to changes in the market value of the assets of which it is composed (i.e., market capitalization). The return of the portfolio in the interval $[t, t+1)$, is

$$r_P = \frac{P(t+1) - P(t)}{P(t)} = \sum_{i=1}^{N} w_i r_i = \mathbf{w}^{\mathrm{T}} \cdot \mathbf{r},$$

where $\mathbf{r}^{\mathrm{T}} = (r_1, r_2, \ldots, r_N)$ is the transposed vector of asset returns

$$r_i = \frac{S_i(t+1) - S_i(t)}{S_i(t)}, \tag{7.2}$$

and $\mathbf{w}^{\mathrm{T}} = (w_1, w_2, \ldots, w_N)$ is the transposed vector of asset weights

$$w_i = \frac{x_i S_i(t)}{\sum_{j=1}^{N} x_j S_j(t)}; \quad \sum_{i=1}^{N} w_i = 1; \quad 0 \leq w_i \leq 1, \tag{7.3}$$

which determines the fraction of capital invested in each asset. Thus, the return of the portfolio is a convex combination of the returns of the assets included in the portfolio. Following Markowitz, we make the assumption that the distribution of the returns is completely determined by the vector of means $\hat{\mathbf{r}}$ and the covariance matrix $\mathbf{\Sigma}$. In this investigation these parameters are assumed to be known and given as input to the model. Their estimation from historical and market data is itself an active research field (Ledoit and Wolf (2004); Leonard and Hsu (1992)).

The expected value and the variance of the portfolio returns can be expressed in matrix form as

$$\mathrm{E}(r_P) = \sum_{i=1}^{N} w_i \hat{r}_i = \mathbf{w}^{\mathrm{T}} \cdot \hat{\mathbf{r}} \tag{7.4}$$

$$\mathrm{Var}(r_P) = \sum_{i=1}^{N} \sum_{j=1}^{N} w_i w_j \Sigma_{ij} = \mathbf{w}^{\mathrm{T}} \cdot \mathbf{\Sigma} \cdot \mathbf{w}. \tag{7.5}$$

The portfolio selection problem can be formulated as the solution of two different optimization problems that have common solutions. In the first formulation, the goal is

to minimize a measure of risk for a fixed target value R^* of the expected return

$$\begin{aligned} \min_{\mathbf{w}} \quad & \mathrm{Var}(r_P) && = \mathbf{w}^{\mathrm{T}} \cdot \boldsymbol{\Sigma} \cdot \mathbf{w} \\ \text{s.t.} \quad & \mathbf{w}^{\mathrm{T}} \cdot \hat{\mathbf{r}} && = R^* \\ & \textstyle\sum_{i=1}^{N} w_i && = 1 \quad w_i \geq 0,\, i = 1, \ldots, N \end{aligned} \tag{7.6}$$

The dual of this convex optimization problem (Fletcher (2000)) consists in maximizing the expected return while the variance of the portfolio returns is held constant at a specified value $(\sigma^2)^*$, which is determined by the risk profile of the investor. The solution involves optimizing a linear function with linear and quadratic constraints:

$$\begin{aligned} \max_{\mathbf{w}} \quad & \mathrm{E}(r_P) && = \mathbf{w}^{\mathrm{T}} \cdot \hat{\mathbf{r}} \\ \text{s.t.} \quad & \mathbf{w}^{\mathrm{T}} \cdot \boldsymbol{\Sigma} \cdot \mathbf{w} && = (\sigma^2)^* \\ & \textstyle\sum_{i=1}^{N} w_i && = 1 \quad w_i \geq 0,\, i = 1, \ldots, N \end{aligned} \tag{7.7}$$

There are several efficient quadratic optimization techniques that can be used to minimize a quadratic form subject to linear equality and inequality constraints (Gill et al. (1991)). For this reason the first formulation has been chosen. The set of portfolios that are optimal (i.e. that minimize the risk) for fixed values of the expected return R^*, where R^* is allowed to vary in the interval $[R_{\min}, R_{\max}]$ is the *efficient frontier*. Each point in the efficient frontier is said to be *Pareto efficient*. Points on the efficient frontier correspond to minimum-risk portfolios for a given expected return, or, alternatively, portfolios that have the largest expected return from a family of portfolios with equal risk.

7.3.2 Constraints in Portfolio Selection Problems

There are several ways of refining the standard Markowitz model to incorporate constraints that are commonly used in real-world portfolio selection problems. These restrictions are a consequence of market rules and conditions for investment or simply reflect different investor profiles and preferences. For instance, constraints can be included to specify the amount of diversification; e.g., by establishing lower and upper bounds on the investment on an individual asset or on a group of assets. An investor may also want to limit the maximum number of assets included in her portfolio, either to simplify the management of the portfolio or to reduce transaction costs.

In this section, we assume that the investor constructs a portfolio from scratch assuming that there are no transaction costs. The constrained optimization problem

is

$$\min_{\mathbf{z},\mathbf{w}} \quad \mathrm{Var}(r_P) = \mathbf{w}^{[\mathbf{z}]^{\mathrm{T}}} \cdot \boldsymbol{\Sigma}^{[\mathbf{z},\mathbf{z}]} \cdot \mathbf{w}^{[\mathbf{z}]} \tag{7.8}$$

$$\text{s.t.} \quad \mathbf{w}^{[\mathbf{z}]^{\mathrm{T}}} \cdot \hat{\mathbf{r}}^{[\mathbf{z}]} = R^* \tag{7.9}$$

$$\mathbf{a}^{[\mathbf{z}]} \leq \mathbf{w}^{[\mathbf{z}]} \leq \mathbf{b}^{[\mathbf{z}]}, \;\; \mathbf{a}^{[\mathbf{z}]} \geq \mathbf{0}, \;\; \mathbf{b}^{[\mathbf{z}]} \geq \mathbf{0} \tag{7.10}$$

$$\mathbf{l} \leq \mathbf{A}^{[\mathbf{z}]} \cdot \mathbf{w}^{[\mathbf{z}]} \leq \mathbf{u} \tag{7.11}$$

$$\mathbf{z}^{\mathrm{T}} \cdot \mathbf{1} \leq K \tag{7.12}$$

$$\mathbf{w}^{\mathrm{T}} \cdot \mathbf{1} = 1, \;\; \mathbf{w} \geq \mathbf{0}. \tag{7.13}$$

The elements of the binary vector $\mathbf{z}$ specify whether product i is included in the final portfolio ($z_i = 1$) or not ($z_i = 0$). The column vector $\mathbf{w}^{[\mathbf{z}]}$ is obtained by removing from $\mathbf{w}$ components i for which $w_i = 0$. Similarly, matrix $\mathbf{A}^{[\mathbf{z}]}$ is obtained by eliminating the i-th column of $\mathbf{A}$ whenever $z_i = 0$. Finally, $\boldsymbol{\Sigma}^{[\mathbf{z},\mathbf{z}]}$ is obtained by removing from $\boldsymbol{\Sigma}$ the rows and columns for which the corresponding indicator is zero ($z_i = 0$). The symbols $\mathbf{0}$ and $\mathbf{1}$ denote vectors whose entries are all equal to 0 or to 1, respectively.

Minimum and maximum investment constraints, which set a lower and an upper bound on the investment of each asset in the portfolio are captured by (7.10). Vectors $\mathbf{a}$ and $\mathbf{b}$ are $N \times 1$ column vectors with the lower and upper bounds on the portfolio weights, respectively. The inequality (7.11) reflects the M *concentration of capital* constraints. The m-th row of the $M \times N$ matrix $\mathbf{A}$ is the vector of coefficients of the linear combination that defines the constraint. The $M \times 1$ column vectors $\mathbf{l}$ and $\mathbf{u}$ correspond to the lower and upper bounds of the M linear restrictions, respectively. Concentration of capital constraints can be used, for instance, to limit the amount of capital invested in a group of assets, so that investor preferences on certain asset classes can be formally expressed. Since these constraints are linear, they do not increase the difficulty of the problem, which can still be efficiently solved by quadratic programming. Expression (7.12) corresponds to the *cardinality constraint*, which sets a bound on the number of assets that can be included in the final portfolio. This constraint transforms the problem into a mixed-integer quadratic programming problem, which is no longer convex. Finally, equation (7.13) is a budget constraint that ensures that the whole amount of capital is invested in the portfolio. Note that the portfolio selection problem with the cardinality constraint as an inequality (7.12) can be solved by selecting the best of the optimal solutions of a collection of problems that use equality cardinality constraints $\sum_{i=1}^{N} z_i = k$, for $k = 1, 2, \ldots, K$.

7.3.3 Hybrid approaches to optimal portfolio selection

In this section we review various hybrid metaheuristic approaches to solve the problem. Specifically we consider genetic algorithms, simulated annealing and estimation of distribution algorithms. As discussed in Subsection 2.2.1, these metaheuristics are used to explore the combinatorial search space of subsets of the given cardinality. For each subset, a quadratic solver (for instance, the one described in Gill et al. (1991)) is used to address the problem (7.8)-(7.13); that is, the risk of the optimal portfolio that

invests only in the products present in the subset proposed by the metaheuristic is calculated. The dimensionality reduction techniques described in Section 2.4 are applied in a preprocessing step and their impact on the performance of the algorithms is evaluated.

7.3.3.1 Genetic algorithms

We apply genetic algorithms with binary and set encoding as introduced in Chapter 6 for the index tracking problem. As crossover operators for the set encoding, we use RER (Algorithm 13), RRR (Algorithm 12), RAR (Algorithm 14) and TransRAR (Algorithm 15).

7.3.3.2 Simulated Annealing

The simulated annealing metaheuristic is used as defined in Subsection 2.3.1 and in Algorithm 1. Candidate solutions are represented by sets of fixed cardinality as in Chapter 6. The neighborhood operator is defined as the exchange of a product in the portfolio with another one not included in it. Note that this is the mutation operator of the set-based genetic algorithm. The comparison of the performance of SA and GA with sets should therefore reveal to what extent the use of crossover contributes to the efficacy of the algorithm. The free energy function is the risk of the optimal portfolio that invests only in the products included in the set that defines the candidate solution.

7.3.3.3 Estimation of Distribution Algorithms

We apply the estimation of distribution algorithms UMDA, PBIL, PBIL$_c$ and EMNA, which were described in Section 2.3.3. Individuals of a specified cardinality are sampled using Algorithm 4. The objective of using EDAs for portfolio selection is to determine whether this family of algorithms can be effective in a complex combinatorial optimization task such as portfolio selection with cardinality constraints. As will be illustrated by the experiments performed, algorithms of the EDA type have difficulties for problems in which the search takes place in a high-dimensional space. To overcome this limitation, the pruning heuristics outlined in Section 2.4 play a central role.

7.3.4 Empirical evaluation

In this section we analyze the performance of the different portfolio selection methods in terms of optimality of the selected portfolio and computational cost. Experiments are performed on data from the OR-Library (Beasley (1990)), which includes a variety of benchmark problems in the field of Operations Research. For portfolio selection, the data consist in expected returns and covariances of returns for assets included in five major world stock indices. Namely, Hang Seng (Hong Kong, $N = 31$), DAX (Germany, $N = 85$), FTSE (United Kingdom, $N = 89$), Standard & Poor's (United States, $N = 98$) and Nikkei (Japan, $N = 225$). The weekly returns are estimated from the series of stock

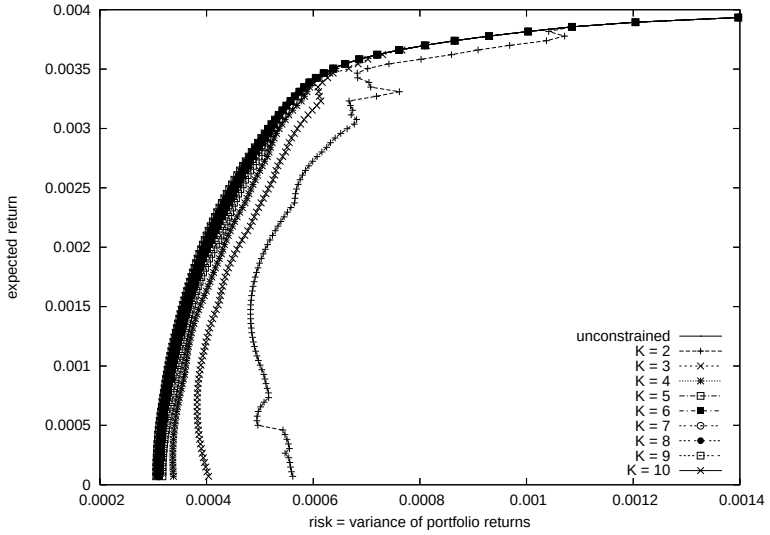


Figure 7.1: Comparison of constrained and unconstrained minimum risk portfolios for different values of the expected return in the Nikkei index problem

prices from March 1992 to September 1997. Stocks with missing values are not considered. This is the reason why, for instance, 89 products are considered for the FTSE index whereas this index is actually composed of 100 assets. The optimizations are performed considering a lower bound for the investment weights ($l_i = 0.01 \forall i$) and with a maximum cardinality constraint of $K = 10$. For each of the hybrid optimization methods the values of the hyperparameters, including the maximum number of generations needed for convergence in the evolutionary algorithms, are determined in exploratory experiments.

For the unconstrained problem, an efficient frontier of $N_F = 100$ points is calculated. These points correspond to optimal portfolios without cardinality or lower bound constraints, whose expected returns are evenly spaced between the largest and the smallest values that can be achieved by portfolios on the efficient frontier. For each of the values of the expected return considered, the minimum risk portfolios for various cardinality constraints ($K = 2, 3, \ldots, 10$) are also computed. To reduce the probability of getting trapped in a subpotimal solution, each point is computed by selecting the best of 5 executions of the hybrid optimization algorithm analyzed.

Figures 7.1, 7.2 and 7.3 show a detailed comparison between the minimum risk portfolios obtained with different values for the cardinality constraint $K = 2, \ldots, 10$, and the unconstrained efficient frontier in the Nikkei index problem using a RAR-GA algorithm with parameter $w = 1$ for the RAR crossover operator. In general, the higher the value of K the closer are the solutions of the cardinality-constrained and the unconstrained problems.

When cardinality constraints are considered, the optimization problem is no longer convex. As a consequence, the solutions identified by minimizing the risk for a fixed value of the expected return need not be Pareto optimal. That is, it may be possible to build portfolios with the same risk and a higher expected return. This could introduce discontinuities in the shape of the efficient frontier (Jobst et al. (2001)). The

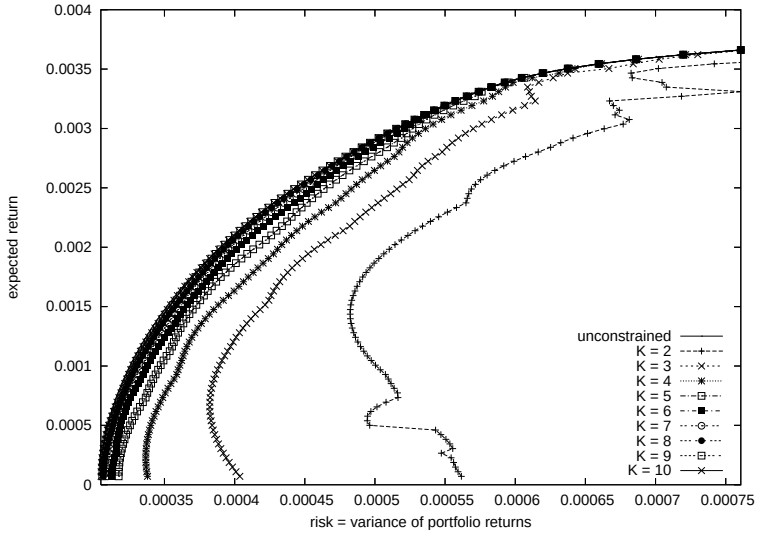


Figure 7.2: Detailed comparison of minimum risk portfolios for different values of the expected return in the Nikkei index problem

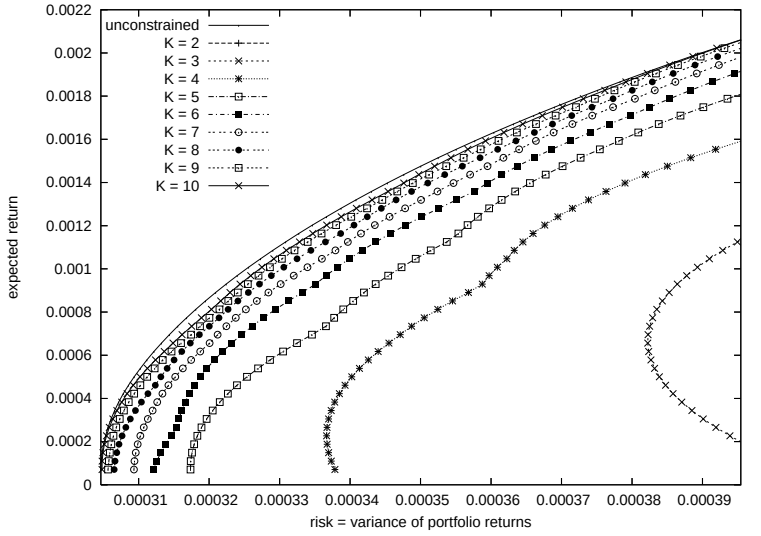


Figure 7.3: A more detailed comparison of minimum risk portfolios for different values of the expected return in the Nikkei index problem

anomalies are specially pronounced for small values of K, the upper bound in the cardinality constraint. To see how this phenomenon can occur, consider a problem with $K = 2$. The efficient frontier can be obtained by performing the *union* among the efficient frontiers corresponding to all possible pairs of assets. The *union* operation is not meant in the set-theoretical sense, but in the portfolio-dominance sense: the set of all non-dominated portfolios forms the final efficient frontier. This operation removes any non-convex regions from the efficient frontier (i.e. all dominated portfolios) and as a result discontinuities can appear. These regions correspond to portfolios that no rational investor would choose, because there is at least one portfolio with a higher value of expected return for the same risk. For instance, consider the curve depicted in Figure 7.3 for $K = 3$. If we would trace a vertical line for the risk value $\sigma^2 = 0.00039$ two

intersections would be obtained. These intersections represent two portfolios with the same risk value, but one of them has higher expected return than the other. Therefore, a rational investor would only consider the portfolio with the higher expected return. The second portfolio does not belong to the efficient frontier. Nevertheless, for the purpose of comparing the performance of the different algorithms it is useful to consider all the portfolios identified by the optimization algorithms, even if they do not belong to the efficient frontier: For a given value of the expected return, portfolios with a more stringent constraint (that is, with a smaller value of K) necessarily have a higher risk, independently of whether they are Pareto optimal or not. In particular, better solutions of the constrained problem will be closer to the unconstrained efficient frontier, even if they do not belong to the efficient frontier of the constrained problem.

For the GA, a steady state substitution scheme is used in each generation. No elitism is used, since this generally leads to premature convergence to local optima. The probabilities of crossover and mutation are $p_C = 1.0$ and $p_M = 0.01$, respectively. The populations evolved are composed of 100 individuals.

For SA, a geometrical cooling scheme with coefficient $\alpha = 0.9$ is used. The initial temperature is chosen following the recommendations in Crama and Schyns (1999): The initial temperature is such that χ_0, the probability of acceptance of new configurations, is sufficiently high. In our experiments the value $\chi_0 = 0.8$ is chosen. The average increase Δ in the objective function value is calculated for $L = 300$ iterations and the initial temperature T_0 is then estimated as

$$T_0 = \frac{-\Delta}{\ln \chi_0} \tag{7.14}$$

For the EDA family, population sizes are generally higher than in the case of the GA. This is because larger samples are needed for an accurate estimation of the distribution parameters in EDA. The population size chosen for the experiments is 300. The fraction of individuals considered to estimate the parameters is 10% of the population in all cases except for EMNA, where larger samples (15% of the original population) are needed for the accurate estimation of the distribution parameters.

The quality of the solutions is measured in terms of the relative distances between the unconstrained solutions (without cardinality and bound constraints) and the corresponding constrained solution with the same expected return

$$D = \frac{1}{N_F} \sum_{i=1}^{N_F} \frac{\sigma_{c,i} - \sigma_i^*}{\sigma_i^*}, \tag{7.15}$$

where $\sigma_{c,i}$ is the risk of the best known solution in the constrained problem for the $i-$th frontier point, and σ_i^* is the optimal risk for the unconstrained problem, obtained by quadratic optimization. All the cardinality-constrained minimum risk portfolios are considered, independently of whether they belong to the efficient frontier or not. In this manner, the results for different algorithms with cardinality constraints can be meaningfully compared. Note that every term in D is greater than or equal to zero because the unconstrained problem is a relaxation of the constrained one.

Table 7.1 summarizes the results of the GA approaches. Clearly, the poor results of the GA with binary encoding and linear penalties indicate that this technique is not adequate for handling unfeasible individuals. In contrast, algorithms in which the generated individuals are always feasible perform better. The best results are obtained by the GA with set encoding that employs either the RAR ($w = 1$) or the TransRAR ($p = 1/2$) crossover operators. Between these two algorithms, the lowest computational cost corresponds to the one that uses the TransRAR crossover. The algorithm using RAR with $w = 3$ obtains solutions of comparable quality to RAR with $w = 1$ except in one case. However, the execution times are higher in all cases. The heuristic repair procedure also obtains results comparable to RAR, but the computational costs are higher. This is because more quadratic optimizations have to be performed by the repair mechanism. As can be seen, the total number of quadratic optimizations required by the GA with the heuristic repair mechanism is always higher than the number of QP optimizations needed by the other GAs.

The results of the SA and EDA approaches are shown in Table 7.2. For SA the results obtained are of comparable quality as those achieved by the set GAs, although for the FTSE problem the best solution reached is of slightly inferior quality. The overall performance of the EDAs is rather poor. These types of algorithms obtain good results only in the easier problems (Hang Seng and DAX). In the other problems the results obtained by EDAs are rather poor. The reason is that the performance of the EDAs is more sensitive to the dimensionality of the problem than GAs and SA. Specifically, the estimation of the probability distribution becomes much more difficult as the dimension of the search space increases. Among the different variations of EDAs the best results are obtained with PBIL.

Next, we apply the dimensionality reduction techniques introduced in Section 2.4 as preprocessing step. The performance of the pruning heuristics strongly depends on the values chosen for the parameters: In the block pruning heuristic, the threshold ϵ; for the greedy backward and forward selection heuristics the amount of additional products T. The parameter ϵ for the block pruning heuristic is set to $l_i/2$. The reason for this choice is that products whose weights in an unconstrained optimization are under their lower bound l_i can be assumed to have a lower probability of being included in the final cardinality-constrained solution. The parameter T is chosen after performing exploratory tests with several values in the range $T \in [0, 12]$. After pruning, an exhaustive search is performed in the pruned universe with a cardinality constraint of $K = 10$. The results for these experiments are shown in Figure 7.4. In general, for low values of T ($T \leq 3$) the performance of backward selection is better than forward selection. Above this value, the results for the Hang Seng (7.4(a)) and Nikkei (7.4(e)) index problems do not show significant differences. For the other problems, the quality of the results obtained by both heuristics is similar for values above $T = 8$. For the DAX (7.4(b)) and the FTSE (7.4(c)) problems greedy forward selection is slightly better for the range $3 < T < 8$, while the opposite is true for the S&P problem (7.4(d)). Therefore the value $T = 8$ is chosen for the final experiments.

Tables 7.3, 7.4 and 7.5 display the results of the experiments using exhaustive search and exact optimization on the pruned universes. In these experiments, pruning is first

Table 7.1: Results for portfolio selection using the different GA approaches.

Algorithm	Index	Best D	Success rate	Time (s)	Optimizations
Linear penalty	Hang Seng	**0.00321150**	0.87	808.1	$1.35 \cdot 10^7$
	DAX	2.53807879	0.70	3235.8	$4.60 \cdot 10^7$
	FTSE	1.93268316	0.53	3855.2	$5.81 \cdot 10^7$
	S&P	**4.69373181**	0.77	4922.3	$7.03 \cdot 10^7$
	Nikkei	0.22223473	0.42	5064.4	$6.48 \cdot 10^7$
Heuristic Repair	Hang Seng	**0.00321150**	1.00	1171.7	$2.18 \cdot 10^7$
	DAX	**2.53162860**	1.00	4878.6	$7.45 \cdot 10^7$
	FTSE	**1.92150019**	0.96	6485.6	$9.70 \cdot 10^7$
	S&P	**4.69373181**	0.98	8226.3	$11.42 \cdot 10^7$
	Nikkei	**0.20197748**	0.99	9658.3	$11.47 \cdot 10^7$
GA-RER	Hang Seng	**0.00321150**	0.93	802.7	$1.40 \cdot 10^7$
	DAX	2.77353490	0.36	1915.2	$2.83 \cdot 10^7$
	FTSE	2.00581544	0.39	3653.4	$5.52 \cdot 10^7$
	S&P	4.81011478	0.39	3938.1	$5.90 \cdot 10^7$
	Nikkei	1.00264537	0.25	3321.0	$5.11 \cdot 10^7$
GA-RRR	Hang Seng	**0.00321150**	0.93	802.7	$1.40 \cdot 10^7$
	DAX	2.83690197	0.40	1628.9	$2.81 \cdot 10^7$
	FTSE	1.97722629	0.40	3252.6	$5.50 \cdot 10^7$
	S&P	4.76271495	0.41	3597.2	$5.97 \cdot 10^7$
	Nikkei	1.03869098	0.25	3045.5	$5.20 \cdot 10^7$
GA-RAR $w = 1$	Hang Seng	**0.00321150**	1.00	539.1	$8.59 \cdot 10^6$
	DAX	**2.53162860**	1.00	2368.6	$3.12 \cdot 10^7$
	FTSE	**1.92150019**	0.95	4716.3	$6.09 \cdot 10^7$
	S&P	**4.69373181**	0.99	4931.9	$6.25 \cdot 10^7$
	Nikkei	**0.20197748**	1.00	7537.7	$7.18 \cdot 10^7$
GA-RAR $w = 3$	Hang Seng	**0.00321150**	1.00	884.4	$1.41 \cdot 10^7$
	DAX	**2.53162860**	1.00	2977.2	$3.89 \cdot 10^7$
	FTSE	1.92158975	0.96	4736.8	$6.09 \cdot 10^7$
	S&P	**4.69373181**	0.99	5013.9	$6.25 \cdot 10^7$
	Nikkei	**0.20197748**	1.00	7812.9	$7.18 \cdot 10^7$
GA-TransRAR	Hang Seng	**0.00321150**	1.00	497.6	$8.56 \cdot 10^6$
	DAX	**2.53162860**	1.00	1966.2	$3.11 \cdot 10^7$
	FTSE	**1.92150019**	1.00	3731.2	$6.07 \cdot 10^7$
	S&P	**4.69373181**	1.00	3912.8	$6.22 \cdot 10^7$
	Nikkei	**0.20197748**	1.00	4710.5	$7.15 \cdot 10^7$

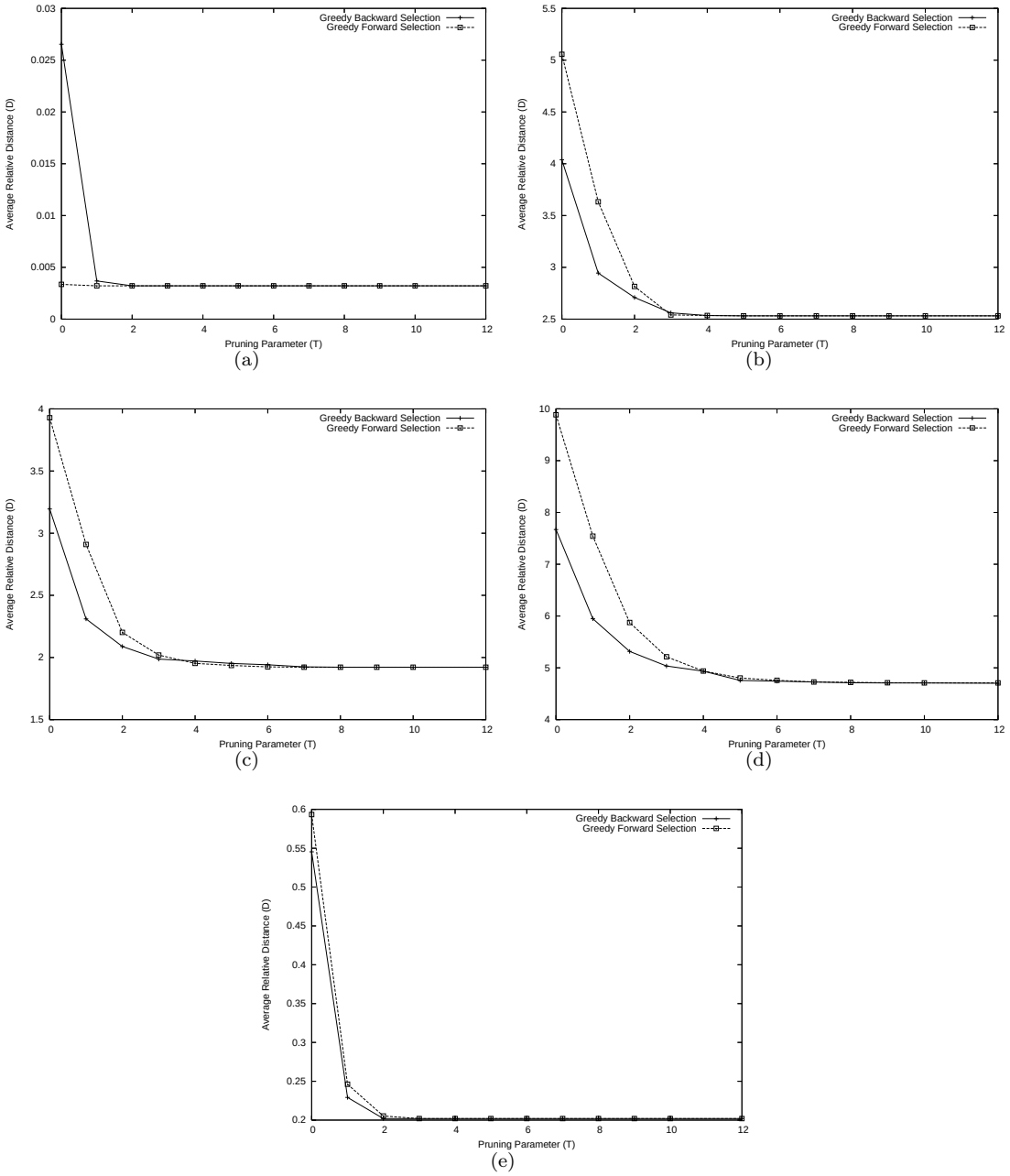


Figure 7.4: Results of experiments using exhaustive search and greedy backward and forward selection. Above the chosen value $T = 8$ the results for both heuristics are very similar. 7.4(a) Hang Seng, 7.4(b) DAX, 7.4(c) FTSE, 7.4(d) S&P, 7.4(e) Nikkei.

Table 7.2: Results for the SA and EDA approaches.

Algorithm	Index	Best D	Success rate	Time (s)	Optimizations
SA	Hang Seng	**0.00321150**	1.00	1499.9	$3.87 \cdot 10^7$
	DAX	**2.53162860**	0.98	2877.3	$7.63 \cdot 10^7$
	FTSE	1.92205745	0.92	3610.4	$8.87 \cdot 10^7$
	S&P	**4.69373181**	0.91	3567.8	$9.54 \cdot 10^7$
	Nikkei	**0.20197748**	0.95	4274.5	$9.25 \cdot 10^7$
UMDA (EDA)	Hang Seng	**0.00321150**	1.00	1021.6	$2.83 \cdot 10^7$
	DAX	5.05951251	0.43	2450.2	$4.69 \cdot 10^7$
	FTSE	3.55519141	0.33	2483.5	$4.90 \cdot 10^7$
	S&P	8.16812536	0.35	2701.3	$5.17 \cdot 10^7$
	Nikkei	6.97027831	0.25	2530.4	$3.70 \cdot 10^7$
PBIL (EDA)	Hang Seng	**0.00321150**	1.00	2292.8	$5.55 \cdot 10^7$
	DAX	**2.53162860**	0.94	4489.1	$7.70 \cdot 10^7$
	FTSE	1.92208910	0.85	4782.3	$8.06 \cdot 10^7$
	S&P	4.69570006	0.88	5100.2	$8.28 \cdot 10^7$
	Nikkei	0.30164777	0.43	7486.5	$8.21 \cdot 10^7$
PBIL_c (EDA)	Hang Seng	**0.00321150**	1.00	3095.7	$5.58 \cdot 10^7$
	DAX	**2.53162860**	0.80	8122.6	$7.71 \cdot 10^7$
	FTSE	1.92670560	0.65	8735.1	$8.02 \cdot 10^7$
	S&P	4.70389481	0.67	9659.7	$8.27 \cdot 10^7$
	Nikkei	0.36541190	0.32	17569.5	$7.39 \cdot 10^7$
EMNA (EDA)	Hang Seng	**0.00321150**	1.00	9892.6	$16.72 \cdot 10^7$
	DAX	2.56612811	0.48	38667.6	$22.37 \cdot 10^7$
	FTSE	2.00176394	0.41	42505.9	$23.15 \cdot 10^7$
	S&P	4.91456924	0.35	49267.4	$23.57 \cdot 10^7$
	Nikkei	743.55201383	0.26	68726.9	$9.08 \cdot 10^7$

applied to eliminate assets that are not likely to appear in the globally optimal solution. Then, the exact cardinality constrained optimal solution (with $K = 10$) of the reduced problem is obtained by exhaustive search. This establishes a benchmark with which to compare the effectiveness of the proposed metaheuristics in the reduced problem. The best solutions are obtained using block pruning. Since block pruning does not set an upper bound on the maximum number of variables, it may be the case that not many variables are discarded. For instance, this is what actually happens in the S&P index. Nonetheless, it is remarkable that, in the problems investigated, the heuristic with the lowest computational cost performs best. Greedy forward and backward selection attain more regular execution times, but they clearly prune too many products that do in fact appear in the optimal solutions.

In Table 7.6 we show the results obtained by RAR-GA, TransRAR-GA, SA and the EDA approaches with a block pruning preprocessing step. This table includes a column labeled *Speed-up factor* that displays the improvements in the times of computation achieved. Speed-up factors are calculated as the ratio of execution times without and with pruning. The corresponding tables for greedy forward and backward selection are included in Appendix B for completeness. The improvements in efficiency are fairly

Table 7.3: Results for block pruning and exhaustive search.

Index	D	Time (s)	Optimizations
Hang Seng	**0.00321150**	1.06	$2.27 \cdot 10^4$
DAX	**2.53162860**	3643.6	$6.69 \cdot 10^7$
FTSE	1.92152413	16136.5	$2.90 \cdot 10^8$
S&P	**4.69373181**	79746.2	$1.24 \cdot 10^9$
Nikkei	**0.20197748**	47.8	$4.17 \cdot 10^5$

Table 7.4: Results for greedy backward selection and exhaustive search.

Index	D	Time (s)	Optimizations
Hang Seng	**0.00321150**	1.05	$2.29 \cdot 10^4$
DAX	**2.53162860**	270.05	$5.49 \cdot 10^6$
FTSE	1.92152413	265.8	$5.56 \cdot 10^6$
S&P	4.71092502	300.72	$6.35 \cdot 10^6$
Nikkei	**0.20197748**	115.5	$4.27 \cdot 10^5$

Table 7.5: Results for greedy backward selection and exhaustive search.

Index	D	Time (s)	Optimizations
Hang Seng	**0.00321150**	5.44	$1.04 \cdot 10^5$
DAX	**2.53162860**	211.55	$4.62 \cdot 10^6$
FTSE	1.92152413	163.4	$3.67 \cdot 10^6$
S&P	4.72021851	175.1	$4.01 \cdot 10^6$
Nikkei	**0.20197748**	102.4	$3.28 \cdot 10^6$

large, especially for the Hang Seng index. In this case, the use of block pruning already yields fewer than K products in many cases, so that no combinatorial search is needed. The speed-up factors range from ≈ 9 (UMDA, Nikkei) to ≈ 650 (SA, Hang Seng). Regarding the quality of the solutions, in the case of the RAR-GA approach, the best solution is also reached (i.e., no product appearing in a best known solution is pruned), except in the case of the FTSE index, where a single product that is included in the best known solution is actually pruned. Nonetheless, in this case, the difference between the best known result and the result obtained after pruning is almost negligible. The success rates remain of the same order in all cases. The improvements in performance are particularly significant in the EDA-based approaches. When the investment universe is pruned, PBIL achieves the same quality as RAR-GA, although the execution times are higher. Even the worst-performing algorithm of the EDA family, EMNA, identifies fairly good solutions with a reasonable computational cost when pruning is used: For the Nikkei index, the best known solution is reached with a 99% success rate and a lower execution time. Pruning also significantly improves the quality of the solutions obtained with UMDA.

Table 7.6: Results for the RAR-GA, TransRAR-GA, SA and EDA approaches with block pruning

Algorithm	Index	Best D	Success rate	Time(s)	Speed-up factor	Opt.
RAR-GA ($w = 1$)	Hang Seng	**0.00321150**	1.00	3.81	315.17	$4.92 \cdot 10^4$
	DAX	**2.53162860**	1.00	142.5	22.31	$1.96 \cdot 10^6$
	FTSE	1.92152413	0.94	211.5	30.19	$2.88 \cdot 10^6$
	S&P	**4.69373181**	0.99	205.8	31.95	$2.76 \cdot 10^6$
	Nikkei	**0.20197748**	1.00	241.5	40.97	$2.75 \cdot 10^6$
TransRAR GA	Hang Seng	**0.00321150**	1.00	3.14	158.47	$4.84 \cdot 10^4$
	DAX	**2.53162860**	1.00	117.1	16.79	$1.92 \cdot 10^6$
	FTSE	1.92152413	1.00	175.3	21.28	$2.82 \cdot 10^6$
	S&P	**4.69373181**	1.00	167.9	23.30	$2.70 \cdot 10^6$
	Nikkei	**0.20197748**	1.00	202.5	23.26	$2.70 \cdot 10^6$
SA	Hang Seng	**0.00321150**	1.00	2.31	649.31	$3.85 \cdot 10^4$
	DAX	**2.53162860**	0.96	17.3	166.32	$3.17 \cdot 10^5$
	FTSE	1.92270249	0.88	22.2	162.63	$4.33 \cdot 10^5$
	S&P	4.69759052	0.85	21.9	162.91	$4.03 \cdot 10^5$
	Nikkei	**0.20197748**	0.99	47.0	90.95	$2.61 \cdot 10^5$
UMDA (EDA)	Hang Seng	**0.00321150**	1.00	24.0	42.57	$5.19 \cdot 10^5$
	DAX	**2.53162860**	0.97	211.4	11.59	$4.02 \cdot 10^6$
	FTSE	1.92639746	0.84	243.9	10.18	$4.47 \cdot 10^6$
	S&P	4.71355872	0.77	237.9	11.35	$4.28 \cdot 10^6$
	Nikkei	**0.20197748**	0.99	274.9	9.20	$3.43 \cdot 10^6$
PBIL (EDA)	Hang Seng	**0.00321150**	1.00	22.9	100.12	$4.92 \cdot 10^5$
	DAX	**2.53162860**	0.99	203.0	22.11	$3.81 \cdot 10^6$
	FTSE	1.92152413	0.91	237.4	20.14	$4.24 \cdot 10^6$
	S&P	**4.69373181**	0.93	226.8	22.49	$4.06 \cdot 10^6$
	Nikkei	**0.20197748**	1.00	263.1	28.45	$3.26 \cdot 10^6$
PBIL_c (EDA)	Hang Seng	**0.00321150**	1.00	19.1	162.08	$3.72 \cdot 10^5$
	DAX	**2.53162860**	0.99	164.9	49.26	$2.88 \cdot 10^6$
	FTSE	1.92396198	0.87	190.8	45.78	$3.21 \cdot 10^6$
	S&P	**4.69373181**	0.87	196.2	49.23	$3.07 \cdot 10^6$
	Nikkei	**0.20197748**	1.00	183.7	95.64	$2.46 \cdot 10^6$
EMNA (EDA)	Hang Seng	**0.00321150**	1.00	28.2	350.80	$6.20 \cdot 10^5$
	DAX	2.53517771	0.71	259.1	149.24	$4.81 \cdot 10^6$
	FTSE	1.94102813	0.60	302.6	140.47	$5.35 \cdot 10^6$
	S&P	4.74273727	0.55	311.8	158.01	$5.12 \cdot 10^6$
	Nikkei	**0.20197748**	0.99	232.4	295.73	$4.11 \cdot 10^6$

7.3.5 Discussion

The method that uses SA to solve the combinatorial part of the optimization problem identifies portfolios that are almost as good as those obtained by RAR-GA and TransRAR-GA at a similar or lower computational cost. The portfolios obtained by the GA with binary encoding and heuristic repair are also very good, but require longer computations because of the costs of the repair mechanism. In contrast, EDA algorithms on their own are not competitive and perform poorly. The algorithms of the EDA family have difficulties with high-dimensional problems: When the number of products in the universe of the optimization increases, sampling and estimation of the evolved probability distributions becomes increasingly difficult. A possible solution to this shortcoming is to identify the assets that are not likely to be included in the optimal portfolio and remove them from the investment universe. These assets are identified by solving a relaxed optimization problem. The efficiency of the search performed by the metaheuristic methods in the reduced space is significantly improved. In particular, the EDA algorithms (especially PBIL) become competitive with SA, RAR-GA and TransRAR-GA. Pruning can in fact be so effective that, in some cases (e.g. for the Hang Seng and the Nikkei problems), the exact solution of the reduced optimization problem by exhaustive search in the pruned space can be found at a lower computational cost than using GA, SA or EDAs.

7.4 Portfolio Selection with Transaction costs

Consider the problem of managing a portfolio that invests in a set of N assets and can be rebalanced at times $t = 1, \ldots, T$. Let $\left\{\{S_i(t)\}_{i=1}^N;\, t = 1, \ldots, T\right\}$ be the time series of the asset prices. The composition of the portfolio in the interval $[t-1, t)$ is given by the vector

$$\mathbf{x}(t-1) = \{x_i(t-1)\}_{i=1}^N, \tag{7.16}$$

where $x_i(t-1)S_i(\tau)$ is the amount of capital invested in asset i, at time τ in the interval $t-1 \leq \tau < t$. The value of the portfolio is

$$P(\tau) = \sum_{i=1}^{N} x_i(t-1)S_i(\tau), \quad t-1 \leq \tau < t. \tag{7.17}$$

Let

$$P(t^-) = \sum_{i=1}^{N} x_i(t-1)S_i(t). \tag{7.18}$$

be the value of the portfolio at the end of the interval $[t-1, t)$. At time t the portfolio is rebalanced with the goal of maximizing the expected return in the interval $[t, t+1)$ while minimizing the associated risk. The investment decision is made based on the available information up to t. The new portfolio has a different composition $\mathbf{x}(t) = \{x_i(t)\}_{i=1}^N$

that is held constant during the period $[t, t+1)$. Its value is

$$P(\tau) = \sum_{i=1}^{N} x_i(t) S_i(\tau), \quad t \leq \tau < t+1. \tag{7.19}$$

An alternative way of specifying the composition of the portfolio is to use the vector of weights $\mathbf{w}(\tau) = \{w_i(\tau)\}_{i=1}^{N}$. The components of this vector are the fraction of $P(t^-)$, the wealth available for investment at time t^-, allocated to each of the assets in the period $[t, t+1)$

$$w_i(\tau) = \frac{x_i(t) S_i(\tau)}{P(t^-)} = \frac{x_i(t) S_i(\tau)}{\sum_{j=1}^{N} x_j(t-1) S_j(t)}, \quad t \leq \tau < t+1. \tag{7.20}$$

If there are transaction costs these weights satisfy the inequality constraint

$$\sum_{i=1}^{N} w_i(t) \leq 1, \tag{7.21}$$

with equality only if the transaction costs are zero.

The rebalancing is made with the restriction that the portfolio is self-financing

$$P(t^-) = \Phi(\mathbf{x}, t) + P(t), \tag{7.22}$$

where $P(t^-)$ is the value of the portfolio before rebalancing, $P(t)$ the value of the portfolio after rebalancing and $\Phi(\mathbf{x}, t)$ are the costs incurred in the transactions that are needed to rebalance the portfolio. In this work we assume piecewise linear transaction costs

$$\Phi(\mathbf{x}, t) = \sum_{i=1}^{N} \kappa_i \left| x_i(t) S_i(t) - x_i(t-1) S_i(t) \right|, \tag{7.23}$$

where κ_i is the fee associated with buying or selling one dollar worth of asset i. The generalization of (7.23) to consider different selling and buying costs is straightforward.

Using the explicit form of the transaction costs (7.23) the self-balancing constraint (7.22) becomes

$$P(t^-) = \sum_{i=1}^{N} x_i(t) S_i(t) + \sum_{i=1}^{N} \kappa_i \left| x_i(t) S_i(t) - x_i(t-1) S_i(t) \right|. \tag{7.24}$$

Dividing both sides by $P(t^-)$ one obtains

$$1 = \frac{\sum_{i=1}^{N} x_i(t) S_i(t)}{\sum_{j=1}^{N} x_j(t-1) S_j(t)} + \sum_{i=1}^{N} \kappa_i \left| \frac{x_i(t) S_i(t)}{\sum_{j=1}^{N} x_j(t-1) S_j(t)} - \frac{x_i(t-1) S_i(t)}{\sum_{j=1}^{N} x_j(t-1) S_j(t)} \right|. \tag{7.25}$$

In terms of $\mathbf{w}(t) = \{w_i(t)\}_{i=1}^N$ the self-financing constraint is

$$\sum_{i=1}^{N} w_i(t) + \sum_{i=1}^{N} \kappa_i \left| w_i(t) - w_i^{(0)}(t) \right| = 1, \tag{7.26}$$

where the vector of weights immediately before rebalancing is

$$\begin{aligned} w_i^{(0)}(t) &\equiv \frac{x_i(t-1)S_i(t)}{\sum_{j=1}^N x_j(t-1)S_j(t)}, \quad i = 1, \ldots, N, \\ \sum_{i=1}^{N} w_i^{(0)}(t) &= 1. \end{aligned} \tag{7.27}$$

The goal of the optimization is to minimize the risk of the portfolio and maximize the expected return. In terms of the returns of the individual assets in the period $[t, t+1)$

$$r_i(t) = \frac{S_i(t+1)}{S_i(t)} - 1, \quad i = 1, \ldots, N, \tag{7.28}$$

the return of the portfolio in that interval is

$$\begin{aligned} r_P(t) &= \frac{\sum_{i=1}^N x_i(t)S_i(t+1)}{\sum_{i=1}^N x_i(t-1)S_i(t)} - 1 = \sum_{i=1}^{N} w_i(t)\frac{S_i(t+1)}{S_i(t)} - 1 = \\ &= \sum_{i=1}^{N} w_i(t)r_i(t) + \sum_{i=1}^{N} w_i(t) - 1 = \sum_{i=1}^{N} w_i(t)r_i(t) - \kappa_i \left| w_i(t) - w_i^{(0)}(t) \right|. \end{aligned} \tag{7.29}$$

The expected value of the portfolio return is

$$\mathbb{E}\left[r_P(t)\right] = \sum_{i=1}^{N} w_i(t)\hat{r}_i - \sum_{i=1}^{N} \kappa_i \left| w_i(t) - w_i^{(0)}(t) \right|, \tag{7.30}$$

where $\{\hat{r}_i = \mathbb{E}\left[r_i(t)\right]\}_{i=1}^N$ are the expected returns of the individual assets, which are assumed to be constant by stationarity.

The risk associated with the investment is quantified in terms of the variance of the portfolio

$$\mathrm{Var}\left[r_P(t)\right] = \mathbf{w}^{\mathrm{T}}(t) \cdot \mathbf{\Sigma} \cdot \mathbf{w}(t), \tag{7.31}$$

where $\mathbf{\Sigma}$ is the $N \times N$ covariance matrix of the asset returns. Since the optimization period is fixed, we drop the time index and simply use $\mathbf{w}^{(0)}$ for the vector of portfolio weights prior to rebalancing and $\mathbf{w}$ for the vector of portfolio weights immediately after rebalancing.

To encode the cardinality constraints it is convenient to introduce an N-dimensional vector of binary variables $\mathbf{z}$. The ith component of this vector specifies whether product i is included in the final portfolio ($z_i = 1$) or not ($z_i = 0$). Using these conventions, the

optimal portfolio is the solution of the constrained minimization problem

$$\min_{\mathbf{z},\mathbf{w}} \left[{\mathbf{w}^{[\mathbf{z}]}}^{\mathrm{T}} \cdot \boldsymbol{\Sigma}^{[\mathbf{z},\mathbf{z}]} \cdot \mathbf{w}^{[\mathbf{z}]} - \alpha \left({\mathbf{w}^{[\mathbf{z}]}}^{\mathrm{T}} \cdot \hat{\mathbf{r}}^{\mathrm{T}} - \boldsymbol{\kappa}^{T} \cdot \left| \mathbf{w} - \mathbf{w}^{(\mathbf{0})} \right| \right) \right] \tag{7.32}$$

$${\mathbf{w}^{[\mathbf{z}]}}^{\mathrm{T}} \cdot \mathbf{1} + \boldsymbol{\kappa}^{T} \cdot \left| \mathbf{w} - \mathbf{w}^{(\mathbf{0})} \right| = 1 \tag{7.33}$$

$$\mathbf{a}^{[\mathbf{z}]} \leq \mathbf{w}^{[\mathbf{z}]} \leq \mathbf{b}^{[\mathbf{z}]}, \;\; \mathbf{a}^{[\mathbf{z}]} \geq \mathbf{0}, \;\; \mathbf{b}^{[\mathbf{z}]} \geq \mathbf{0} \tag{7.34}$$

$$\mathbf{l} \leq \mathbf{A}^{[\mathbf{z}]} \cdot \mathbf{w}^{[\mathbf{z}]} \leq \mathbf{u} \tag{7.35}$$

$$\mathbf{z}^{\mathrm{T}} \cdot \mathbf{1} \leq K \tag{7.36}$$

$$w_i \geq w_i^{(0)} + P_i \;\; \text{or} \;\; w_i \leq w_i^{(0)} - S_i \;\; \text{or} \;\; w_i = w_i^{(0)}$$
$$i = 1, \ldots, N. \tag{7.37}$$

The column vector $\mathbf{w}^{[\mathbf{z}]}$ is obtained by removing from $\mathbf{w}$ those components i for which $z_i = 0$. Similarly, the matrix $\mathbf{A}^{[\mathbf{z}]}$ is obtained by eliminating the i-th column of $\mathbf{A}$ whenever $z_i = 0$. Finally, $\boldsymbol{\Sigma}^{[\mathbf{z},\mathbf{z}]}$ is obtained by removing from $\boldsymbol{\Sigma}$ the rows and columns for which the corresponding indicator is zero ($z_i = 0$). The symbols $\mathbf{0}$ and $\mathbf{1}$ denote vectors whose components are all 0 or all 1, respectively.

The objective function consists of three terms: The first one is the variance of the portfolio, which is to be minimized. The second corresponds to the expected return of the portfolio, which we wish to maximize and is therefore included with a negative sign. The last one corresponds to the adjustment of the expected returns due to transaction costs. The positive constant $\alpha > 0$ determines the importance of the terms corresponding to the cost-adjusted expected return in the objective function.

Equation (7.33) reflects the *self-financing* constraint, which ensures that the value of the portfolio before rebalancing is equal to the value of the portfolio after rebalancing plus the transaction costs incurred. *Minimum and maximum investment* constraints, which set a lower and an upper bound on the investment of each asset in the portfolio, are encoded in the restriction (7.34). In this constraint $\mathbf{a}$ and $\mathbf{b}$ are $N \times 1$ column vectors whose components are the lower and upper bounds on the portfolio weights, respectively. Inequality (7.35) corresponds to *capital concentration* constraints. The m-th row of the $M \times N$ matrix $\mathbf{A}$ is the vector of coefficients of the linear combination that defines this constraint. The $M \times 1$ column vectors $\mathbf{l}$ and $\mathbf{u}$ correspond to the lower and upper bounds of the M linear restrictions, respectively. Capital concentration constraints can be used, for instance, to limit the amount of capital invested in a group of assets, so that investor preferences for certain asset classes can be taken into account in the optimization. Expression (7.36) is the *cardinality constraint*, which sets a bound on the maximum number of assets that can be included in the final portfolio. Finally, the investor can impose *trading size* or *turnover* constraints (7.37). These constraints reflect the fact that the investor may not wish to modify the portfolio by buying or selling small quantities of assets (Crama and Schyns (1999)). Market restrictions that specify minimal trading volumes can be handled in a similar way. Trading size constraints are difficult to handle because they are disjunctive. The solution space is partitioned into multiple feasible regions that are separated by forbidden regions. Specifically, for each asset, only one out of three mutually exclusive alternatives can occur: (i) The change

is greater or equal than $S_i \geq 0$ when selling, (*ii*) an amount of product greater or equal than $P_i \geq 0$ is purchased, (*iii*) the product is neither sold nor purchased.

7.4.1 Lasso penalties

The term corresponding to transaction costs in the objective function (7.32) can be seen as a lasso penalty term. The lasso ("least absolute shrinkage and selection operator") is a statistical technique for regression in which a penalty term proportional to the L_1 norm of the vector of regression coefficients is included in the objective function to be minimized (Tibshirani (1996)). Norm-constrained portfolios in which the standard Markowitz framework is extended by including a penalty term proportional to some norm of the portfolio weight vector $||\mathbf{w}||_1$ in the cost function have been investigated in DeMiguel et al. (2009a). If an L_1 norm is used, provided that the strength of the the penalty is sufficiently large, some coefficients in $\mathbf{w}$ are forced to be zero (Tibshirani (1996)). Therefore, increasing the weight of this penalty in the cost function tends to reduce the cardinality of the portfolio.

The L_1 penalty associated with transaction costs is of a different type. It is proportional to

$$\left|\mathbf{w} - \mathbf{w}^{(\mathbf{0})}\right|. \tag{7.38}$$

This term penalizes deviations from the *initial portfolio* $\mathbf{w}^{(\mathbf{0})}$. The sparsifying effect of this L_1 penalty favors the selection of portfolios in which some of the components of $\mathbf{w} - \mathbf{w}^{(\mathbf{0})}$ are exactly zero. This means that there is a tendency not to perform transactions unless they lead to large expected returns with a low risk. The result is a regularization effect that avoids large fluctuations in the composition of the portfolio. Note that such fluctuations are undesirable because they result in large transaction costs, which reduce the net return of the portfolio. As shown in DeMiguel et al. (2009a) the out-of-sample performance of a non-regularized portfolio can be very poor. The main reason is that the inputs of the optimization model (the means and the covariance matrix of the asset returns) are estimated from historical data, which can be a poor predictor of future behavior. Furthermore, small changes in these estimates can produce large modifications of the estimated optimal portfolio, which is an undesirable instability. This lack of stability and sensitivity to the model inputs generally results in poor out-of-sample performance. Several authors have pointed out that regularization techniques can be a way to avoid overfitting and improve the generalization performance of the portfolios selected (DeMiguel et al. (2009a); Jagannathan and Ma (2003)). Including this L_1 penalty can also be seen as a form of regularization that is expected to improve the out-of-sample performance of the portfolio.

The observation that transaction costs in the portfolio selection problem can have a regularization effect suggests the possibility of minimizing a modified objective function

$$\min_{\mathbf{z},\mathbf{w}} \ \mathbf{w}^{[\mathbf{z}]^{\mathrm{T}}} \cdot \mathbf{\Sigma}^{[\mathbf{z},\mathbf{z}]} \cdot \mathbf{w}^{[\mathbf{z}]} - \alpha \mathbf{w}^T \cdot \hat{\mathbf{r}} + \boldsymbol{\gamma}^T \cdot \left|\mathbf{w} - \mathbf{w}^{(\mathbf{0})}\right|, \tag{7.39}$$

in which $\boldsymbol{\gamma}$ represents the strength of the regularization term, which can be set independently of the actual transaction costs. In subsection 7.4.3 we perform experiments in which this cost function is minimized with $\boldsymbol{\gamma} = \gamma\mathbf{1}$. The value of γ is selected by cross validation. Typically, a large value is chosen, which means that the portfolios selected tend to be very stable. We will refer to this portfolio selection strategy as the *lasso approach*.

7.4.2 Hybrid Approach to Portfolio Selection under Transaction Costs

The portfolio selection problem without transaction costs and without the constraints (7.36) and (7.37) can be solved in polynomial time using standard quadratic optimization techniques (for instance, the one used in Gill et al. (1991)). These techniques guarantee that the global optimum is reached, provided that some standard assumptions on the objective function and the constraints (positive-definiteness of the Hessian, continuous derivatives, quadratic or linear constraints) hold. However, the piecewise linear transaction costs cannot be handled by a standard QP solver because they are non-differentiable. Furthermore, the optimization problem with cardinality or turnover constraints becomes NP-Complete (Moral-Escudero et al. (2006)). Specifically, the inclusion of cardinality constraints means that one needs to solve the combinatorial optimization problem of selecting the optimal subset of $k \leq K$ products from the original investment universe, where K is the upper bound on the number of products that can be included in the final portfolio. Finally, the restrictions on the minimum trading size introduce further combinatorial complexity in the problem: one needs to know whether the portfolio rebalancing process involves buying, selling or holding the position in each of the assets.

In this section, we propose a *memetic* algorithm to address this hybrid optimization problem. Memetic algorithms (Moscato and Cotta (2003)) are a specific kind of *hybrid* metaheuristic techniques (Raidl (2006)) in which evolutionary algorithms are combined with specific knowledge of the problem at hand. As expressed by the No-Free-Lunch theorems for optimization (Wolpert and Macready (1997)), no general-purpose algorithm can perform better than random search when averaged over all classes of optimization problems. Therefore, to design effective algorithms, it is necessary to introduce some kind of bias that incorporates in the search specific knowledge of the problem to be solved. A simple way of incorporating this knowledge is to perform a local optimization step right after mutation or recombination. In combinatorial problems, hill climbing heuristics are frequently used to improve the offspring (Moscato and Cotta (2003)).

The memetic approach proposed in this section handles the problem by treating the combinatorial and the continuous aspects of the optimization task separately. A genetic algorithm with an *extended* set representation is used to address the combinatorial aspect of the problem. This algorithm generates candidate solutions that specify the subset of assets of the specified cardinality that should be included in the portfolio and the type of trades to be made when rebalancing the current portfolio. The fitness of this candidate solution is the optimal value of the portfolio selection problem in the restricted universe of investment specified by the candidate solution proposed by the genetic algorithm. This

subordinate optimization problem does not have cardinality or turnover constraints, which means that it can be solved using standard QP solvers.

In the extended set encoding the candidate solutions are represented as a subset of the appropriate cardinality. Assets that belong in this set are included in the rebalanced portfolio. There are transaction costs associated with the asset trades that are needed to build the new portfolio, characterized by the vector of weights $\mathbf{w}$, from the original portfolio, characterized by the vector of weights $\mathbf{w}^{(0)}$. For each element in the set we include an additional attribute that specifies whether the corresponding asset is sold, purchased or is left unchanged in this portfolio rebalancing operation. Including the information in the chromosome is advantageous for two reasons: First, it is a way of directly handling the turnover constraint (7.37). Once the information of the presence or absence of a trade and its direction for each asset is known, only one of the three inequalities in (7.37) is relevant. Since each of the inequalities is linear when considered in isolation, the selected inequality can be included in the set of linear constraints of the subordinate optimization problem. Second, the absolute values in the objective function and in the budget constraint (7.33) can be eliminated once this attribute is known by making the substitution

$$\left|\mathbf{w} - \mathbf{w}^{(\mathbf{0})}\right| = \sum_{i \in \text{Sold}} \left(w_i^{(0)} - w_i\right) + \sum_{i \in \text{Purchased}} \left(w_i - w_i^{(0)}\right). \tag{7.40}$$

In this manner, these terms become differentiable. Furthermore, there is no need to increase the number of variables from N to $3N$ as is usually done to eliminate the absolute values. Therefore, the simplifications that result from using the information provided by the candidate solutions in the extended set encoding allow the use of standard QP solvers to address the subordinate optimization problem. Note that this approach remains valid even if the transaction costs take a more complicated form (e.g., if they are different for buying and selling).

The combinatorial search takes place in the space

$$\Theta = \{(s,t) : s \in \cup_{k=1}^{K} C_k(N),\, t \in \mathcal{T}\} \tag{7.41}$$

where $C_k(N)$ is the subset of all subsets of $\{1,\ldots,N\}$ with cardinality k and $\mathcal{T} = \{\text{'buy'}, \text{'hold'}, \text{'sell'}\}$ is the set of values of the attributes that determine the trade that is made in the portfolio. The size of the search space is exponential in N

$$|\Theta| = 3\sum_{k=1}^{K} \binom{N}{k}. \tag{7.42}$$

The GA encodes the candidate solutions as sets of fixed cardinality. In this extended representation, each element in the set has an additional attribute whose value is in $\mathcal{T}$. The mutation operator exchanges a randomly selected product in the portfolio encoded by the candidate solution with another product that is not present in the portfolio. If the new product was not in the original portfolio, which is characterized by the vector of weights $\mathbf{w}^{(0)}$, the value of the trading attribute is set to 'buy'. Otherwise, a

random value in $\mathcal{T}$ for the trading direction attribute is assigned to this new element. A separate optimization is carried out for every possible value of the cardinality constraint in the range $k = 1, \ldots, K$. The best among the solutions obtained for the different optimizations is finally selected.

In Ruiz-Torrubiano et al. (2010), the performance of genetic algorithms that use different crossover operators specially designed for set encodings were compared in several cardinality constrained optimization problems. These included portfolio selection without transaction costs. The best overall results in this study were obtained by Random Assortment Recombination (RAR) (Radcliffe (1993)). In this section we propose to adapt this operator so that it can be applied to chromosomes with an extended set encoding. The resulting algorithm is referred to as extended RAR (eRAR). This extended version of RAR is detailed in Algorithm 16. The operator includes a positive integer parameter w that controls the amount of common information from the parents retained by the offspring. The RAR operator makes use of six sets: Set A is the intersection set, which contains products that appear in both parents. Set B includes the products not present in any parent. Sets C and D contain the products present in only one parent. Set E is initially empty ($E = \emptyset$). An additional set G is then created with w copies of the products from A and B and one copy from the products in C and D. The elements in G retain the label of the set from which they originate. A child chromosome is generated by selecting a product at random from G in each iteration. If the product originally comes from A or C and is not in E, then it is included in the child. Otherwise, if it originated in B or in D, then it is included in set E.

Algorithm 16 Extended Random Assortment Recombination algorithm (eRAR)

1. Create auxiliary sets A, B, C, D, E:
   - A = elements present in both parents.
   - B = elements not present in any of the parents.
   - $C \equiv D$ elements present in only one parent.
   - $E = \emptyset$.
2. Build set G with w copies of elements from A and B, and 1 copy of the elements in C and D.
3. Initialize child chromosome $\phi = \emptyset$.
4. While $|\phi| < n$ and $G \neq \emptyset$:
   - Extract $g \in G$ without replacement.
   - DetermineAttribute(g).
   - If $g \in A$ or $g \in C$, and $g \notin E$, $\phi = \phi \cup \{g\}$.
   - If $g \in B$ or $g \in D$, $E = E \cup \{g\}$.
5. If $|\phi| < n$, add elements not yet included chosen at random until chromosome is complete.

The process is terminated when the child has the specified cardinality or when $G = \emptyset$. If the latter happens, then the child is completed with assets selected at random from those which have not been included up to that moment. Note that this step allows the introduction in the child of products not present in any parent. The extended version eRAR handles the additional attribute that specifies the direction of the trade for each asset in the rebalancing operation by means of the function DetermineAttribute(g), which is described in Algorithm 17.

Algorithm 17 DetermineAttribute(g): Extended attribute selection in eRAR

1. If the product g is not present in the original portfolio, which is characterized by the weight vector $\mathbf{w}^{(0)}$, then the value of the attribute is 'buy'.
2. Otherwise
   (a) If the product g is not present in any parent, then the trading direction attribute is selected with equal probability in the set $\mathcal{T}$.
   (b) If the product g is present in only one parent, then the trading direction attribute of g in the child is set to the same value as in the parent.
   (c) If product g is present in both parents:
   - If the trading direction attribute in both parents is equal, the child has the same value of this attribute as its parents.
   - If the trading direction attribute is different in the parents the combination of attributes with the highest fitness is chosen.

When there is a disagreement between the attributes that determine the type of trade for that asset in the parents, we consider two strategies (i) Either we pick the one that has the highest fitness among all possible combinations of attributes or (ii) the value of the attribute in which the parents disagree is selected at random. The best performance corresponds to (i). However, the computations are unfeasible for large values of the cardinality constraint. Therefore, in the implementation of the algorithm procedure (ii) is used for values of the cardinality constraint higher than 15.

The fitness of the candidate solution is defined in terms of the solution of the subordinate optimization problem (7.32)

$$\text{Fitness}(\mathbf{z}) = -\min_{\mathbf{w}} \left(\mathbf{w}^{[\mathbf{z}]^{\mathrm{T}}} \cdot \boldsymbol{\Sigma}^{[\mathbf{z},\mathbf{z}]} \cdot \mathbf{w}^{[\mathbf{z}]} - \alpha \left(\mathbf{w}^{[\mathbf{z}]^{\mathrm{T}}} \cdot \hat{\mathbf{r}}^{[\mathbf{z}]} - \boldsymbol{\kappa}^{T} \cdot \left| \mathbf{w} - \mathbf{w}^{(\mathbf{0})} \right| \right) \right) \tag{7.43}$$

subject to (7.33)-(7.35), and one of the inequalities of (7.37) for each included product. A standard QP solver (Gill et al. (1991)) is used to address this subordinate optimization problem.

In the next section the effectiveness of this approach is illustrated in a series of experiments on actual financial data. In this empirical study the hybrid method is compared with a number of standard and advanced strategies. The main conclusion of the study is that cardinality constraints and transaction costs act as regularization

terms that allow the selection of sparse portfolios that are stable, robust and generally exhibit good out-of-sample performance.

7.4.3 Empirical evaluation

In this section we present the results of an empirical evaluation of the proposed hybrid method. The performance of this algorithm is compared with reference strategies for portfolio selection and management. Special attention is given to the effects of transaction costs and of cardinality constraints in both the in-sample and the out-of-sample performance of the portfolio. The experiments are carried out on three different datasets compiled by Fama and French[2]. These data consist of time series of non-annualized monthly returns from June 1971 until December 2009. The first dataset (FF48) includes 48 industry portfolios. The second one (FF100) is the intersections of 10 portfolios formed on size (market equity, ME) and 10 portfolios formed on the ratio of book equity to market equity (BE/ME). The 10 portfolios formed on size are constructed by ranking assets from small to large ME values and defining 10 deciles. A similar ranking is constructed based on the BE/ME values and, again, 10 deciles are defined. For each decile, a portfolio is constructed with the assets in that decile. Then, 100 portfolios are formed by constructing portfolios for all possible combinations of each decile of ME and BE/ME portfolios. In other words, one portfolio is formed by combining the decile i size portfolio with the decile j BE/ME portfolio, where $i, j = 1, \ldots, 10$. The third dataset (FF38) contains 38 industry portfolios different from those included in FF48.

The first part of the empirical study is devoted to in-sample evaluation. The goal of in-sample evaluation is to determine the quality of the memetic algorithm as an optimization method. The question is how close is the portfolio selected by this algorithm to the globally optimal portfolio. Given that the cardinality constrained portfolio optimization problem is NP-hard, only results relative to the best known solution can be given in most cases. Several studies have shown that portfolios that are optimal in-sample can have poor out-of-sample performance (see e.g. DeMiguel et al. (2009b)). The reason is that the inputs for the optimization are based on estimations that are insufficient or inadequate for prediction. In machine learning this discrepancy between in-sample (training) and out-of-sample (test) performance is referred to as overfitting. It is a result of erroneously identifying regularities in the data that are used to estimate the inputs to the optimization problem (training data) as patterns that are relevant to make predictions on an independent test set. The reliance on this spurious patterns is misleading and hinders the generalization capacity of the system (Bishop (2006)). For this reason, the second part of this subsection is devoted to the assessment of the out-of-sample performance of the hybrid method proposed in this research.

[2] http://mba.tuck.dartmouth.edu/pages/faculty/ken.french/data_library.html

7.4.3.1 In-sample evaluation

The in-sample performance is assessed using all the available data (approximately 400 monthly returns) to estimate the vector of expected returns

$$\hat{\mathbf{r}}_i = \frac{1}{T}\sum_{t=1}^{T} r_i(t), \qquad i = 1, \ldots, N \tag{7.44}$$

and the covariance matrix of these returns

$$\hat{\mathbf{\Sigma}}_{ij} = \frac{1}{T-1}\sum_{t=1}^{T} \left(r_i(t) - \hat{\mathbf{r}}_i\right)\left(r_j(t) - \hat{\mathbf{r}}_j\right) \ \forall i, j. \tag{7.45}$$

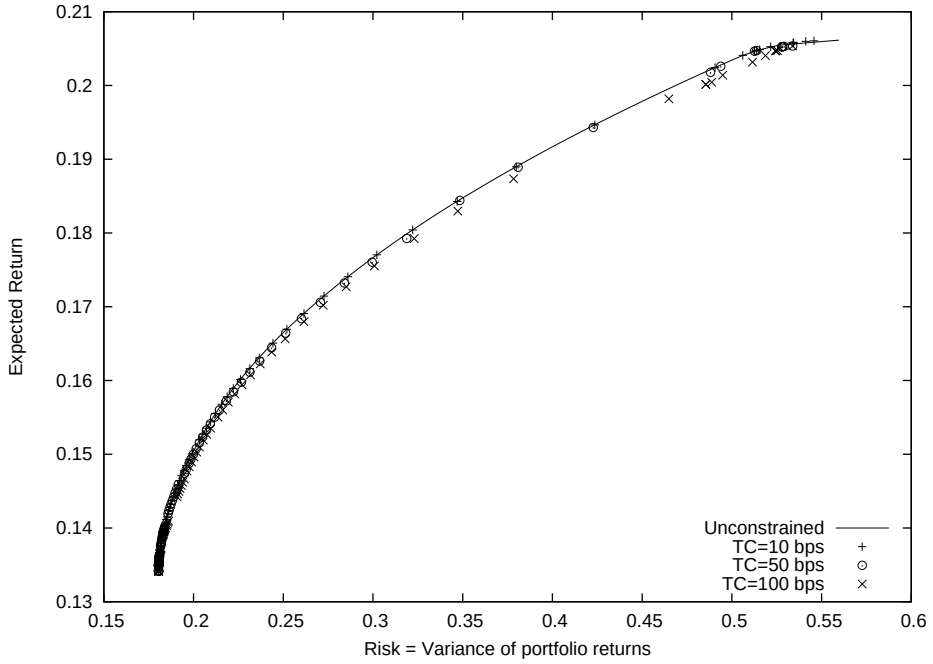
The hybrid metaheuristic introduced in the previous subsection is used to solve the optimization problem (7.32)-(7.37) with the sample estimates of the expected value (7.44) and the covariance matrix of the returns (7.45) as inputs. The GA uses a steady-state population of 100 individuals. Crossover is always performed. A child is generated by applying the eRAR operator to two parents selected in separate binary tournaments. In each binary tournament two individuals are picked at random. The best one is then selected for crossover. The newly generated offspring replaces the worst individual of the original population. The mutation operator described in the previous subsection is applied with probability 10^{-2}.

A first set of optimizations is made to calculate the *efficient frontier* of Pareto optimal portfolios. The efficient frontier is the collection of portfolios whose returns have the lowest possible variance for a fixed value of the expected portfolio return. From the dual perspective, Pareto optimal portfolios have a maximum expected return for a fixed value of the variance. These portfolios are the solution of the collection of optimization problems obtained by using in (7.32)-(7.37) as objective function

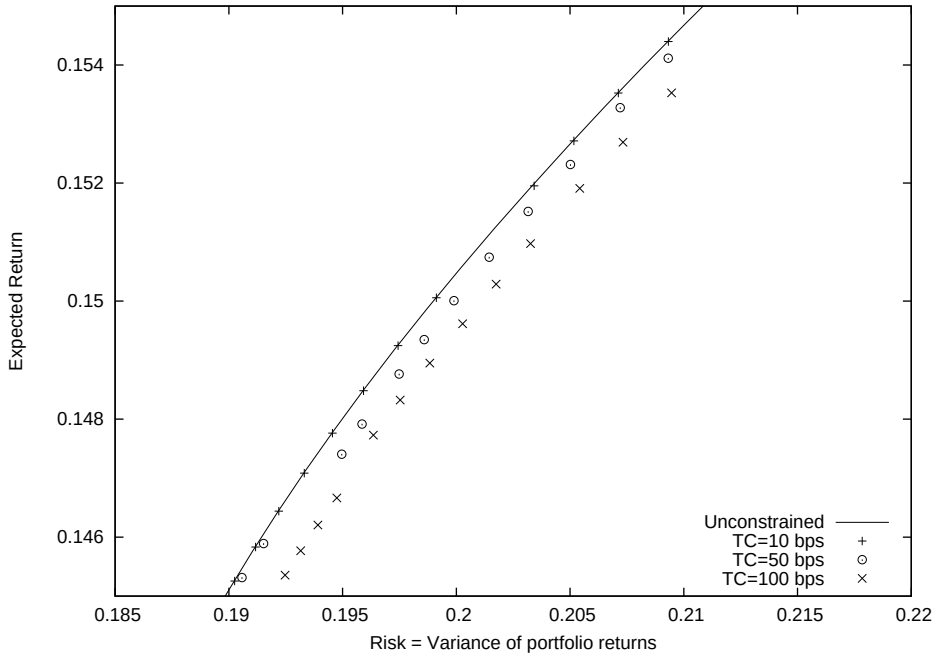
$$(1-\lambda)\mathbf{w}^{[\mathbf{z}]^{\mathrm{T}}} \cdot \mathbf{\Sigma}^{[\mathbf{z},\mathbf{z}]} \cdot \mathbf{w}^{[\mathbf{z}]} - \lambda(\mathbf{w}^{[\mathbf{z}]^{\mathrm{T}}} \cdot \hat{\mathbf{r}}^{[\mathbf{z}]} - \boldsymbol{\kappa}^{T} \cdot \left|\mathbf{w} - \mathbf{w}^{(\mathbf{0})}\right|). \tag{7.46}$$

The efficient frontier is parameterized in terms of $\lambda \in [0, 1]$. For the sake of simplicity, we assume equal transaction costs for all the assets $\{\kappa_i = \kappa,\ i = 1, \ldots N\}$. Taking into account different costs for different products is straightforward and does not increase the difficulty of the problem. The efficient frontiers are then computed for several values of the transaction costs as given by the value κ: 0, 10, 50 and 100 basis points[3]. In all cases, the portfolios are restricted to invest in at most $K = 10$ different products. We compute $N_F = 100$ portfolios in the efficient frontier by taking a grid of equidistant values of λ in the range $[0, 1]$. The efficient frontier that would be obtained if all the constraints were removed is also computed for reference. The efficient frontier for the FF48 dataset is displayed in Figure 7.5(a) and, in more detail in Figure 7.5(b). As expected, the solutions that are optimal when transaction costs are considered are dominated by the solutions on the unconstrained efficient frontier: the higher the transaction costs, the more distant is the efficient frontier from the unconstrained one. When low-to-moderate

[3] 1 basis point (bps) = 0.001%



(a) Comparison of efficient frontiers in the FF48 dataset for different values of the transaction costs.



(b) Detailed comparison of efficient frontiers in the FF48 dataset for different values of the transaction costs.

Figure 7.5: In 7.5(a) the whole efficient frontier is represented. In 7.5(b) a detailed comparison shows that the distance to the unconstrained efficient frontier increases with higher transaction costs.

Table 7.7: Comparison of in-sample results in the FF48 dataset using different values of the transaction costs.

Transaction costs	Best D	Success rate	Time (s)	Optimizations
0 bps	0.01378271	1.00	4262.4	$7.64 \cdot 10^7$
10 bps	0.13378017	1.00	4564.5	$8.91 \cdot 10^7$
20 bps	0.18238682	1.00	4487.2	$8.96 \cdot 10^7$
30 bps	0.27143231	1.00	4374.0	$9.05 \cdot 10^7$
40 bps	0.31587304	1.00	4379.6	$9.00 \cdot 10^7$
50 bps	0.36184243	1.00	4272.5	$8.95 \cdot 10^7$
100 bps	0.61696361	1.00	3920.7	$8.67 \cdot 10^7$

transaction costs are considered (10 and 50 bps), the efficient frontiers obtained are fairly close to the unconstrained one. Higher transaction costs tend to produce more distant frontiers, as can be seen in the 100 bps case.

Table 7.7 presents a number of measures that characterize the results of this optimization. In this table, we give results for transaction costs values $\kappa = 0$, 10, 20, 30, 40, 50 and 100 bps. The second column of this table displays the values of

$$D = \frac{1}{N_F} \sum_{i=1}^{N_F} \frac{\sigma_i^c - \sigma_i^*}{\sigma_i^*}, \tag{7.47}$$

which is a measure of the average relative distance between the actual and the unconstrained efficient frontiers. The value of σ_i^c in (7.47) is the standard deviation achieved for the ith point on the actual efficient frontier, which is obtained considering all the constraints and transaction costs, and σ_i^* is the corresponding value on the unconstrained efficient frontier. The third column in Table 7.7 presents the success rate obtained by the algorithm. The success rate is the fraction of runs of the algorithm in which the best known solution at each point in the frontier is found. In our experiments, the algorithm is executed 5 times for each point on the frontier. The run-time measured on an Intel Core Duo machine with 2 GHz clock speed and 2 GB RAM is given in the next column. Finally, the last column shows the total number of quadratic optimizations performed. The results in Table 7.7 confirm that increasing the transaction costs results in larger differences with the unconstrained efficient frontier, as measured by (7.47). The success rates, times and number of optimizations are similar in all cases. This means that, in the range of values considered, the difficulty of the optimization problem seems to be fairly independent of how large the transaction costs are.

In a second set of experiments we solve the optimization problem (7.32)-(7.37) using (7.44), (7.45) as inputs. The optimization is carried out with $\alpha = 2$ in (7.32). Similar conclusions are reached for other values of this parameter. The equally weighted $1/N$ portfolio is used as the initial portfolio in all cases

$$\mathbf{w}^{(0)} = \{1/N, 1/N, \ldots, 1/N\}. \tag{7.48}$$

The goal is to rebalance this portfolio so that it satisfies the specified constraints and has the best performance in one investment period, which in the experiments carried out has a duration of one month.

The performance is measured in terms of the expected return of the portfolios selected by the optimization procedure, taking into account the transaction costs

$$R_{exp} = \sum_{i=1}^{N} w_i \hat{\mathbf{r}}_i - \sum_{i=1}^{N} \kappa_i \left| w_i - w_i^{(0)} \right| \tag{7.49}$$

where $\mathbf{w} = \{w_i\}_{i=1}^{N}$ is the composition of the portfolio after rebalancing and $\hat{\mathbf{r}}_i$ are computed using (7.44). The in-sample Sharpe ratio (without taking into account the risk-free rate) is

$$S_R = \frac{R_{exp}}{\sigma} = \frac{\sum_{i=1}^{N} w_i \hat{\mathbf{r}}_i - \sum_{i=1}^{N} \kappa_i \left| w_i - w_i^{(0)} \right|}{\sqrt{\sum_{i=1}^{N} \sum_{j=1}^{N} w_i \hat{\mathbf{\Sigma}}_{ij} w_j}}, \tag{7.50}$$

in terms of the sample estimate of the covariance matrix of the returns (7.45).

The performance of the portfolio selected by the GA algorithm with an extended set encoding is compared to several portfolios that have been built using standard investment strategies. To analyze the effect of transaction costs on a general portfolio, assume that the composition of the portfolio immediately before rebalancing at time t is $\left\{ w_i^{(0)}(t), \quad i = 1, \dots N \right\}$, as in (7.27). The portfolio identified by strategy s after rebalancing at t is characterized by the weights

$$w_i^{(s)}(t), \quad i = 1, \dots N, \tag{7.51}$$

which are assumed to be normalized

$$\sum_{i=1}^{N} w_i^{(s)}(t) = 1. \tag{7.52}$$

Assuming linear transaction costs, the self financing constraint is

$$P(t^-) = P(t) + \sum_{i=1}^{N} \kappa_i \left| w_i^{(s)}(t) P(t) - w_i^{(0)}(t) P(t^-) \right|. \tag{7.53}$$

This implicit nonlinear equation can be solved to obtain $P(t)$, the value of the portfolio obtained by means of the investment strategy considered, as a function of the value of the portfolio before rebalancing $P(t^-)$. Taking into account the transaction costs, the expected return is

$$R_{exp}^{(s)}(t) = \frac{P(t)}{P(t^-)} \left(\sum_{i=1}^{N} w_i^{(s)}(t) \hat{\mathbf{r}}_i + 1 \right) - 1 = \frac{P(t)}{P(t^-)} \left(\sum_{i=1}^{N} w_i^{(s)}(t) \hat{\mathbf{r}}_i - \left(\frac{P(t^-)}{P(t)} - 1 \right) \right). \tag{7.54}$$

From the form of this expression, one can see that the effect of the transaction costs is, on the one hand, to lower the returns and, on the other hand, to reduce the amount of capital that is available for investment. The Sharpe ratio is

$$S_R^{(s)}(t) = \frac{\sum_{i=1}^{N} w_i^{(s)}(t)\hat{\mathbf{r}}_i - \left(\frac{P(t^-)}{P(t)} - 1\right)}{\sqrt{\sum_{i=1}^{N}\sum_{j=1}^{N} w_i^{(s)}\hat{\mathbf{\Sigma}}_{ij}w_j^{(s)}}}. \tag{7.55}$$

In the next series of experiments, the performance of the proposed eRAR strategy, taking into account transaction costs, is compared to the following five benchmark portfolios:

1. $1/N$: The naïve diversified portfolio in which all N products are given the same weight $1/N$. The transaction costs are ignored in the portfolio selection. As in all the cases considered the performance evaluation is made taking into account the actual transaction costs.

2. MinVar: The minimum variance portfolio. This portfolio is constructed by dropping the expected return constraint in the standard Markowitz model. The transaction costs are ignored in the portfolio selection.

3. NoCard: A portfolio built without the cardinality constraint but taking into account transaction costs. The problem can be formulated as a quadratic program in $3N$ dimensions by including two additional variables per asset: $d_i^+, d_i^- \in \mathbb{R}^+ \cup \{0\}$, $i = 1, \ldots, N$. Two new linear constraints per variable need to be included:

$$w_i - d_i^+ \leq w_i^{(0)} \tag{7.56}$$

$$d_i^- + w_i \geq w_i^{(0)}. \tag{7.57}$$

   The terms corresponding to the transaction costs in the objective function and in the constraint (7.33) are replaced by

$$\sum_{i=1}^{N} \kappa_i (d_i^+ + d_i^-). \tag{7.58}$$

   This strategy is referred to as the standard Markowitz portfolio in the discussion.

4. Lasso: This type of portfolio is obtained using the lasso approach described in Subsection 7.4.1 without the cardinality constraint. The value of $\boldsymbol{\gamma}^*$ used in the final evaluation is estimated by leave-one-out cross-validation: Let the training period be $[T_i, T_f]$. For each $t = T_i, \ldots, T_f$ we leave the t-th return out and use the resulting training set to select an optimal portfolio according to those data. The portfolio $\mathbf{w}^\gamma(t)$, obtained using the value γ for the lasso penalty, is held on $[t, t+1)$. Its out-of-sample return in that period $(r_{out}^\gamma(t))$ is then recorded. As a result of this process, we have a time series $\{r_{out}^\gamma(t)\}_{t=T_i}^{T_f}$. We then calculate the mean return of this series $\hat{r}_{out}^\gamma$ and choose $\gamma^* = \max_\gamma \hat{r}_{out}^\gamma$. Using the returns in

Table 7.8: Comparison of expected in-sample returns for the different strategies in the FF48 dataset.

TC	$1/N$ Ignore TC	MinVar Ignore TC	No Card. With TC	Lasso $\gamma = 3300$ bps	eRAR Ignore TC	eRAR With TC
0	0.0150000	0.011891	**0.015951**	0.014763	0.014812	0.014812
10	**0.015000**	0.010860	0.014456	0.014693	0.013038	0.012797
20	**0.015000**	0.009830	0.012975	0.014622	0.011267	0.010786
30	**0.015000**	0.008800	0.011510	0.014551	0.009500	0.008779
40	**0.015000**	0.007770	0.010060	0.014480	0.007737	0.006776
50	**0.015000**	0.006740	0.008624	0.014409	0.005977	0.004777

the first 60 months as training set, the value selected was $\boldsymbol{\gamma}^* = 3300$ bps for the FF48 dataset, $\boldsymbol{\gamma}^* = 3550$ bps for FF100 and $\boldsymbol{\gamma}^* = 250$ bps for FF38.

5. IgnoreTC: A portfolio constructed taking into account the cardinality constraint but ignoring transaction costs. The portfolio optimization is performed with zero transaction costs. However, the evaluation of the portfolio is made using the actual transaction costs. The proposed hybrid GA approach is used for this strategy.

In the tables in which the results of this empirical evaluation are presented, the best value is highlighted in boldface and the second best value is underlined.

The in-sample results for the FF48 dataset are shown in tables 7.8 and 7.9. In the absence of transaction costs, the strategy that obtains the best in-sample expected return is the standard Markowitz mean-variance portfolio (column "No Card"). When nonzero transaction costs are considered, the $1/N$ strategy, which does not incur transaction costs, has the best expected return. The second best expected returns without transaction costs correspond to the $1/N$ strategy. However, when transaction costs are considered, the second best results are obtained by the lasso strategy. This should be expected because the value of $\gamma = 3300$ bps estimated by cross-validation is quite large, which means that the the lasso and the $1/N$ portfolios are very similar. In terms of Sharpe ratios, the best results are obtained by the portfolios selected by eRAR without taking transaction costs into account. With transaction costs up to 20 bps, the No Card. strategy performs best. Above that value, the lasso strategy obtains the best results. Note that, in terms of expected returns, the strategy without cardinality constraints (No. Card.) always obtains better results than the cardinality-constrained eRAR strategies. This should be expected because the removal of a constraint necessarily improves the value of the optimum of the objective function.

Tables 7.10 and 7.11 summarize the in-sample results for the FF100 dataset. Without transaction costs, the standard Markowitz portfolio obtains the best in-sample returns, followed by the eRAR strategies. The lasso strategy is the best one when transaction costs are considered. The second best strategy is in this case the No Card. strategy, which, as in the previous case, obtains better results than the eRAR cardinality-constrained portfolios. In terms of Sharpe ratios, the best results for low transaction cost values are achieved by the eRAR strategies. However, the lasso strategy obtains the best Sharpe ratios with higher transaction costs

Table 7.9: Comparison of in-sample Sharpe ratios for the different strategies in the FF48 dataset.

TC	$1/N$ Ignore TC	MinVar Ignore TC	No Card. With TC	Lasso $\gamma = 3300$ bps	eRAR Ignore TC	eRAR With TC
0	0.018795	0.019705	0.026014	0.019448	**0.026185**	**0.026185**
10	0.018795	0.018016	**0.023637**	0.019357	0.023089	0.022658
20	0.018795	0.016324	**0.021270**	0.019265	0.019989	0.019127
30	0.018795	0.014628	0.018916	**0.019173**	0.016883	0.016222
40	0.018795	0.012929	0.016574	**0.019080**	0.013773	0.012053
50	0.018795	0.011227	0.014243	**0.018988**	0.010659	0.008510

Table 7.10: Comparison of in-sample expected returns for the different strategies in the FF100 dataset.

TC	$1/N$ Ignore TC	MinVar Ignore TC	No Card. With TC	Lasso $\gamma = 3550$ bps	eRAR Ignore TC	eRAR With TC
0	0.007800	0.007800	**0.012575**	0.011759	0.012435	0.012435
10	0.007800	0.007800	0.010994	**0.011714**	0.010584	0.010847
20	0.007800	0.007800	0.009460	**0.011669**	0.008716	0.009362
30	0.007800	0.007800	0.007953	**0.011624**	0.006863	0.007881
40	0.007800	0.007800	0.006487	**0.011579**	0.005013	0.005827
50	0.007800	0.007800	0.005055	**0.011534**	0.003167	0.005215

Table 7.11: Comparison of in-sample Sharpe ratios for the different strategies in the FF100 dataset.

TC	$1/N$ Ignore TC	MinVar Ignore TC	No Card. With TC	Lasso $\gamma = 3550$ bps	eRAR Ignore TC	eRAR With TC
0	0.010900	0.010900	0.022150	0.016730	**0.022714**	**0.022714**
10	0.010900	0.010900	0.019420	0.016667	0.019350	**0.019818**
20	0.010900	0.010900	0.016761	0.016604	0.015981	**0.017104**
30	0.010900	0.010900	0.014135	**0.016541**	0.012605	0.014395
40	0.010900	0.010900	0.011561	**0.016477**	0.009225	0.010659
50	0.010900	0.010900	0.009033	**0.016414**	0.005838	0.009518

The in-sample results for the FF38 dataset are summarized in tables 7.12 and 7.13. As in the previous cases, the best strategy with zero transaction costs is the No Card. strategy. For higher values of the transaction costs, the $1/N$ strategy has the best expected returns. The lasso strategy, which is again very similar to the $1/N$ strategy is the second best for higher transaction costs. The same observation as in the previous cases holds for the No Card. and the eRAR strategies: The model with no cardinality constraints obtains better expected returns. In terms of Sharpe ratios, the No Card. portfolios obtain the best results without transaction costs, followed by the eRAR strategies.

7.4.3.2 Out-of-sample evaluation

The out-of-sample performance of the different strategies is evaluated in a simulated investment exercise. We are given a collection of N assets from which an investment portfolio can be built. As in the in-sample case, the data available consist of time series of returns for each of these assets $\left\{\{r_i(t)\}_{t=1}^{T};\ i = 1, \ldots, N\right\}$. We fix a time horizon

Table 7.12: Comparison of in-sample expected returns for the different strategies in the FF38 dataset.

TC	$1/N$ Ignore TC	MinVar Ignore TC	No Card. With TC	Lasso $\gamma = 250$ bps	eRAR Ignore TC	eRAR With TC
0	0.014690	0.010892	**0.015055**	0.014611	0.014144	0.014144
10	**0.014690**	0.009579	0.013615	0.013447	0.012424	0.012752
20	**0.014690**	0.008267	0.012207	0.012284	0.010708	0.011166
30	**0.014690**	0.006955	0.010824	0.011123	0.008996	0.009011
40	**0.014690**	0.005645	0.009466	0.009964	0.007287	0.009524
50	**0.014690**	0.004336	0.008120	0.008805	0.005581	0.008522

Table 7.13: Comparison of in-sample Sharpe ratios for the different strategies in the FF38 dataset.

TC	$1/N$ Ignore TC	MinVar Ignore TC	No Card. With TC	Lasso $\gamma = 250$ bps	eRAR Ignore TC	eRAR With TC
0	0.018516	0.019355	**0.025376**	0.024823	0.025366	0.025366
10	0.018516	0.017044	**0.022996**	0.022871	0.022320	0.022957
20	0.018516	0.014728	0.020658	**0.020918**	0.019270	0.020197
30	0.018516	0.012408	0.018353	**0.018962**	0.016216	0.016222
40	**0.018516**	0.010084	0.016080	0.017005	0.013158	0.017063
50	**0.018516**	0.007755	0.013822	0.015045	0.010095	0.015245

$t_{tr} \leq T$ that determines the amount of training data. The expected returns used as input in the optimization are estimated from these training data

$$\hat{\mathbf{r}}_i = \frac{1}{t_{tr}} \sum_{t=1}^{t_{tr}} r_i(t) \qquad i = 1, \ldots, N. \tag{7.59}$$

The sample estimate of the covariance matrix of these returns is

$$\hat{\mathbf{\Sigma}}_{ij} = \frac{1}{t_{tr} - 1} \sum_{t=1}^{t_{tr}} \left(r_i(t) - \hat{\mathbf{r}}_i\right) \left(r_j(t) - \hat{\mathbf{r}}_j\right) \ \forall i, j. \tag{7.60}$$

The equally weighted portfolio

$$\mathbf{w}_s^{(0)}(t_{tr}^-) = \{1/N, 1/N, \ldots, 1/N\} \tag{7.61}$$

is the initial portfolio for all strategies s that are analyzed. For simplicity, equal transaction costs are assumed for all assets $\{\kappa_i = \kappa;\ i = 1, \ldots, N\}$. We then select a portfolio $\mathbf{w}^s(t_{tr})$ using each of the strategies considered. The composition of this portfolio is then held fixed for the period $[t_{tr}, t_{tr} + 1)$. Even if the composition of the portfolio does not change, the portfolio weights evolves during this period because of changes in the market prices of its constituents. The training data window is then shifted by one month. The portfolio that results from the evolution of the market prices of the constituent assets is rebalanced using as inputs the expected means and covariance matrix of the asset returns estimated on the data from the shifted time window. The process is repeated until the last period of data available.

Consider the portfolio selected by strategy s after rebalancing at time t, which is characterized by the vector of weights $\mathbf{w}^{(s)}(t) = \left\{w_i^{(s)}(t);\, i = 1, \ldots, N\right\}$. When no transaction costs are considered to compute these weights the cost-adjusted return of the portfolio in the period $[t, t+1)$ is

$$R^{(s)}(t) = \frac{P(t)}{P(t^-)} \left(\sum_{i=1}^{N} w_i^{(s)}(t) r_i(t) - \left(\frac{P(t^-)}{P(t)} - 1 \right) \right),$$

where $r_i(t)$ are the actual returns for the ith asset in that period, $P(t^-)$ is the value of the portfolio before rebalancing at t, and $P(t)$ is the value of the portfolio after rebalancing. The procedure for the computation of $P(t)$ from $P(t^-)$ and $\mathbf{w}^{(s)}(t)$ has been described in the section on in-sample evaluation.

When the portfolio weights $\mathbf{w}^{(s)}(t)$ are computed taking into account transaction costs, the portfolio return in $[t, t+1)$ is

$$R^{(s)}(t) = \sum_{i=1}^{N} w_i^{(s)}(t) r_i(t) - \sum_{i=1}^{N} \kappa_i \left| w_i^{(s)}(t) - w_i^{(0)}(t) \right|, \tag{7.62}$$

where $w_i^{(0)}(t)$ are the (normalized) portfolio weights immediately before rebalancing at time t. Note that $w_i^{(0)}(t)$ will in general be different from $w_i^{(0)}(t-1)$. These weights evolve during the period $[t-1, t)$ because of changes in the market prices of the assets in the portfolio.

The accumulated return in the testing (out-of-sample) period $[t_{tr}+1, T]$ is

$$R_{acc}^{(s)}(t_{tr}+1, T) = \frac{\mathbb{E}\left[P(T)\right]}{P(t_{tr})} - 1 = \prod_{t=t_{tr}+1}^{T} (1 + R^{(s)}(t)) - 1. \tag{7.63}$$

The average Sharpe ratio is

$$S_{av}^{(s)}(t_{tr}+1, T) = \frac{\mathrm{Av}\left[\left\{R^{(s)}(t)\right\}_{t=t_{tr}+1}^{T}\right]}{\mathrm{Stdev}\left[\left\{R^{(s)}(t)\right\}_{t=t_{tr}+1}^{T}\right]}. \tag{7.64}$$

In this expression the numerator represents the sample average and the denominator the sample standard deviation of the time series of portfolio returns. To quantify the amount of trading that is performed, the average turnover in terms of normalized weights

$$T^s(t_{tr}+1, T) = \frac{1}{T - t_{tr} + 1} \sum_{t=t_{tr}}^{T-1} \sum_{i=1}^{N} \left| \frac{w_i^{(s)}(t+1)}{\sum_{j=1}^{N} w_j^{(s)}(t+1)} - \frac{w_i^{(s)}(t)}{\sum_{j=1}^{N} w_j^{(s)}(t)} \right| \tag{7.65}$$

is also computed.

In the experiments performed, 5 years of data are used for training. The first training period is from June 1971 until July 1976. The testing period runs up to December 2009, the last month for which data are available. The performance of the portfolios selected by the genetic algorithm with set encoding, cardinality constraints and transaction costs,

are compared with the five benchmark portfolios described in the section on in-sample results. We also report the results for a PASSIVE strategy, in which the composition of the portfolio is not changed with time. Initially the N assets have the same weight $(1/N)$. Even though the composition of the portfolio does not change, the asset weights change because of the evolution of their market prices. It is interesting to benchmark against the passive strategy because it does not involve any rebalancing and therefore does not incur transaction costs.

The parameters used in the GA optimizations are the same as those used for in-sample evaluation (Section 7.4.3.1). Table 7.14 displays the accumulated returns for the different strategies in the FF48 dataset. The corresponding average Sharpe ratios are shown in Table 7.15. The values reported correspond to an investment period from June 1976 until December 2009. The accumulated returns should therefore be interpreted as the accumulated profit (final minus initial portfolio wealth) at the end of December 2009 that results from an investment of 1$ at the beginning of June 1976. The results are calculated for transaction costs that range between 0 and 50 bps.

The first important observation is that the portfolio built using the eRAR strategy but without taking into account transaction costs when rebalancing the portfolio has in most cases a lower accumulated return than the same strategy with transaction costs, which also has smaller values of the average turnover. In fact, this strategy has larger accumulated return than all the portfolios that are selected using strategies that ignore transaction costs. The minimum variance portfolios are always better than $1/N$ portfolios both in terms of accumulated return and of average Sharpe ratios. The passive portfolio has fairly high expected returns, better than the standard mean-variance optimal, the min-variance and the $1/N$ portfolios, and only slightly worse than the Lasso portfolio. Another important observation is that including cardinality constraints improves the out-of-sample performance in most cases. Nonetheless, the cardinality constraint by themselves are not sufficient. One also needs to take transaction costs into account. The eRAR strategy that considers transaction costs has the best accumulated return in most cases.

The average values of the Sharpe ratios are presented in Table 7.15. Despite the differences in accumulated returns, all the portfolios, except for the passive and the $1/N$ strategies, have similar values of this performance measure. Despite their lower accumulated return, portfolios that do not have a cardinality constraint generally have lower variances, as a result of diversification. The best values of the average Sharpe ratio correspond to the MinVar strategy. This means that minimizing the variance in-sample is an effective strategy to minimize the out-of-sample variance. The second largest values of the Sharpe ratios are achieved by the eRAR strategy with transaction costs.

The values of the average turnover of the different portfolios is shown in table 7.16. The $1/N$ strategy has the largest overall turnover: the portfolio needs to be continuously rebalanced to compensate the changes in portfolio weights resulting from the changes in the prices of the assets in the portfolio. This explains the poor performance of this portfolio when transaction costs are taken into account. The passive strategy, which does not involve any rebalancing, has zero turnover. The lasso strategy, in which trades are penalized, has a small turnover. As expected, in the strategies that take into account

Table 7.14: Accumulated returns for the different strategies in the FF48 dataset.

TC	Passive No TC	$1/N$ Ignore TC	MinVar Ignore TC	No Card. With TC	Lasso $\gamma = 3300$ bps	eRAR Ignore TC	eRAR With TC
0	95.670663	86.276413	89.350936	86.813818	97.189846	**128.155098**	**128.155098**
10		73.408415	88.132098	83.336660	97.104098	119.855927	**125.369070**
20		62.436569	86.929599	80.114928	97.018307	**112.087376**	109.483665
30		53.081622	85.743219	77.145857	96.932477	104.365905	**111.592719**
40		45.105426	84.572742	74.190147	96.846614	98.009242	**105.724173**
50		38.304883	83.417955	71.356740	96.760665	91.178698	**99.803953**

Table 7.15: Average Sharpe ratios for the different strategies in the FF48 dataset.

TC	Passive No TC	$1/N$ Ignore TC	MinVar Ignore TC	No Card. With TC	Lasso $\gamma = 3300$ bps	eRAR Ignore TC	eRAR With TC
0	0.263390	0.269102	**0.364336**	0.322772	0.304987	0.346159	0.346159
10		0.260463	**0.363313**	0.320258	0.304931	0.341792	0.344805
20		0.251818	**0.362288**	0.317803	0.304874	0.337416	0.338135
30		0.243168	**0.361262**	0.315423	0.304818	0.332688	0.339387
40		0.234514	**0.360235**	0.312971	0.304761	0.328638	0.335418
50		0.225856	**0.359207**	0.310449	0.304704	0.323845	0.330935

Table 7.16: Average turnover for the different strategies in the FF48 dataset.

TC	Passive No TC	$1/N$ Ignore TC	MinVar Ignore TC	No Card. With TC	Lasso $\gamma = 3300$ bps	eRAR Ignore TC	eRAR With TC
0	0.000000	0.474748	0.044916	0.092204	0.002220	0.188056	0.188056
10				0.088368			0.139953
20				0.085043			0.123841
30				0.082032			0.106235
40				0.079307			0.100318
50				0.076869			0.092705

transaction costs in the selection of optimal portfolio weights (No Card., eRAR with TC), the average turnover decreases with increasing transaction costs.

To investigate the regularization effects of cardinality constraints and of L_1 penalties proportional to the absolute value of the difference between the weights of the rebalanced portfolio and of the original portfolio, we compare the out-of-sample accumulated returns for lasso portfolios with cardinality constraints $K = 10$, $K = 20$ and without a cardinality constraint. Figure 7.6 displays the accumulated return of these lasso portfolios as a function of the value of $\boldsymbol{\gamma}$ used to train the model. Using either cardinality constraints or high lasso penalties ($\gamma \approx 2500$ bps) are useful strategies that can be used to select portfolios with good out-of-sample performance. However, using *both* cardinality constraints and a high lasso penalty seems to be detrimental for the out-of-sample performance. From these results we conclude that including both types of regularization is not an effective strategy in the problems investigated.

To illustrate the evolution of the porftolios that are selected when both transaction costs and cardinality constraints are considered, we present results in an investment universe of $N = 3$ assets. The portfolio is restricted to have at most $K \leq 2$ assets at a given instant. Figure 7.7 displays the evolution of the portfolio for transaction costs $\kappa = 0, 25, 50, 75, 150, 250$ bps. Several features of the evolution of the investment are

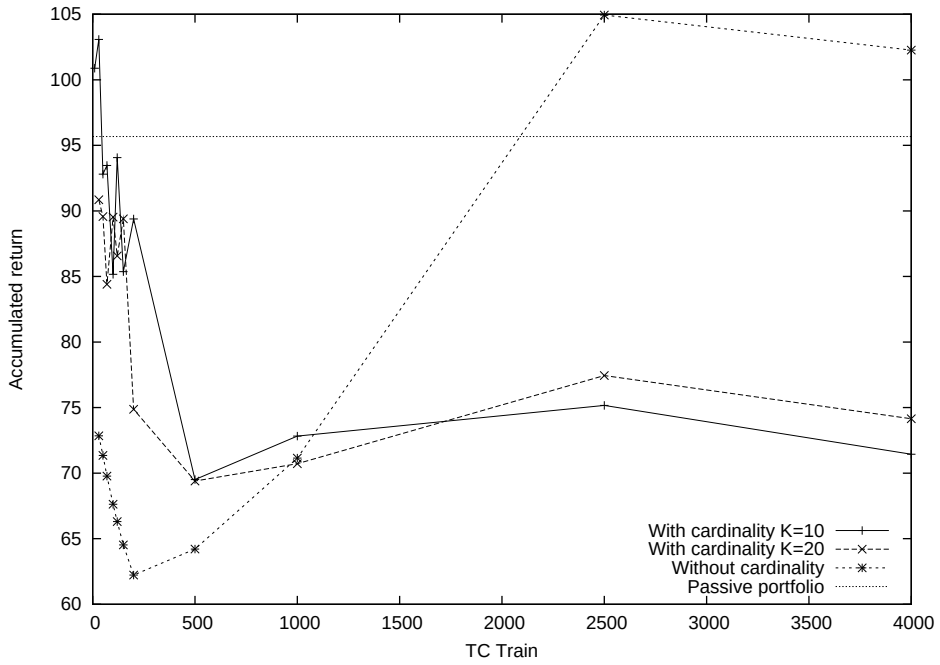
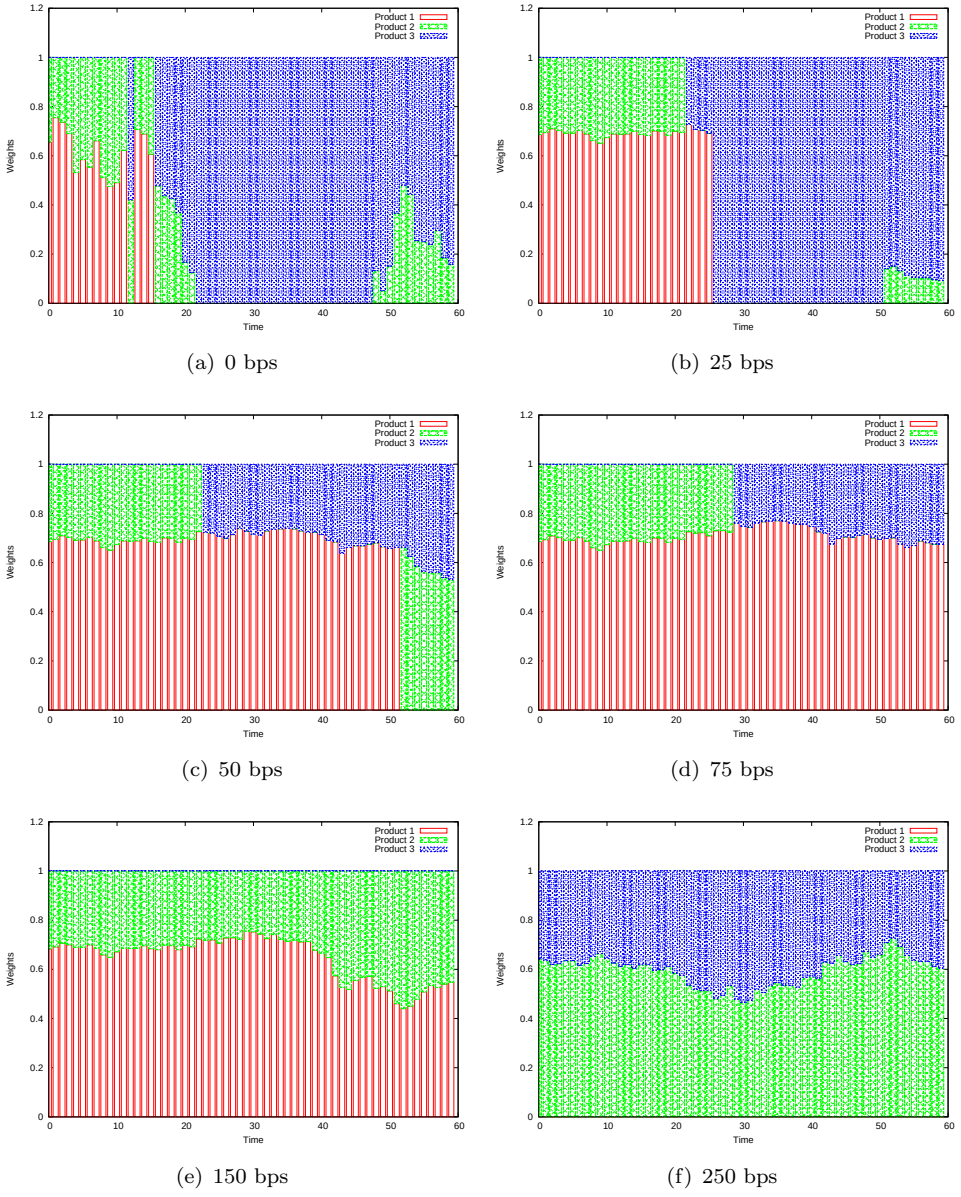


Figure 7.6: Accumulated returns as a function of the transaction costs used for training. Transaction costs in test are set to 50 bps.

noteworthy: For low costs the assets included in the portfolio change often to take advantage of local trends. For larger transaction costs, the changes in the weights of the portfolio are smaller and the positions in a given asset are held longer. The composition is modified only if the trend detected is strong. In the experiments performed this is illustrated by the fact that the composition of the portfolios changes when the transaction costs are low (0-75 bps). In contrast, the when the transaction costs are higher (150 and 250 bps), the portfolios invest in the same two assets during the whole investment period.

The out-of-sample performance measures for the FF100 dataset are presented in Tables 7.17, 7.18 and 7.19. Since the number of assets is much higher in this case, we use a cardinality constraint of $K = 25$. The best results are obtained by the lasso and the passive strategies. From the remaining strategies, the eRAR portfolios with a cardinality constraint show a good overall performance, although clearly inferior to the passive or the lasso strategies. Portfolios that do not consider transaction costs have lower accumulated returns. The fact that these portfolios have lower variance (because they are more diversified) means that the average Sharpe ratios are only slightly inferior. From the results one also concludes that it is crucial to take into account the effects of transaction costs in the optimization. The performance obtained by portfolios that are selected by strategies that ignore transaction costs significantly deteriorates with increasing transaction costs. The eRAR portfolios with transaction costs exhibit a good overall performance.

Out-of-sample performance measures for the dataset FF38 are given in tables 7.20, 7.21 and 7.22. The conclusions are similar to the ones obtained from the results in the FF100 dataset. The lasso strategy is the best strategy in terms of accumulated



(a) 0 bps (b) 25 bps

(c) 50 bps (d) 75 bps

(e) 150 bps (f) 250 bps

Figure 7.7: Evolution of the weights for different transaction costs in a 3 product universe.

Table 7.17: Accumulated returns for the different strategies in the FF100 dataset.

TC	Passive No TC	$1/N$ Ignore TC	MinVar Ignore TC	No Card. With TC	Lasso $\gamma = 3550$ bps	eRAR Ignore TC	eRAR With TC
0		48.913566	48.913595	58.043442	**116.864211**	65.097650	65.097650
10		41.973967	48.471808	55.467468	**116.820847**	57.519813	60.885255
20	110.500376	35.998316	48.033867	53.010446	**116.777430**	50.858165	60.464414
30		30.852828	47.599738	50.854126	**116.733955**	44.951999	60.337586
40		26.422274	47.169388	49.061071	**116.690428**	39.715959	62.201936
50		22.607402	46.742783	47.423537	**116.646838**	35.074299	59.685210

Table 7.18: Average Sharpe ratios for the different strategies in the FF100 dataset.

TC	Passive No TC	$1/N$ Ignore TC	MinVar Ignore TC	No Card. With TC	Lasso $\gamma = 3550$ bps	eRAR Ignore TC	eRAR With TC
0	0.301013	0.253470	0.253470	0.276608	**0.320164**	0.278596	0.278596
10		0.244719	0.252947	0.273972	**0.320139**	0.270960	0.274496
20		0.235964	0.252424	0.271277	**0.320115**	0.263753	0.273100
30		0.227206	0.251901	0.268755	**0.320091**	0.256527	0.274703
40		0.218443	0.251377	0.266597	**0.320066**	0.249285	0.277696
50		0.209678	0.250854	0.264584	**0.320042**	0.242025	0.276162

Table 7.19: Average turnover for the different strategies in the FF100 dataset.

TC	Passive No TC	$1/N$ Ignore TC	MinVar Ignore TC	No Card. With TC	Lasso $\gamma = 3550$ bps	eRAR Ignore TC	eRAR With TC
0	0.000000	0.444945	0.024699	0.122999	0.001260	0.334983	0.334983
10				0.116488			0.256403
20				0.110776			0.214784
30				0.105754			0.188396
40				0.101169			0.159414
50				0.097206			0.154017

Table 7.20: Accumulated returns for the different strategies in the FF38 dataset.

TC	Passive No TC	$1/N$ Ignore TC	MinVar Ignore TC	No Card. With TC	Lasso $\gamma = 250$ bps	eRAR Ignore TC	eRAR With TC
0	85.489189	88.216696	83.930429	68.433229	**108.195586**	100.819826	100.819826
10	85.489189	87.358726	82.615493	65.320255	**107.987756**	95.417455	96.622172
20	85.489189	86.508980	81.320665	62.357596	**107.780255**	90.300079	91.269580
30	85.489189	85.667380	80.045640	59.576223	**107.573084**	85.452748	86.713065
40	85.489189	84.833848	78.790119	57.023553	**107.366241**	80.861293	86.898443
50	85.489189	84.008308	77.553806	54.635601	**107.159726**	76.512285	80.780989

Table 7.21: Average Sharpe ratios for the different strategies in the FF38 dataset.

TC	Passive No TC	$1/N$ Ignore TC	MinVar Ignore TC	No Card. With TC	Lasso $\gamma = 250$ bps	eRAR Ignore TC	eRAR With TC
0	0.257746	0.269579	**0.384662**	0.317762	0.349240	0.342409	0.342409
10		0.269058	**0.383389**	0.314790	0.349106	0.338656	0.339036
20		0.268536	**0.382114**	0.311862	0.348972	0.334895	0.335146
30		0.268014	**0.380837**	0.308960	0.348838	0.331126	0.331543
40		0.267493	**0.379557**	0.306183	0.348703	0.327351	0.333211
50		0.266971	**0.378275**	0.303434	0.348568	0.323569	0.326676

out-of-sample returns. The portfolio selected by eRAR with transaction costs also has large accumulated returns, although they are inferior to the lasso. As in the previous cases eliminating the cardinality constraint or ignoring transaction costs leads to the selection of unregularized portfolios that have lower accumulated returns. In contrast, the average Sharpe ratios do not exhibit this effect. As a matter of fact, the minimum variance portfolio has the best average Sharpe ratio. This is because these types of portfolio are more diversified, and, in consequence, tend to have a lower variance.

Table 7.22: Average turnover for the different strategies in the FF38 dataset.

TC	Passive No TC	$1/N$ Ignore TC	MinVar Ignore TC	No Card. With TC	Lasso $\gamma = 250$ bps	eRAR Ignore TC	eRAR With TC
0	0.000000	0.027791	0.050512	0.059326	0.005024	0.159599	0.119182
10				0.047544			0.089804
20				0.038947			0.080920
30				0.032244			0.073050
40				0.027461			0.070214
50				0.023389			0.066470

7.4.3.3 Discussion

When transaction costs are not considered, the naïvely diversified ($1/N$) portfolio has good out-of-sample performance (DeMiguel et al. (2009b)). However, the performance of the $1/N$ portfolio quickly deteriorates when transaction costs are considered. The reason is that a very active trading strategy is needed to compensate for the changes in portfolio weights that result from the evolution of the market prices of the assets in the portfolio. This strategy has large turnover rates and incurs high transaction costs. A better benchmark when transaction costs are considered is the passive strategy. Since no rebalancing is performed, one does not incur any transaction costs. Asymptotically, for long investment periods, the portfolio is dominated by the best performing assets. This means that, in practice, the expected return from this investment is large. However, the variance of the portfolio returns also tend to be large because of the lack of diversification.

From the results of the empirical study carried out, the observation that in-sample performance is not necessarily a good estimate of the out-of-sample performance is confirmed. To obtain good out-of-sample performance one needs to include some form of regularization in the optimization. This regularization can be in the form of terms in the objective function that penalize excessive portfolio rebalancing in response to spurious trends in the training data, or of cardinality constraints. Exploratory experiments show that including both types of regularization does not seem to be an effective strategy. Nonetheless a more extensive evaluation should be carried out to provide further evidence of this observation. Besides the passive strategy, the best out-of-sample returns are obtained by portfolios that are built using regularization: the lasso strategy and the eRAR strategy that takes into account the actual transaction costs and also considers cardinality constraints.

In terms of Sharpe ratios, the differences between regularized and non-regularize strategies are smaller. In particular, the average Sharpe ratios of minimum variance portfolios are generally among the best. These portfolios are well diversified and their out-of-sample returns have in general low variance. The standard mean-variance optimal portfolio, which has an excellent in-sample performance has a poor out-of-sample performance in all the cases investigated. This can be ascribed to some form of overfitting to the training data (DeMiguel et al. (2009b)).

7.5 Summary and Discussion

In this chapter we have presented several metaheuristics for the problem of optimal portfolio selection with cardinality constraints. Additionally, an adaptation of the RAR crossover operator that operates with extended sets has been used to solve an extension of the problem that takes into account transaction costs and turnover constraints. It has been shown that a preprocessing step that discards products with a low probability of being present in the optimal solution is a useful way to drastically improve performance. Another conclusion is that GAs with a set encoding and specially designed mutation and crossover operators generally performs better than a GA with binary encoding and standard genetic operators. This illustrates that the use of forma theory is useful to design genetic representations and operators with improved performance. Regarding EDAs, the difficulties in exploring high-dimensional search spaces suggests that their practical applicability to real portfolio selection problems is very limited. EDAs require the application of preprocessing steps in which the dimensionality of the problem is reduced to be competitive with GAs or SA.

One of the main contributions of this chapter is the adaptation of the RAR crossover operator to manipulate the additional attributes in the genetic representation that specify the trade direction during rebalacing. This extended RAR crossover (eRAR) allows to handle the transaction costs, cardinality constraints and minimum trading size restrictions in such a way that the offspring generated are always feasible. Analyzing the results of the extensive empirical evaluation performed, we conclude that it is important to incorporate transaction costs explicitly in the optimization to obtain portfolios that have good in-sample, but specially out-of-sample performance.

In summary, the conclusions of this chapter are:

- Hybrid methods based on a set-encoding for the candidate solutions (RAR-GA, TransRAR-GA and SA) are effective and efficient methods for solving the portfolio selection problem with cardinality constraints. The performance of these algorithms does not deteriorate with the size of the problems in the cases investigated. These methods have been used to identify near-optimal portfolios even in problems with hundreds of assets.

- EDA approaches, which are based on estimating the distribution of a population of individuals, perform poorly when the universe of assets available for investment is very large, due to the *curse of dimensionality*: As the number of variables in the problem increases, the estimation and sampling of the probability distributions is not accurate enough to provide reliable guidance to the search.

- The efficiency and accuracy of the different hybrid approaches to portfolio selection with cardinality constraints considered can be significantly improved when pruning techniques are used to reduce the number of variables in the problem. These pruning heuristics work by eliminating products that have zero or low weights in optimal solutions to relaxed versions of the problem that can be solved numerically in an exact manner.

- Another important conclusion is that including a cardinality constraint can also improve the out-of-sample performance of the portfolio. In general, portfolios with cardinality constraints have better out-of-sample performance than portfolios that invest in all assets. In summary, both transaction costs and cardinality constraints can be seen as regularization strategies that allow the identification of stable and robust portfolios with good out-of-sample performance.

CHAPTER 8

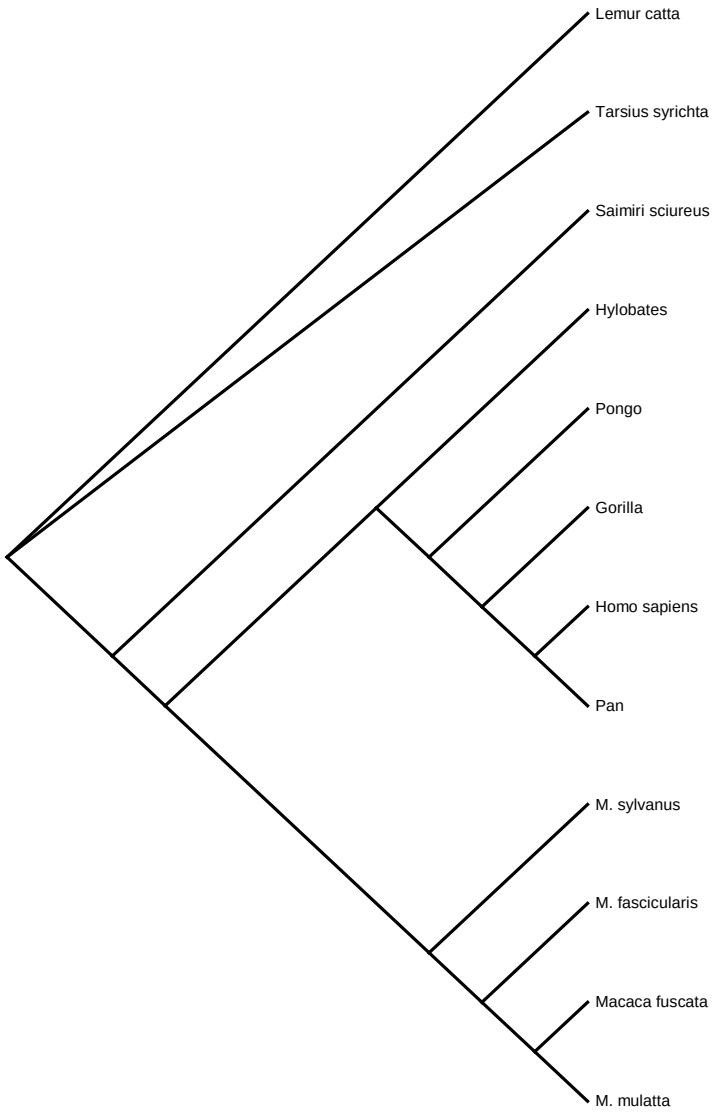
THE CONSENSUS TREE PROBLEM

Phylogenetic trees are hierarchical structures that reflect the relations among a set of taxa based on their evolutionary proximity. Different phylogenetic trees can be obtained from the same genetic information using different methods. The Consensus Tree Problem consists in building a single tree that optimally summarizes the information in these phylogenetic trees. Obtaining this optimal consensus tree is a complex combinatorial optimization problem for which both exact and metaheuristic approaches exist. We present dimensionality reduction techniques and hybrid metaheuristic approaches that can be used to obtain high quality consensus trees with a reduced computational cost.

8.1 Introduction

The goal of phylogenetics is to determine the relations between groups of organisms in terms of their genetic proximity, usually expressed by phylogenetic trees. A phylogenetic tree captures information on the evolutionary history of a set of *taxa*. Taxa can either be a group of species, populations of the same or distinct species, or homologous genes in populations of different species (Andreatta and Ribeiro (2002)). This information is represented in the phylogenetic tree by the evolutionary distance among the set of taxa under consideration. An example of a phylogenetic tree with 12 taxa is given in Figure 8.1. From this tree we can infer, for instance, that the taxa labeled as "Homo sapiens" and "Hylobates" are believed to be more closely related than "Homo sapiens" and "Lemur catta". The problem of phylogeny consists in building phylogenetic trees from molecular information, such as DNA sequences (Holmes (1999)). Several criteria can be used to build phylogenetic trees (Kim and Warnow (1999)). For instance, one can formulate a stochastic evolution model. The *maximum likelihood* criterion selects the tree that maximizes the likelihood of the model given the data. The *maximum parsimony* criterion minimizes the number of evolutionary changes needed to explain the observed data. The *distance-based* criterion uses a distance matrix to create the tree which best represents the degree of closeness between any pair of taxa. All of these

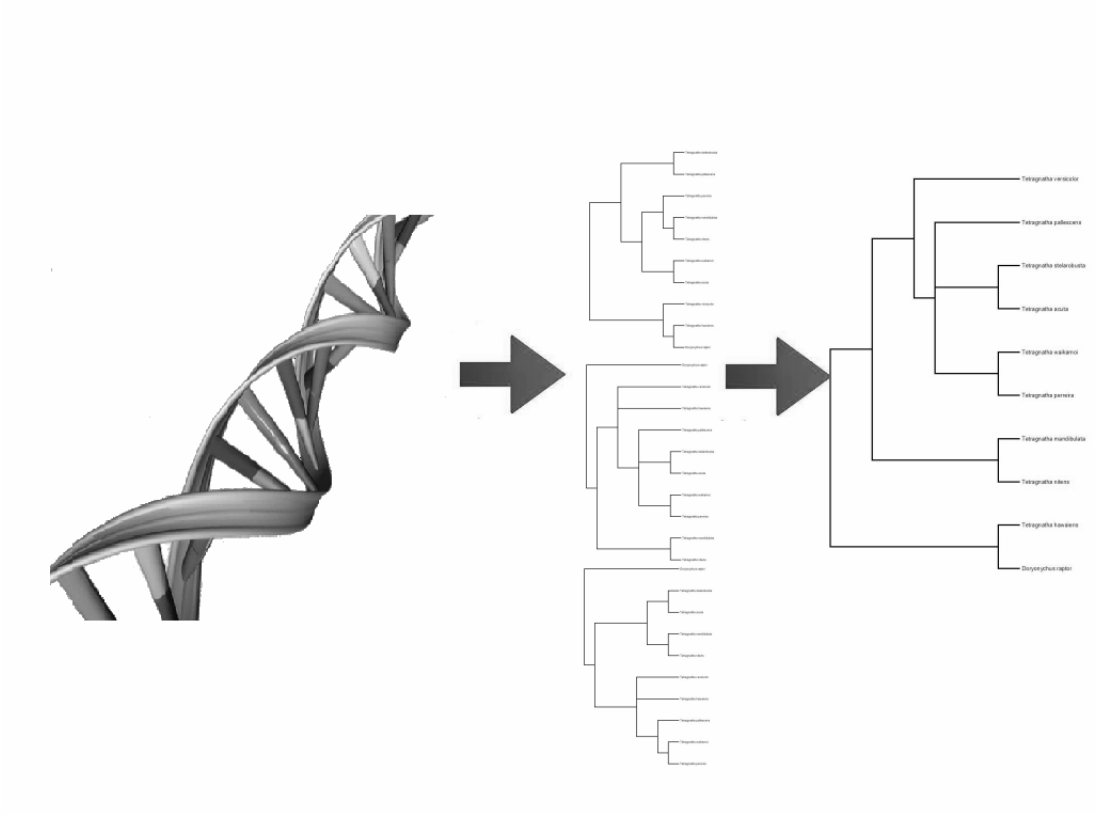
Figure 8.1: A phylogenetic tree with 12 taxa representing the relations among several species of primates.



approaches can be proven to yield NP-complete problems (Foulds and Graham (1982), Day et al. (1986)).

Using the different criteria different phylogenetic trees can be built. Each of these trees may reveal particular features of the true underlying solution. Our objective is to obtain the *consensus tree*, the tree that best integrates the information and structures present in the available trees. Since these trees have been generated from the same genetic information, they are unlikely to have common structures that are spurious.

Figure 8.2: A schematic representation of the Consensus Tree Problem. From molecular genetic information (i.e. DNA sequences) several phylogenetic trees are produced. The consensus tree is obtained from the input tree collection by solving a combinatorial optimization problem.

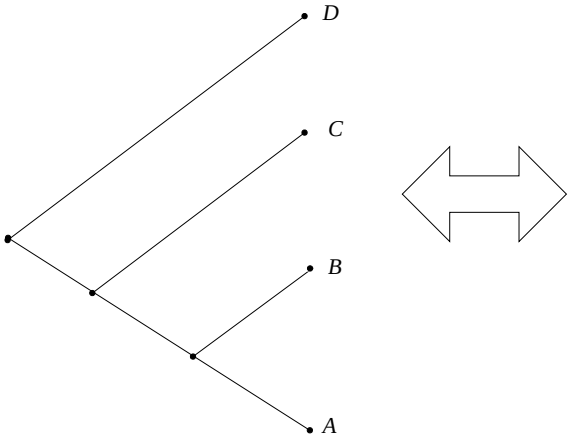


Therefore it is likely that these common structures are also a part of the true solution (Swofford (1991)). Figure 8.2 displays a schematic representation of the process: The consensus tree condenses the information present in the collection of phylogenetic trees obtained from the original molecular information.

In this chapter, we propose several exact and hybrid approaches to address the consensus tree problem. Based on its formulation as an integer linear program (ILP), solution methods from the branch-and-bound family are considered. These are combined with heuristics and metaheuristics to obtain high-quality consensus trees. The effectiveness of the different techniques analyzed is evaluated in synthetic and actual phylogenetic trees.

This chapter is organized as follows: In Section 8.2, several criteria that can be used to build consensus trees are introduced. Section 8.3 describes different strategies that can be used to obtain a solution to the problem. These include dimensionality reduction techniques (pruning of variables), using lazy constraints, improving of incumbent solutions for the branch-and-bound method using metaheuristics and heuristic column generation. Extensive computational tests are detailed in Section 8.4. A summary of the results and conclusions for this chapter is given in Section 8.5.

Figure 8.3: Two equivalent representations for a tree: rooted triplets and UpDown matrix.



	A	B	C	D
A	0	1	2	3
B	1	0	2	3
C	1	1	0	2
D	1	1	1	0

8.2 Optimization model

A phylogenetic tree can be modeled by a rooted binary tree and described in terms of *rooted triplets*: Let $\mathcal{L} = \{1, \ldots, n\}$ be the set of taxa. A rooted triplet $(a, b|c) \in \mathcal{L} \times \mathcal{L} \times \mathcal{L}$ represents the fact that the least common ancestor of a and b is a descendant of the least common ancestor of a, b and c. Consider, for instance, the tree in figure 8.3. This tree has four taxa: A, B, C and D. It can be represented by three rooted triplets: $(A, B|C)$, $(A, B|D)$, $(B, C|D)$. An alternative representation of a rooted tree is the *UpDown matrix*. This structure is composed of two matrices, the *Up matrix* and the *Down matrix*. The Up (Down) matrix contains, for each pair of taxa, the number of up (down) branches that need to be traversed to go from one taxon to the other. In the tree depicted in figure 8.3, two up branches need to be traversed to go from B to C and three up branches to go from A to D. Since the information from one of the matrices is enough to completely determine the tree, the Up matrix $\mathbf{U} \equiv \{u_{ab}; a, b \in \mathcal{L}\}$ will be used henceforth.

Different criteria can be used to build an optimal consensus tree from a set of input phylogenetic trees $\mathcal{T} = \{T_1, \ldots, T_K\}$. A possible objective function is the TreeRank (TR) measure (Wang et al. (2003))

$$TR(T, \mathcal{T}) = \frac{1}{K} \sum_{k=1}^{K} \left(1 - \frac{\sum_{a,b \in \mathcal{L}} |u_{ab} - u_{ab}^k|}{\sum_{a,b \in \mathcal{L}} u_{ab}} \right), \tag{8.1}$$

where $\{u_{ab}; a, b \in \mathcal{L}\}$ are the elements of the up matrix of the consensus tree and $\left\{u_{ab}^k; a, b \in \mathcal{L}\right\}$ are the elements of the up matrix of the k-th input tree. This function has the disadvantage that it is very non-linear because of the normalization factor. Therefore, it can be difficult for mathematical programming techniques to reach a solution.

The Inverse TreeRank measure avoids this complexity by using a normalization factor with respect to the input trees instead

$$TR^I(T,\mathcal{T}) = \frac{1}{K}\sum_{k=1}^{K}\left(1 - \frac{\sum_{a,b\in\mathcal{L}}|u_{ab}-u^k_{ab}|}{\sum_{a,b\in\mathcal{L}}u^k_{ab}}\right). \tag{8.2}$$

Both eq. (8.1) and (8.2) are based on the measurement of the (mean) absolute (L_1) distance between the up matrices of the obtained tree and the input trees. Removing the normalization factor, one obtains the UpDown distance

$$UD(T,\mathcal{T}) = \frac{1}{K}\sum_{k=1}^{K}\sum_{a,b\in\mathcal{L}}|u_{ab}-u^k_{ab}|, \tag{8.3}$$

which is a simpler measure that is easier to optimize.

Another measure proposed in the literature (Bryant (2003)) defines the consensus tree as the tree that maximizes the number of common rooted triplets. Let $R(T)$ be the set of rooted triplets defining tree T. The Weighted Triplet measure is

$$WT(T,\mathcal{T}) = \sum_{k=1}^{K}|R(T)\cap R(T_k)| = \sum_{t_{a,b|c}\in R(T)} w^{\mathcal{T}}_{a,b|c}t_{a,b|c}, \tag{8.4}$$

where the coefficients $w^{\mathcal{T}}_{a,b|c}$ are defined as the number of input trees in which triplet $(a,b|c)$ is present. The binary variables $t_{a,b|c}$ indicate whether triplet $(a,b|c)$ is present ($t_{a,b|c}=1$) or not ($t_{a,b|c}=0$). This measure is linear and easy to calculate. Moreover, it allows to use weights in the objective function that represent the confidence on the phylogenetic inference method used to generate the input trees.

Using one of these objective functions, the consensus tree problem can be formulated as a constrained integer optimization problem. We now introduce different ways in which the problem can be formulated.

8.2.1 The UpDown Matrix Model

The UpDown Matrix Model (UDMM) is formulated in terms of both $\{u_{ab}; a,b\in\mathcal{L}\}$, the elements in the UpDown matrix, and $t_{a,b|c}$, the variables that indicate the presence or absence of every possible triplet. The purpose of this is to allow the use of an objective function based on the values of the UpDown matrix, which is more fine-grained, and at the same time obtain the tree defined by the triplet variables to avoid the transformation from one representation to another, which can be very costly.

A number of remarks are useful in defining and implementing this model. First, to define all triplets of a tree, some elements of the UpDown matrix are unnecessary. This can be stated more formally using the following lemma:

Lemma 8.1. *Let* $\mathbf{U}$ *be an UpDown matrix. The triplet of taxa* $(ab|c)$ *belongs to the tree if and only if for the submatrix* $\mathbf{U}_{abc}$ *composed of the rows and columns of* a, b *and* c, *there exists a permutation* π *of rows and columns such that the following inequalities hold:*

$$u_{ab} < u_{ac} \tag{8.5}$$

$$u_{ba} < u_{bc} \tag{8.6}$$

$$u_{ca} = u_{cb} \tag{8.7}$$

$$u_{ac} - u_{ab} = u_{bc} - u_{ba}. \tag{8.8}$$

The proof of this lemma is included in Appendix C.

In summary, the first row of every appropriately permuted feasible 3×3 UpDown submatrix is sufficient to define the triplets that make up the tree ($u_{ab} < u_{ac}$). Nonetheless, the other rows in the matrix need to be considered as well to ensure feasibility; i.e., if triplet $ab|c$ is selected then $u_{ac} - u_{ab} = u_{bc} - u_{ba}$ and $u_{ca} = u_{cb}$.

Taking into account these observations, the consensus tree problem can be formulated as the integer linear program

$$\min_{t,u} \quad f(t,u) = \sum_{k=1}^{K} \sum_{a,b \in \mathcal{L}} |u_{ab} - u_{ab}^k| \tag{8.9}$$

$$\text{s.t.} \quad u_{aa} = 0 \qquad \forall a \in \mathcal{L} \tag{8.10}$$

$$1 \leq u_{ab} \leq n - 1 \qquad \forall a, b \in \mathcal{L} \tag{8.11}$$

$$u_{ab} < u_{ac} + M(1 - t_{a,b|c}) \qquad \forall \{a,b,c\} \subset \mathcal{L} \tag{8.12}$$

$$u_{ba} < u_{bc} + M(1 - t_{a,b|c}) \qquad \forall \{a,b,c\} \subset \mathcal{L} \tag{8.13}$$

$$u_{ca} \leq u_{cb} + M(1 - t_{a,b|c}) \qquad \forall \{a,b,c\} \subset \mathcal{L} \tag{8.14}$$

$$u_{cb} \leq u_{ca} + M(1 - t_{a,b|c}) \qquad \forall \{a,b,c\} \subset \mathcal{L} \tag{8.15}$$

$$u_{ac} - u_{ab} \leq u_{bc} - u_{ba} + M(1 - t_{a,b|c}) \qquad \forall \{a,b,c\} \subset \mathcal{L} \tag{8.16}$$

$$u_{bc} - u_{ba} \leq u_{ac} - u_{ab} + M(1 - t_{a,b|c}) \qquad \forall \{a,b,c\} \subset \mathcal{L} \tag{8.17}$$

$$\min\{u_{ab} \mid b \in \mathcal{L} \setminus \{a\}\} = 1 \qquad \forall a \in \mathcal{L} \tag{8.18}$$

$$t_{a,b|c} + t_{b,c|a} + t_{a,c|b} = 1 \qquad \forall a < b < c \in \mathcal{L} \tag{8.19}$$

$$t_{a,b|c} + t_{a,d|c} - t_{b,d|c} \leq 1 \qquad \forall \{a,b,c,d\} \subset \mathcal{L} \tag{8.20}$$

$$t_{a,b|c} + t_{a,c|d} - t_{a,b|d} \leq 1 \qquad \forall \{a,b,c,d\} \subset \mathcal{L} \tag{8.21}$$

$$t_{ab|c} = t_{ba|c} \qquad \forall a < b, c \in \mathcal{L}. \tag{8.22}$$

The meaning of the restrictions is: The distance from a taxa to itself is zero (8.10). The distance between two different taxa is at least 1 and at most the number of taxa minus 1 (8.11), since the latter is the maximum depth of a binary tree composed of n leaves. Expression (8.19) ensures that only one triplet $ab|c$, $bc|a$ or $ac|b$ is possible and expressions (8.12)-(8.17) that the UpDown matrix is consistent, where M is a large positive constant. Equations (8.16)-(8.17) are referred to as *path constraints*. They ensure that the values of the UpDown matrix, and not only the relative distances, are consistent. The *row-min constraints* (8.18) ensure that no "artificial" inner nodes can be added to lower specific taxa, which might otherwise happen, depending on the objective function used. The inequality (8.20) expresses the transitivity property, which ensures that if triplets $ab|c$ and $ad|c$ are present, then $bd|c$ must be present as well. The telescopic

condition is expressed by (8.21): if triplets $ab|c$ and $ac|d$ are part of the tree, then it must be the case that $ab|d$ is also in the tree. Finally, equality (8.22) states that triplets $ab|c$ and $ba|c$ are equivalent. As a consequence of Theorem 8.1 no additional inequalities are needed. This model involves $\Theta(n^3)$ variables and $\Theta(n^4)$ constraints. The main advantage of this model is that by using redundant variables (the elements of the u_{ab} for the UpDown matrix and the triplet variables $t_{a,b|c}$) it is possible to formulate a convenient objective function in terms of the elements of the UpDown matrix and restrictions based on both types of variables. Furthermore, it is not necessary to perform costly conversions between the two types of representations.

8.2.2 The Triplet Model

The Triplet Model (TM) is a modification of the UDMM where the variables u_{ab} are eliminated and the objective function is formulated in terms of the $t_{a,b|c}$ variables only. Using, for instance, the WT score this model can be expressed as follows

$$\max_{t} \quad f(t) = \sum_{t_{a,b|c} \in R(T)} w^{\mathcal{T}}_{a,b|c} t_{a,b|c} \tag{8.23}$$

$$\text{s.t.} \quad t_{a,b|c} + t_{b,c|a} + t_{a,c|b} = 1 \qquad \forall a < b < c \in \mathcal{L} \tag{8.24}$$

$$t_{a,b|c} + t_{a,d|c} - t_{b,d|c} \leq 1 \qquad \forall \{a,b,c,d\} \subset \mathcal{L} \tag{8.25}$$

$$t_{a,b|c} + t_{a,c|d} - t_{a,b|d} \leq 1 \qquad \forall \{a,b,c,d\} \subset \mathcal{L} \tag{8.26}$$

$$t_{ab|c} = t_{ba|c} \qquad \forall a < b, c \in \mathcal{L}. \tag{8.27}$$

8.3 Solution methods

In this section, different methods that can be used to solve the problem are detailed. The use of additional techniques that can improve the efficiency of the solution methods, such as lazy constraints, variable pruning, incumbent improving heuristics and column generation, are also discussed.

8.3.1 Preprocessing step

In practice, the phylogenetic trees generated by different methods are often similar to each other. A reasonable assumption is that common subtrees should be part of the final consensus tree. Therefore, these common subtrees can be eliminated from the input trees and, later, directly included in the final solution. The complexity of this preprocessing step depends on the number of taxa n and the number of input trees K. The subtrees of a tree can be obtained in $O(n)$ time, since the number of internal nodes is $O(n)$. The number of possible comparisons is $O(n^K)$, since all possible trees must be compared (i.e., all possible $K-$tuples must be generated). Assuming that each comparison can be done in $O(n)$ time, the total complexity is $O(n^{K+1})$.

8.3.2 Using lazy constraints

We say that a constraint is lazy if the solver has delayed considering it until the constraint is violated by the current (not necessarily integer) solution. The use of lazy constraints allows to begin with a reduced problem formulation, in which some constraints are discarded. This means that less time needs to be spent in LP relaxations of the problem because feasible solutions should be easier to obtain when some constraints are not considered. In particular, the constraints (8.20)-(8.21) are unlikely to be violated, because they just ensure feasibility of the obtained tree and the input trees already satisfy these constraints. These initially discarded constraints are then incorporated as soon as the algorithm detects that they are not satisfied by the current solution. This results in general in large improvements in efficiency in both the UDMM and TM formulations because of the large number of constraints that need to be considered ($\Theta(n^4)$).

8.3.3 Reduction of dimensionality

Triplets that do not appear in any input tree are not likely to be included in the optimal solution. Therefore, our pruning method discards triplets that are not present in any input tree. More formally, the set of variables considered is limited to $t' = \{t_{a,b|c} : (a, b|c) \in R(T_k), k = 1, \ldots, K\}$. Clearly, the possibility that some eliminated triplet actually belongs to the optimal solution cannot be discarded, and therefore the obtained solution will not be optimal in general. However, we expect to greatly improve the efficiency of the branch-and-cut algorithm with only a small deterioration of the quality of the solutions obtained.

8.3.4 Refinement of incumbents

An important part of the branch-and-bound algorithm is the search for new incumbent (integer-feasible) solutions. These solutions are obtained at those nodes of the branch-and-bound tree in which the LP relaxation is close to or already integer feasible. Obtaining good incumbents improves the bounds on the problem (if a minimization is performed, the upper bound), increasing the probability of fathoming nodes by bounds (that is, if the LP relaxation of the problem in a given node has an optimal value which is greater than the upper bound, that node and all of its descendants can be pruned from the search tree). Therefore, the efficiency of the algorithm improves if better incumbents are found, because the branch-and-bound tree tends to be smaller. A possible approach is to apply a general metaheuristic (in our case, Variable Neighborhood Search (VNS) (Hansen and Mladenovic (2003)) together with Variable Neighborhood Descent (VND) as local search heuristic) to improve the incumbents obtained, thus improving the upper bound. The question is whether the computational effort spent by the metaheuristic actually pays off with respect to the algorithm without heuristic incumbent improvement. The use of VNS seems to be appropriate for this problem because several types of tree neighborhoods can be easily defined (Andreatta and Ribeiro (2002)). In the following section, fast and incremental formulations of algorithms to update the UpDown matrix are given (Pirkwieser and Ruiz-Torrubiano (2007)).

8.3.4.1 SWAP

This operator simply swaps two taxa. The UpDown matrix can be modified efficiently by just interchanging the rows and columns i and j. The computational cost of this operation is linear $\Theta(n)$. Since the topology of the tree remains the same, not every tree in the search space can be reached starting from a given tree. The number of possible SWAP moves is $n(n-1)/2$.

8.3.4.2 STEP

A STEP move (Algorithm 18) removes a taxon and inserts it at a random position. This modification is the smallest possible change in the topology of the tree.

The worst-case complexity of this algorithm is $O(n^2)$. The algorithm has two separate phases: *deletion* and *insertion*. In the deletion phase, all the distances are updated so that the node which is the nearest ancestor of a is deleted. In the insertion phase, two cases must be considered. First, if a is added above b, then the nearest ancestor of a is also the nearest ancestor of the nearest ancestor of b. If a is added beneath b, then a and b have the same ancestor. All distances must be updated accordingly. The number of possible STEP moves is $2n(n-1)$.

8.3.4.3 NNI or ROTATE

Nearest neighbor interchange (NNI) or rotations within the tree (ROTATE) define the same neighborhood. A ROTATE move, as the name suggests, rotates either the left or right subtree to the left or to the right. Therefore, there are four possible moves (in list preorder notation):

- $R_R^1[(h,(h',T_{LL},T_{LR}),T_R)] = (h,T_{LL},(h',T_{LR},T_R))$.
- $R_R^2[(h,(h',T_{LL},T_{LR}),T_R)] = (h,T_{LR},(h',T_{LL},T_R))$.
- $R_L^1[(h,T_L,(h',T_{RL},T_{RR}))] = (h,(h',T_L,T_{RL}),T_{RR})$.
- $R_L^2[(h,T_L,(h',T_{RL},T_{RR}))] = (h,(h',T_L,T_{RR}),T_{RL})$.

The incremental change of the UpDown matrix is given in Algorithm 19 for the R_R^1 case, but can be performed in an analogous way for the other moves. The number of possible ROTATE moves is $2n-4$.

8.3.4.4 SPR

A subtree prune and regraft (SPR) move deletes an internal branch of the tree and regrafts it elsewhere in the remaining tree. We proceed by selecting the smallest subtree containing a given taxon A. Then we select another taxon B, which is not included in this subtree, and choose whether to add this subtree beneath or above B. An incremental algorithm for updating the UpDown matrix is given in Algorithm 20.

Algorithm 18 An efficient version of the STEP operator.

- Choose randomly taxon $a \in \mathcal{L}$ to be removed.
- Choose randomly taxon $b \neq a \in \mathcal{L}$ as the new nearest neighbor of a.
- **(Deletion phase)** Let $T_a = \{c : u_{ac} = 1\}$ be the minimum subtree containing a. Let $T'_a = \mathcal{L} - T_a - \{a\}$.
- For each $c \in T_a$, $d \in T'_a$, $u_{cd} := u_{cd} - 1$.
- Choose randomly to add a **above** or **beneath** b.
- **(Insertion phase)** If a must be added **above** b:
  - $u_{ab} = 1$; $u_{ba} = 2$.
  - Let $T_b = \{c : u_{bc} = 1\}$, and $T'_b = \mathcal{L} - T'_b - \{a, b\}$.
  - For each $c \in T_b$,
    - $u_{ac} := u_{bc}$.
    - $u_{ca} := u_{cb} + 1$.
  - For each $c \in T_b$, $d \in T'_b$,
    - $u_{cd} := u_{cd} + 1$.
  - For each $c \in T'_b$,
    - $u_{bc} := u_{bc} + 1$.
    - $u_{ac} := u_{bc} - 1$.
    - $u_{ca} := u_{cb}$.
- If a must be added **beneath** b:
  - $u_{ab} = 1$; $u_{ba} = 1$.
  - Let $T_b = \{c : u_{bc} = 1\}$, and $T'_b = \mathcal{L} - T'_b - \{a, b\}$.
  - For each $c \in T_b$,
    - $u_{bc} := u_{bc} + 1$.
    - $u_{ac} := u_{bc}$.
    - $u_{ca} := u_{cb}$.
  - For each $c \in T'_b$,
    - $u_{bc} := u_{bc} + 1$.
    - $u_{ac} := u_{bc}$.
    - $u_{ca} := u_{cb}$.

Algorithm 19 The ROTATE operator.

- remove inner node I':
  - lift taxa in T_{LL}:

$$\forall a \in T_{LL}, \forall b \in \mathcal{L} \setminus \{T_{LL} \cup T_{LR}\} : \; u_{ab} := u_{ab} - 1$$

- insert new inner node I''
  - lower taxa in T_{LR} relative to T_{LL}:

$$\forall a \in T_{LR}, \forall b \in T_{LL} : \; u_{ab} := u_{ab} + 1$$

  - lift taxa in T_{LR} relative to T_R:

$$\forall a \in T_{LR}, \forall b \in T_R : \; u_{ab} := u_{ab} - 1$$

  - lower taxa in T_R relative to all others but T_{LR}:

$$\forall a \in T_R, \forall b \in \mathcal{L} \setminus \{T_R \cup T_{LR}\} : \; u_{ab} := u_{ab} + 1$$

Algorithm 20 The SPR algorithm.

- Choose randomly a taxon a which defines the subtree that will be pruned and regrafted as $T_a = \{c : u_{ac} = 1\}$.
- Choose randomly taxon b as the new nearest neighbor of a outside T_a. If $b \in T_a$, repeat until $\{b\} \cap T_a = \emptyset$.
- **(Deletion phase)** Let $T' = T - T_a$.
- For each $c \in T_a$, $d \in T'$, $u_{cd} := u_{cd} - 1$.
- Choose randomly to add T_a **above** or **beneath** b.
- Let $T_b = \{c : u_{bc} = 1\}$, and let $T'' = T - T_b - \{a\}$.
- **(Insertion phase)** If T_a must be added **above** b:
  - $u_{ab} := 1$, $u_{ba} := 2$.
  - For each $c \in T_b$, $c \neq a$, $u_{ac} := 1$, $u_{ca} := u_{cb} + 1$.
  - For each $c \in T_b$, $c \neq a$, $d \in T''$, $u_{cd} := u_{cd} + 1$.
  - For each $d \in T''$, $u_{ad} := u_{bd}$, $u_{bd} := u_{bd} + 1$, $u_{da} := u_{db}$.
- Else if T_a must be added **beneath** b:
  - $u_{ab} := 1$, $u_{ba} := 1$.
  - For each $c \in T_b$, $c \neq a$, $u_{bc} := u_{bc} + 1$, $u_{ac} := u_{bc}$, $u_{ca} := u_{cb}$.
  - For each $d \in T''$, $u_{bd} := u_{bd} + 1$, $u_{ad} := u_{bd}$, $u_{da} := u_{db}$.

In this algorithm, we differentiate between a *deletion* and an *insertion* phase. The deletion phase simply updates the distances so that T_a is removed from the tree, eliminating the inner node that is the least ancestor of the root of T_a. Then the algorithm proceeds with the insertion phase: If the insertion is to be performed above b, then the distances between a and b are updated accordingly. The distance from a to every element of T_b is 1, because T_a is inserted above b. The distance from every element of T_b to a is now 1 more than the distance to b. All the remaining distances are updated accordingly, taking into account that a new inner node has been introduced (the inner node which is the least ancestor of the root of T_a). If T_a must be inserted beneath b, then the distances between a and b are updated similarly. Now the distances from b to the elements of T_b is 1 more, since a new inner node has been introduced, so that T_a can be inserted. The rest of the distances are now updated taking this new inner node into account. The worst-case complexity of this algorithm is $O(n^2)$.

The number of possible unrestricted SPR moves is in the order of $2n(n-1)$. Note that by implementing the algorithm this way, not every possible subtree can be pruned, since we are restricting ourselves to the minimum subtrees containing a given taxa. Nonetheless, the most general moves can be achieved by several applications of this algorithm.

8.3.5 Generating new variables

The use of exact column generation techniques is an efficient approach to solve large scale integer linear problems (Lübbecke and Desrosiers (2005)). The principal idea is to begin with a reduced number of variables and add new ones as soon as they are needed in the optimization problem. Since in the ILP formulation of the consensus tree problem a large number of variables are needed, a column generation approach seems to be an appropriate technique to solve the problem. Consider an ILP problem of the form

$$\min_{\lambda} \quad \sum_{j \in J} c_j \lambda_j \tag{8.28}$$

$$\text{s.t.} \quad \sum_{j \in J} \mathbf{a}_j \lambda_j \geq \mathbf{b} \tag{8.29}$$

$$\lambda_j \geq 0 \qquad j \in J, \tag{8.30}$$

in which the set J represents a meaningful set of the original problem variables (or *columns*). This problem is referred to as the *master problem*. We begin therefore with this reduced problem and add new variables as soon as they are needed. It is possible to determine the optimal column that should be included in the problem formulation by solving the *pricing problem*: Consider the dual formulation of the master problem (8.28)-(8.30). Let $\mathbf{u} \geq \mathbf{0}$ be the vector of dual variables. Adding the variable with the *minimum negative reduced cost* $\bar{c}_j = c_j - \mathbf{u}^T \mathbf{a}_j \leq 0$ improves the objective value the most. The pricing problem can be stated as follows

$$\min \quad \mathbf{c}(\mathbf{a}) - \bar{\mathbf{u}}^T \mathbf{a} \tag{8.31}$$

$$\text{s.t.} \quad \mathbf{a} \in \mathcal{A}. \tag{8.32}$$

If the optimal solution value is non-negative, the solution to the master problem is optimal. Since for the consensus tree problem it is natural to begin with the reduced set of triplets t', and add new variables as needed, a column generation (or *branch-and-price*) algorithm could be appropriate. Unfortunately, in some cases the pricing problem is difficult to solve and the benefits obtained by solving it do not compensate the computational effort. To reduce the cost of solving the associated pricing problem, we introduce a heuristic to identify the triplets that should be added to improve the objective value. This algorithm is a greedy rounding heuristic (see Algorithm 21), which is executed at selected nodes of the branch-and-bound search tree. The heuristic produces a feasible solution from the current LP relaxation.

Algorithm 21 Greedy rounding heuristic

For each set of triplets $\{t_{a,b|c}, t_{b,c|a}, t_{a,c|b}\}$:

- Select the triplet t^* with the maximum relaxed value $\hat{t}_{a,b|c}$.
- Set $t^* = 1$, and the other two triplets to 0.

If the obtained solution is not feasible, then repair constraints:

- If $t_{T_1} + t_{T_2} - t_{T_3} > 1$ then:
  - Set triplet $t_{T_3} = 1$.
  - Set complementary triplets to 0.

Until a feasible tree is obtained.

Once this feasible solution is obtained, we compare it to the current incumbent. If it is better, the triplets that are included in this solution but are not present in the master formulation are added, and the current incumbent is updated. Note that in this problem reduced costs of single variables do not make much sense: other variables can also be needed, and therefore the reduced cost of a subset of variables and not only of a single variable should be taken into account (that is, the reduced cost of the set of triplets which are needed in order to obtain a feasible tree). For instance, see constraint (8.24). If triplet $t_{a,b|c}$ is added, triplets $t_{b,c|a}$ and $t_{a,c|b}$ are needed as well. We refer to this procedure as *heuristic column generation*.

8.4 Results

In this section we present an empirical evaluation of the methods and auxiliary techniques described in the previous section. The formulation of the consensus tree problem

that is considered is (8.23)-(8.26). If a similarity measure based on the values of the Up-Down matrix is used, the complete model (8.9)-(8.22) should be used instead.

The comparisons carried out involve the following techniques

1. Using lazy constraints.
2. Using input triplets only.
3. VND/VNS incumbent improving heuristic.
4. Heuristic column generation.

The ILP solver CPLEX was used for the first, second and third approaches. The open-source application COIN-BCP[1] was used for column generation, because this approach is not implemented in CPLEX. The following measures are used for evaluation of the different methods

- Best solution found.
- Total execution time.
- Total execution time without solver initialization.
- Total lazy constraints (when possible).
- Used lazy constraints (when possible).
- Amount of pruning (when possible).
- Amount of added variables (when possible).

The methods are evaluated using both synthetic (`instancex_r`) and actual phylogenetic data (`mammals20` and `mammals34`). The size of the problems considered range from 9 to 34 taxa. The input trees from the synthetic problems generally are rather dissimilar. By contrast, in real instances the input trees present more similarities.

Table 8.1 shows that the performance of the solver that uses lazy constraints is generally efficient. The largest problem that uses actual genetic data (34 taxa) is solved in less than eight seconds. The largest synthetic problem (25 taxa) is solved in less than nine seconds. Nonetheless, problems with larger number of taxa are not necessarily more difficult to solve. For instance, more than fifty seconds are required to solve `instance7_r`, which includes 20 taxa. There are several possible explanations for this behavior. First, it takes longer (88 nodes must be visited) to find the first integer feasible solution. By contrast in the other problems with 20 taxa, the first feasible solution is already found in the root node. Second, 4182 lazy constraints had to be considered in the initial formulation to find the optimal solution. In the other problems with 20 taxa it was sufficient to include around 2000 lazy constraints.

[1] https://projects.coin-or.org/Bcp

Instance	Taxa	CPLEX + lazy constraints						
		Best	Best (TR)	Best (TR) In	Time	Time (w.o. init)	Total lazy	Used lazy
`instance1_r`	9	141	57.51	66.32	0.06	0.04	4536	202
`instance2_r`	12	367	44.30	57.57	5.47	5.4	17820	1374
`instance3_r`	12	392	51.38	60.88	0.17	0.1	17820	288
`instance4_r`	15	819	48.75	56.73	12.25	12.06	49140	2994
`instance5_r`	15	1014	68.73	75.90	0.23	0.04	49140	1073
`instance6_r`	15	1088	72.94	74.29	0.25	0.05	49140	433
`instance7_r`	20	2113	52.94	69.84	53.81	53.11	174420	4182
`instance8_r`	20	2397	60.67	69.75	1.62	0.91	174420	1993
`instance9_r`	20	2275	54.89	69.34	1.36	0.69	174420	2080
`instance10_r`	25	4719	62.56	68.54	8.89	7.13	455400	2659
`mammals20_r`	20	3186*	91.60	91.60	0.79	0.1	174420	0
`mammals34_r`	34	16552	86.20	86.72	7.97	1.24	1669536	0

Table 8.1: Results obtained with lazy constraints

A similar phenomenon can be observed in the problems with 15 taxa. Problem `instance4_r` requires much more time than the other two problems with 15 taxa. It is necessary to explore 48 nodes (about 3000 lazy constraints) to find the first incumbent solution. In the other two cases `instance5_r` and `instance6_r` (about 1000 and 500 lazy constraints applied, respectively) a feasible solution is found in the root node. It is remarkable that in problems with actual genetic data, no lazy constraints were necessary. Since the input trees are quite similar, the LP relaxations directly yield a solution satisfying the lazy constraints.

The column "Best (TR)" stands for the TreeRank value of the optimal solution with respect to the WT score. The column labeled "Best (TR) In" shows the TreeRank measure of the best input tree. The WT score and the TreeRank measure are largely uncorrelated: the best WT tree in most cases is worse than the best input tree with respect to TreeRank, except in the case of the `mammals20` instance where the optimal tree with respect to both measures is one of the input trees.

Table 8.2 shows the results obtained after pruning the triplets that do not appear in any input tree. The values obtained are therefore not necessarily optimal, because the discarded triplets could appear in the optimal solution. Nonetheless, the solutions obtained seem to be near-optimal. Note that no lazy constraints are used in the experiments shown in this table. Unfortunately, only in five out of the twelve problem instances the total execution time is improved. A possible explanation is that the pruned problems are more difficult for CPLEX, possibly because the constraints are not sufficiently tight. However, in the instances with actual genetic data the optimal solution is found with the reduced set of triplets. This is also the case in two synthetic instances (`instance3_r` and `instance4_r`). The column labeled "Pruning" shows the percentage of the remaining variables after pruning. More pruning takes place in the instances with actual data than in the random instances. This is mainly due to the larger similarity among the input trees.

The results obtained when the VNS heuristic is used to improve the incumbent solutions are shown in Table 8.3. It is clear from the table that using this procedure does not pay off in terms of execution time. The heuristic does find new incumbents

Instance	Taxa	CPLEX pruning			
		Best	Pruning (%)	Time	Time (w.o. init)
instance1_r	9	140	71.43	0.68	0.67
instance2_r	12	366	71.97	7.95	7.93
instance3_r	12	392	69.70	0.33	0.31
instance4_r	15	819	69.52	24.12	24.06
instance5_r	15	1003	57.95	0.21	0.17
instance6_r	15	1086	52.31	0.20	0.16
instance7_r	20	2071	66.49	207.57	207.4
instance8_r	20	2390	60.70	2.03	1.89
instance9_r	20	2265	63.95	2.44	2.28
instance10_r	25	4712	61.77	6.21	5.85
mammals20_r	20	3186	40.18	0.15	0.08
mammals34_r	34	16552	40.91	2.61	1.88

Table 8.2: Results obtained with pruning

Instance	Taxa	CPLEX + VND/VNS		
		Best	Time	Time (w.o. init)
instance1_r	9	141	0.41	0.39
instance2_r	12	367	3.89	3.85
instance3_r	12	392	1.69	1.65
instance4_r	15	819	206.72	206.61
instance5_r	15	1014	0.6	0.48
instance6_r	15	1088	0.64	0.52
instance7_r	20	2113	5982.76	5982.38
instance8_r	20	2397	6.16	5.79
instance9_r	20	2275	3.41	3.04
instance10_r	25	4719	40.71	39.79
mammals20_r	20	3186	2.23	1.87
mammals34_r	34	16552	22.93	22.56

Table 8.3: Results obtained with VNS incumbent improving

earlier (for example, the first incumbent in `instance7_r` is found at the root node, while without the heuristic is found at node 88). However, the application heuristic dominates in the computational cost of the branch-and-bound algorithm.

Table 8.4 shows the results using the heuristic column generation approach with BCP. The times obtained by CPLEX are better by several orders of magnitude. Since it would not be fair to compare against CPLEX, an additional column labeled Time BCP shows the total time required without column generation by the standard BCP solver. Most of the execution time is spent in this case in the initialization phase. In general, using this approach entails an improvement in efficiency when compared to normal BCP. The quality of the solution also improves when compared to the quality of the solutions obtained by pruning alone. However, the number of added variables is relatively low. This has two possible explanations: Either few variables need to be added to find a solution of reasonably good quality, or the greedy rounding heuristic is not effective in the identification of the variables that need to be added to find the optimal solution. Given the results obtained the second explanation seems to be more likely. Therefore, more effort should be devoted to investigating new variable generation heuristics that yield better results.

Instance	Taxa	BCP + Heur. col. gen.					
		Best	Pruning	Added vars	Time	Time (w.o. init)	Time BCP
instance1_r	9	140	71.43	4	1.24	0.32	1.04
instance2_r	12	367	71.97	5	29.00	13.74	119.12
instance3_r	12	392	69.70	11	24.07	8.87	20.15
instance4_r	15	819	69.52	10	131.14	10.48	238.04
instance5_r	15	1003	57.95	2	117.12	1.33	121.64
instance6_r	15	1086	52.31	18	116.48	2.35	121.40
instance7_r	20	2079	66.49	31	20853.27	19270.70	2725.15
instance8_r	20	2390	60.70	0	1515.83	7.7	1533.81
instance9_r	20	2267	63.95	1	1611.59	20.66	1521.89
instance10_r	25	4712	61.77	0	10331.13	15.92	10409.24
mammals20_r	20	3186	40.18	0	1423.38	0.96	1528.80
mammals34_r	34	16552	40.91	0	130323.55	11.27	139226.51

Table 8.4: Results obtained with heuristic column generation

8.5 Summary and Conclusions

In this chapter, several methods for solving the consensus tree problem have been proposed and analyzed. We have first introduced a new formulation in terms of an ILP optimization that involves both the variables corresponding to the entries of the Up-Down matrix and the triplet variables. The use of a redundant representation means that it is not necessary to perform costly conversion procedures between both representations. The objective function can be formulated in terms of UpDown distances. The constraints involve both types of variables. Alternatively an objective function that uses triplet variables only can be used to address the problem. Second, we have proposed various methods to improve the performance of branch-and-bound approaches to solve the problem. These methods include the use of lazy constraints, pruning of the input variables, metaheuristic incumbent solution improving and heuristic column generation.

In general, the use of lazy constraints is the most successful approach among the techniques considered. Discarding triplets that do not appear in any input tree does not significantly deteriorate the quality of the solution. Therefore, these triplets are not likely to appear in the optimal solution. Finally, the use of the incumbent improving heuristic is not effective, at least when used together with CPLEX. The high computational cost of the heuristic dominates the computational cost of the solver. The main difficulty in the search for new incumbents is to improve the current upper bound which is the objective value of the best incumbent solution found so far. If these incumbent solutions are good enough, more nodes can be fathomed by bounds and the search tree becomes more tractable. Finally, the use of heuristic variable generation is clearly beneficial in this problem both in terms of the quality of the final solutions and the improvements in efficiency obtained.

As future work, new and efficient heuristics are needed to identify the variables that should be included in the problem. Additionally, a more exhaustive evaluation is needed to determine if using a heuristic column generation approach, instead of the exact one, is advantageous. More efficient metaheuristics could be used to improve the incumbent solutions. A possibility would be SA with one of the neighborhood operators described

in Section 8.3. This simpler metaheuristic should introduce less overhead than VNS for this problem.

CHAPTER 9

CONCLUSIONS AND FUTURE WORK

In this thesis we have investigated the design of hybrid methods that can be used to efficiently find near-optimal solutions of problems with cardinality constraints. Metaheuristics have been used to address numerous optimization problems of practical interest. However, the standard formulations of these metaheuristics have difficulties when additional restrictions are considered, especially in domains with both continuous and discrete aspects. Moreover, as expressed by no-free lunch theorems in optimization, general metaheuristic approaches cannot perform better than random search when averaged over all problem instances. The hybrid optimization methods proposed in this thesis have been designed taking into account these issues. By separating the problem into a discrete and a subordinate continuous optimization task, the metaheuristic can be specifically designed to address the combinatorial part of the search efficiently. In this manner, the choice of representation for the candidate solutions and of successor operators is not complicated by the entanglement of different aspects of the problem. In particular, the design of the metaheuristic does not need to take into account the continuous constraints and can focus on the discrete ones. One of the important characteristic shared by the most successful metaheuristics designed is that only feasible candidate solutions are generated in the search. The candidate solutions proposed by the metaheuristic are evaluated in terms of the solution of a subordinate optimization problem. In the cases investigated this subordinate problem can be efficiently solved using standard continuous optimization techniques. Specific knowledge of the structure of the problem can be incorporated in the form of a meaningful genetic representation. In particular, using the concepts of forma theory, we have defined a genetic set representation that makes explicit the basis of equivalence relations that appropriately reflects the structure of optimization problems with cardinality constraints. In the experimental evaluation performed, genetic algorithms that use this encoding generally outperform standard genetic algorithms that employ a binary encoding.

In the second part of this thesis, we have carried out a comparison of the solutions obtained using genetic algorithms, simulated annealing and EDAs in different application domains: the knapsack problem, sparse PCA, index tracking and optimal portfolio

selection. In most of the problems investigated, simulated annealing generally obtains slightly worse solutions than the genetic algorithm, despite the fact that the same type of set encoding is used in both cases. The neighborhood operator in simulated annealing is the same as the mutation operator in the set GA. From these observations, one concludes that the crossover operator plays an important role in improving the efficacy of the optimization process. Estimation of distribution algorithms rely on the accurate estimation of the probability distribution of the genes from the current population of individuals. In high-dimensional problems the estimation and sampling from the probability distribution becomes increasingly difficult. The experiments performed confirm this analysis: algorithms of the EDA family do not converge to good solutions in the problems investigated with a large number of variables. In summary, the best overall results both in terms of quality of the solutions and of computational efficiency in the problems investigated are obtained by the GA with set encoding. The use of specially designed mutation and genetic operators that work on sets that encode the candidate solutions has been found to be the most appropriate approach to solving problems with cardinality constraints. The results obtained also highlight the importance of the crossover operator in improving the efficiency of the search.

Besides standard cardinality-preserving crossover operators that have been introduced in previous work by other authors, such as RRR or RAR, in this thesis we have introduced the TransRAR crossover operator. The design of the TransRAR operator incorporates a number of desirable properties identified by the theory of formae in a novel way. Specifically, it provides a good balance between the properties of assortment, which tends to enhance exploration, and transmission, that is related to exploitation in the search. In the problems to which it has been applied, the use of the TransRAR crossover in the set GA consistently outperforms the GA with the RAR operator. TransRAR focuses on enforcing transmission (the offspring is constructed using alleles present in at least one of the parents) instead of respect (the common information in the parents is always present in the offspring, but information not present in any parent may be used as well), which is one of the properties favored by RAR. Transmission seems to be preferable to respect because it enhances the exploitation of useful information identified in the search.

Another important contribution of this thesis is to analyze the effectiveness of dimensionality reduction techniques when used in combination with metaheuristics to address large-scale problems with cardinality constraints. In standard optimization problems, discarding variables will in general lead to a deterioration of the solutions. This need not be the case in cardinality-constrained problems. Because of the cardinality constraints some of the original variables do not appear in the optimal solution. If those variables are eliminated from the problem the quality of the solution will not be affected. The dimensionality reduction is implemented in a preprocessing step using pruning heuristics. Pruning heuristics have been designed that attempt to identify and eliminate variables that are not likely to be included in the optimal solution. Three different heuristics are analyzed. They are based on solving relaxed versions of the problem and performing the selection according to the results of these auxiliary optimizations. This procedure

is practicable only if the relaxed optimizations can be efficiently solved. In the experiments performed we show that the dimensionality of the problem can be significantly reduced with only a minor deterioration of the quality of the final solutions. Pruning can therefore be used to reduce the computational cost of the optimization in all the metaheuristics investigated, and specially for EDAs.

Cardinality constraints are useful because they allow the identification of sparse solutions, which are more stable to the inputs of the optimization. This property is illustrated in the portfolio selection problem with piecewise linear transaction costs. Standard portfolios selected using the classical Markowitz framework, which are optimal in-sample, often exhibit poor out-of-sample performance. The inclusion of transaction costs and cardinality constraints in the optimization model has a regularization effect that allows the selection of portfolios that are more robust, stable and, in general, have better out-of-sample performance. A lasso-based strategy, which penalizes large fluctuations in the composition of the portfolio, and the strategy with cardinality constraints were found to obtain the best overall results in the experiments performed. Combining cardinality constraints with a lasso-type penalty results in too much regularization in the problems analyzed and the composition of the portfolios obtained are too static and do not achieve good out-of-sample performance.

In the consensus tree problem, the performance of a branch-and-cut based approach to solve the problem can be improved by eliminating variables (triplets) that do not appear in any input tree. The use of lazy constraints and the heuristic generation of new variables as soon as they are needed in the search process were identified as effective procedures to obtain high quality consensus trees.

9.1 Future Work

There are several directions in which the investigations carried out in this thesis can be extended. Specifically, the strategies that have used to address optimization problems with mixed discrete and continuous aspects need to be extended and improved. The design principles used in this thesis can also be used to address problems in other areas of application of current interest. The following is a non-exhaustive list of future work along these lines:

- **Genetic representations and operators**: Forma theory was shown to be a useful framework for the design of meaningful genetic representations and operators. However, more work needs to be done to improve the understanding of desirable properties of crossover operators. Specifically, the role of the parameter p in the TransRAR crossover operator should be investigated in detail so that it can be selected in a more principled manner.

- **New heuristics for dimensionality reduction:** An important task is the design of accurate pruning heuristics to identify the relevant variables in cardinality-constrained problems. A possible approach is to use Bayesian networks to model dependencies among the variables. Bayesian networks can be used to incorporate

available domain knowledge, or, if the structure of the network is learned, to infer which are the relevant variables.

- **Applications in which the subordinate optimization problem is complex:** In this thesis, we have focused on the particular case that the subordinate problem can be efficiently solved by standard optimization techniques. If the subordinate problem is more complex (for instance, if the Sharpe ratio is used as the objective function in the portfolio selection problem) one can use a non-linear solver or design a metaheuristic approach to handle the subproblem. An alternative would be the use of evolution strategies in case of a continuous problem. Since the subproblems tend to have a much lower dimensionality than the master cardinality constrained problem, continuous EDAs may be a practicable approach in this case.

- **Application of the framework developed for handling cardinality constraints to other mixed-integer problems:** The advantage of the hybrid framework proposed in this thesis is that it disentangles the combinatorial and the continuous aspects of the optimization. This allows to combine algorithms that are well-suited to solve each of the parts of the problem separately, avoiding the complications that arise when they are addressed jointly. For instance, multi-stage scheduling is a problem in which a similar separation is possible (Harjunkoski and Grossmann (2002)). The problem can be divided into an assignment and a sequencing problem. The assignment problem is discrete and involves assigning jobs to machines. The sequencing problem is continuous. It consists in determining the starting times for each job.

- **Applications in information retrieval:** The ideas used in portfolio selection have recently been applied in the field of information retrieval in Wang and Zhu (2009). The problem consists in retrieving the most relevant documents given a user query. Since there are uncertainties associated with the query, a probabilistic framework is proposed in which the relevance of the documents retrieved is maximized, while minimizing the uncertainty (variance) of this solution. It would be of interest to apply the ideas and hybrid methods developed in this thesis to a cardinality constrained version of the information retrieval problem.

- **Stability of sparse solutions**: More work needs to be done to investigate the relationship between sparsity and desirable statistical properties such as robustness against small changes in the inputs of the optimization and good out-of-sample performance. To this end, other areas of application, especially in the area of machine learning, should be explored as well.

CHAPTER 10

CONCLUSIONES

En esta tesis hemos investigado el diseño de métodos metaheurísticos que pueden ser utilizados para encontrar soluciones cuasi-óptimas a problemas de optimización con restricciones de cardinalidad de manera eficiente. Estos métodos han encontrado un amplio abanico de aplicaciones en problemas de importancia práctica. Sin embargo, las formulaciones generales de estas metaheurísticas encuentran dificultades al ser confrontadas con restricciones adicionales, especialmente en dominios continuos. Adicionalmente, como expresado por los teoremas *no-free-lunch* en optimización, técnicas metaheurísticas generales no pueden funcionar mejor que búsqueda aleatoria al promediar sobre todas las instancias posibles. Las técnicas híbridas de optimización propuestas en esta tesis han sido diseñadas para resolver estos problemas. Por medio de la separación del problema en una parte discreta y una parte continua, la metaheurística puede ser diseñada para tratar la parte combinatoria del problema de manera eficiente. De este modo, la elección de la representación de los candidatos a solución y de los operadores de sucesión no se complica por la interacción de los diferentes aspectos del problema. En particular, el diseño de la metaheurística no necesita tener en cuenta las restricciones continuas y puede concentrarse en las discretas. Una de las características importantes que comparten las metaheurísticas con mejores resultados en los problemas investigados es que únicamente candidatos a solución que cumplen las restricciones son generados en la búsqueda. Las soluciones candidatas propuestas por la metaheurística son evaluadas en términos de la solución de un problema subordinado de optimización. En los casos investigados, este problema subordinado puede ser resuelto de manera eficiente utilizando técnicas estándar de optimización continua. Conocimiento específico del problema puede ser incorporado en forma de una representación genética apropiada. En particular, utilizando los conceptos de la teoría de formas, hemos definido una representación genética de conjuntos que hace explícita la base de relaciones de equivalencia que refleja de manera apropiada la estructura de problemas de optimización con restricciones de cardinalidad. En la evaluación experimental propuesta, los algoritmos genéticos que utilizan esta codificación tienen en general mejor rendimiento que algoritmos genéticos que emplean una codificación binaria para este tipo de problemas.

En la segunda parte de esta tesis hemos realizado asimismo una comparación de las soluciones obtenidas utilizando algoritmos genéticos, temple simulado y algoritmos de estimación de distribuciones en varios dominios de aplicación: el problema de la mochila, análisis de componentes principales dispersas, replicación de índices y selección óptima de carteras. En la mayoría de los problemas investigados, el algoritmo de temple simulado no es capaz de alcanzar en general la misma calidad en las soluciones que el algoritmo genético, aunque la codificación utilizada es idéntica. El operador de vecindad en el temple simulado es igual que el operador de mutación en el algoritmo genético. A partir de estas observaciones, se puede concluir que el operador de cruzamiento asume un papel importante en la eficacia de la optimización. Los algoritmos de estimación de distribuciones dependen de una estimación lo más exacta posible de la distribución de probabilidad de los genes en la población actual de individuos. Sin embargo, incrementar la dimensionalidad del problema hace la estimación y el muestreo de la distribución de probabilidad cada vez más difícil. Los experimentos realizados confirman este análisis: los algoritmos de esta familia no son capaces de obtener buenos resultados en los problemas investigados con un gran número de variables. En resumen, el algoritmo genético con codificación de conjuntos alcanza los mejores resultados en términos tanto de calidad de las soluciones como de eficiencia computacional en los problemas investigados. El uso de operadores de mutación y de cruzamiento que operan en conjuntos que codifican los candidatos a soluciones ha sido determinado como la manera más apropiada de resolver problemas con restricciones de cardinalidad entre las alternativas investigadas. Los resultados obtenidos también resaltan la importancia del operador de cruzamiento para mejorar la eficiencia de la búsqueda.

Además de operadores de cruzamiento estándar que ya habían sido propuestos en trabajos previos de otros autores, como RRR y RAR, en esta tesis hemos propuesto el operador de cruzamiento TransRAR. El diseño de este operador incorpora propiedades deseables identificadas por la teoría de formas de manera novedosa. Específicamente, este operador presenta un buen equilibrio entre las propiedades de variedad, que es responsable de la capacidad exploratoria de la búsqueda, y transmisión, responsable de la parte de explotación. En los problemas en los que ha sido aplicado, el uso del operador TransRAR en el algoritmo genético mejora consistentemente los resultados del algoritmo con RAR. TransRAR otorga más importancia al concepto de transmisión (los descendientes son construidos con alelos procedentes de algún padre) que respeto (la información común de ambos padres es utilizada en la descendencia, pero información que no está presente en ninguno puede ser utilizada a su vez), que es la propiedad favorecida por RAR. Transmisión parece ser más preferible que respeto porque aumenta la capacidad de explotación de la búsqueda.

Otra contribución importante de esta tesis es el análisis de la efectividad de técnicas de reducción de la dimensionalidad usadas conjuntamente con metaheurísticas para problemas con restricciones de cardinalidad con un elevado número de variables. En problemas estándar de optimización, eliminar variables provoca en general un deterioro de la calidad de las soluciones obtenidas. Sin embargo, esto no es necesariamente cierto en problemas con restricciones de cardinalidad. Debido a las restricciones de cardinalidad, algunas de las variables originales no aparecen en la solución óptima. Si estas

variables se eliminan del problema, la calidad de la solución permanecerá inalterada. La reducción de dimensionalidad se implementa en un paso de preprocesamiento utilizando una heurística de poda. Esta heurística identifica y selecciona las variables del problema que tienen poca probabilidad de tener un valor distinto de cero en la solución óptima. Tres heurísticas de poda han sido propuestas, las cuales se basan en resolver relajaciones del problema y realizar la selección de variables de acuerdo con los resultados de estas optimizaciones auxiliares. Este procedimiento sólo es factible si las optimizaciones relajadas se pueden resolver de manera eficiente. En los experimentos propuestos se muestra que la dimensionalidad del problema puede ser reducida de forma significativa sin afectar considerablemente la calidad de las soluciones finales. Por lo tanto, los métodos de poda pueden ser utilizados para reducir el coste computacional de la optimización para todas las metaheurísticas investigadas, especialmente para algoritmos de estimación de distribuciones.

Las restricciones de cardinalidad son útiles porque permiten la identificación de soluciones dispersas, las cuales son más estables respecto a los parámetros de entrada de la optimización. Este efecto se ilustra en el problema de selección de carteras con costes de transacción. Carteras estándar seleccionadas con el modelo clásico de Markowitz, que son óptimas dentro de muestra, exhiben un rendimiento muy bajo fuera de muestra. La introducción de costes de transacción en el modelo de optimización y la inclusión de restricciones de cardinalidad inducen un efecto de regularización que permite la selección de carteras que son más robustas, estables y tienen mejor rendimiento fuera de muestra. Una estrategia basada en el *lasso* que penaliza fluctuaciones grandes en la composición de la cartera y la estrategia con restricciones de cardinalidad obtuvieron los mejores resultados en los experimentos realizados. Combinar restricciones de cardinalidad y penalizaciones de tipo *lasso* resulta en los problemas analizados en una regularización excesiva y las soluciones obtenidas son demasiado estáticas para alcanzar buenos resultados fuera de muestra.

En el problema de árboles filogenéticos de consenso se mostró que el rendimiento de una técnica basada en *branch-and-cut* para resolver el problema puede ser mejorado eliminando variables (tripletes) que no aparecen en los árboles de entrada. Adicionalmente, generar nuevas restricciones y variables a lo largo del proceso de búsqueda se identificaron como técnicas efectivas para obtener árboles de consenso de gran calidad.

APPENDIX A

APPENDIX FOR CHAPTER 3

A.1 Proof of Lemma 3.11

Lemma 3.11 provides a connection between the fundamental concepts of dynastic potential and similarity set. A proof of this lemma follows:

Lemma. Let $P \subseteq \mathcal{S}$ be a given set of candidate solutions. Then their dynastic potential is contained in their similarity set $\Gamma(P) \subseteq \Sigma(P)$.

Proof. Let $z \in \Gamma(P)$. By definition, $\forall \xi \ni z$, $\exists x \in P$ such that $x \in \xi$. We now distinguish two cases:

- If $\exists\{\xi_1^*, \ldots, \xi_k^*\}$ such that $\forall x \in P$, $x \in \cap_{i=1}^k \xi_i^*$, then it holds that $z \in \cap_{i=1}^k \xi_i^*$. It follows that $\exists \Psi^* \subseteq \Psi$ such that $\forall x \in P$ and $\forall \psi^* \in \Psi^*$, $x\psi^* z$. This implies $\Gamma(P) \subseteq \Sigma(P)$.
- Otherwise, it follows by definition of $\Sigma(P)$ that $\Sigma(P) = \mathcal{S}$. Therefore, $\Gamma(P) \subseteq \Sigma(P)$, and the proof is completed.

□

APPENDIX B

APPENDIX FOR CHAPTER 7

B.1 Tables for Greedy Backward and Forward Selection

Table B.1: Results for the RAR-GA, SA and EDA approaches with greedy backward selection

Algorithm	Index	Best D	Success rate	Time(s)	Speed-up factor	Opt.
RAR-GA ($w = 1$)	Hang Seng	**0.00321150**	1.00	3.9	307.90	$4.92 \cdot 10^4$
	DAX	**2.53162860**	1.00	54.5	58.32	$6.97 \cdot 10^5$
	FTSE	1.92152413	0.94	61.3	104.15	$7.76 \cdot 10^5$
	S&P	4.71092502	0.97	61.1	107.62	$7.42 \cdot 10^5$
	Nikkei	**0.20197748**	1.00	154.1	64.20	$6.03 \cdot 10^5$
TransRAR GA	Hang Seng	**0.00321150**	1.00	3.08	161.56	$4.84 \cdot 10^4$
	DAX	**2.53162860**	1.00	43.9	44.79	$6.87 \cdot 10^5$
	FTSE	1.92152413	1.00	49.6	75.26	$7.64 \cdot 10^5$
	S&P	4.71092502	0.99	49.7	78.73	$7.30 \cdot 10^5$
	Nikkei	**0.20197748**	1.00	118.6	39.72	$5.94 \cdot 10^5$
SA	Hang Seng	**0.00321150**	1.00	2.3	652.13	$3.74 \cdot 10^4$
	DAX	**2.53162860**	0.97	19.4	148.31	$3.18 \cdot 10^5$
	FTSE	1.92152413	0.88	24.7	146.17	$4.21 \cdot 10^5$
	S&P	4.71092502	0.93	25.6	139.37	$4.05 \cdot 10^5$
	Nikkei	**0.20197748**	0.99	111.1	38.47	$2.71 \cdot 10^5$
UMDA (EDA)	Hang Seng	**0.00321150**	1.00	24.2	42.21	$5.19 \cdot 10^5$
	DAX	**2.53162860**	0.98	212.0	11.56	$4.02 \cdot 10^6$
	FTSE	1.92352385	0.90	243.6	10.19	$4.47 \cdot 10^6$
	S&P	4.71154380	0.95	239.7	11.27	$4.28 \cdot 10^6$
	Nikkei	**0.20197748**	1.00	339.0	7.46	$3.44 \cdot 10^6$
PBIL (EDA)	Hang Seng	**0.00321150**	1.00	22.9	100.12	$4.92 \cdot 10^5$
	DAX	**2.53162860**	1.00	203.5	22.06	$3.81 \cdot 10^6$
	FTSE	1.92152413	0.93	232.7	20.55	$4.25 \cdot 10^6$
	S&P	4.71092502	0.98	225.8	22.59	$4.06 \cdot 10^6$
	Nikkei	**0.20197748**	1.00	325.9	22.97	$3.27 \cdot 10^6$
PBIL_c (EDA)	Hang Seng	**0.00321150**	1.00	19.1	162.08	$3.73 \cdot 10^5$
	DAX	**2.53162860**	1.00	160.4	50.64	$2.88 \cdot 10^6$
	FTSE	1.92410411	0.89	181.3	48.18	$3.21 \cdot 10^6$
	S&P	4.71092501	0.96	176.3	54.79	$3.07 \cdot 10^6$
	Nikkei	**0.20197748**	1.00	235.8	74.51	$2.47 \cdot 10^6$
EMNA (EDA)	Hang Seng	**0.00321150**	1.00	28.2	350.80	$6.20 \cdot 10^5$
	DAX	2.53276344	0.85	243.4	158.86	$4.81 \cdot 10^6$
	FTSE	1.92598749	0.85	273.5	155.41	$5.35 \cdot 10^6$
	S&P	4.71092502	0.87	266.4	184.94	$5.12 \cdot 10^6$
	Nikkei	**0.20197748**	0.98	294.8	233.13	$4.12 \cdot 10^6$

Table B.2: Results for the RAR-GA, SA and EDA approaches with greedy forward selection

Algorithm	Index	Best D	Success rate	Time(s)	Speed-up factor	Opt.
RAR-GA ($w = 1$)	Hang Seng	**0.00321150**	1.00	8.1	148.25	$1.30 \cdot 10^5$
	DAX	**2.53162860**	1.00	67.4	47.16	$1.17 \cdot 10^6$
	FTSE	1.92152412	0.95	75.2	84.90	$1.29 \cdot 10^6$
	S&P	4.72021851	1.00	75.7	86.86	$1.35 \cdot 10^6$
	Nikkei	**0.20197748**	1.00	144.3	68.56	$3.46 \cdot 10^6$
TransRAR GA	Hang Seng	**0.00321150**	1.00	6.83	72.85	$1.29 \cdot 10^5$
	DAX	**2.53162860**	1.00	56.7	34.68	$1.16 \cdot 10^6$
	FTSE	1.92152413	0.98	63.0	59.22	$1.28 \cdot 10^6$
	S&P	4.72021851	1.00	64.1	61.04	$1.34 \cdot 10^6$
	Nikkei	**0.20197748**	1.00	125.1	37.65	$3.45 \cdot 10^6$
SA	Hang Seng	**0.00321150**	1.00	6.5	230.75	$1.18 \cdot 10^5$
	DAX	**2.53162860**	0.98	32.4	88.81	$7.84 \cdot 10^5$
	FTSE	1.92152413	0.89	39.0	95.57	$9.35 \cdot 10^5$
	S&P	4.72060631	0.91	42.0	84.95	$1.04 \cdot 10^6$
	Nikkei	**0.20197748**	0.98	102.5	41.70	$3.13 \cdot 10^6$
UMDA (EDA)	Hang Seng	**0.00321150**	1.00	28.2	32.23	$6.00 \cdot 10^5$
	DAX	**2.53162860**	0.97	225.6	10.86	$4.49 \cdot 10^6$
	FTSE	1.92336941	0.91	257.9	9.63	$4.99 \cdot 10^6$
	S&P	4.72257456	0.96	252.6	10.69	$4.89 \cdot 10^3$
	Nikkei	**0.20197748**	1.00	329.4	7.68	$6.30 \cdot 10^6$
PBIL (EDA)	Hang Seng	**0.00321150**	1.00	27.2	84.29	$5.73 \cdot 10^5$
	DAX	**2.53162860**	1.00	216.3	20.75	$4.29 \cdot 10^6$
	FTSE	1.92234549	0.95	250.9	19.06	$4.76 \cdot 10^6$
	S&P	4.72021851	0.99	241.5	21.12	$4.68 \cdot 10^6$
	Nikkei	**0.20197748**	1.00	317.6	23.57	$6.12 \cdot 10^6$
PBIL_c (EDA)	Hang Seng	**0.00321150**	1.00	23.2	133.44	$4.53 \cdot 10^5$
	DAX	**2.53162860**	1.00	171.9	47.25	$3.36 \cdot 10^6$
	FTSE	1.92340339	0.89	193.5	45.14	$3.72 \cdot 10^6$
	S&P	4.72021851	0.96	188.3	51.30	$3.69 \cdot 10^6$
	Nikkei	**0.20197748**	1.00	224.4	78.29	$5.33 \cdot 10^6$
EMNA (EDA)	Hang Seng	**0.00321150**	1.00	32.4	305.32	$7.01 \cdot 10^5$
	DAX	**2.53162860**	0.88	254.7	151.82	$5.28 \cdot 10^6$
	FTSE	1.92289840	0.88	248.2	171.26	$5.87 \cdot 10^6$
	S&P	4.72021851	0.88	276.4	178.25	$5.73 \cdot 10^6$
	Nikkei	**0.20197748**	0.98	286.2	240.13	$6.97 \cdot 10^6$

APPENDIX C

APPENDIX FOR CHAPTER 8

C.1 Proof of Theorem 8.1

Lemma. Let $\mathbf{U}$ be an UpDown matrix. The triplet of taxa $ab|c$ belongs to the tree if and only if for the submatrix $\mathbf{U}_{abc}$ composed of the rows and columns of a, b and c, there exists a permutation π of rows and columns such that the following inequalities hold:

$$u_{ab} < u_{ac} \tag{C.1}$$

$$u_{ba} < u_{bc} \tag{C.2}$$

$$u_{ca} = u_{cb} \tag{C.3}$$

$$u_{ac} - u_{ab} = u_{bc} - u_{ba} \tag{C.4}$$

Proof. We will prove first that given the triplet $ab|c$, the required UpDown submatrix satisfies the relations (C.1)-(C.4). Therefore, suppose that triplet $ab|c$ belongs to the tree. Consider the 3×3 matrix:

$$\begin{pmatrix} 0 & u_{ab} & u_{ac} \\ u_{ba} & 0 & u_{bc} \\ u_{ca} & u_{cb} & 0 \end{pmatrix}$$

If triplet $ab|c$ belongs to the tree, then obviously $u_{ab} < u_{ac}$ holds. It is easy to see that also $u_{ba} < u_{bc}$ holds, because since the triplet is symmetric, $ab|c \equiv ba|c$. We can further state that u_{ca} and u_{cb} must have the same value since the LCA of c and a and the LCA of c and b are equal, therefore also the up-values. Finally it holds that $u_{ac} - u_{ab} = u_{bc} - u_{ba}$, i.e. the distance from the LCA of a and b to the LCA of a and c is the same as the one from the LCA of b and a to the LCA of b and c since both former and both latter LCAs are equal. On the other hand, suppose that we have an UpDown submatrix $\mathbf{U}_{abc}$ and that (C.1)-(C.4) hold. If $u_{ab} < u_{ac}$, then taxa a and b are in the same partition, taking into account the level of their least common ancestor, while c is not. If $u_{ba} < u_{bc}$, then

the same argumentation applies for symmetry reasons and the triplet $ab|c$ belongs to the tree.

□

BIBLIOGRAPHY

Adler, I., Karmarkar, N., Resende, M. G. C., and Veiga, G. (1989). An implementation of Karmarkar's algorithm for linear programming. *Mathematical Programming*, 44:297–335.

Amann, M. and Zimmermann, H. (2001). Tracking error and tactical asset allocation. *Financial Analysts Journal*, 57(2):32–43.

Andreatta, A. A. and Ribeiro, C. C. (2002). Heuristics for the phylogeny problem. *Journal of Heuristics*, 8:429–447.

Argyriou, A., Evgeniou, T., and Pontil, M. (2007). Multi-task feature learning. In *Advances in Neural Information Processing Systems 19*. MIT Press.

Balas, E. and Zemel, E. (1980). An algorithm for large zero-one knapsack problems. *Operations Reserach*, 28:1130–1154.

Baluja, S. (1994). Population-based incremental learning: A method for integrating genetic search based function optimization and competitive learning. Technical Report CMU-CS-94-163, Carnegie Mellon University.

Baluja, S. and Caruana, R. (1995). Removing the genetics from the standard genetic algorithm. In Prieditis, A. and Russell, S., editors, *Proceedings of the International Conference on Machine Learning*, pages 38–46. Morgan-Kaufmann.

Beasley, J. E. (1990). Or-library: Distributing test problems by electronic mail. *Journal of the Operational Research Society*, 41 (11):1069–1072.

Beasley, J. E., Meade, N., and Chang, T. (2003). An evolutionary heuristic for the index tracking problem. *European Journal of Operations Research*, 148 (3):621–643.

Best, M. and Hlouskova, J. (2008). Quadratic programming with transaction costs. *Computers and Operations Research*, 25(35):18–33.

Bienstock, D. (1995). Computational study of a family of mixed-integer quadratic programming problems. In Balas, E. and Clausen, J., editors, *Integer Programming and*

Combinatorial Optimization: 4th International IPCO Conference, Lecture Notes in Computer Science 920. Springer-Verlag, Berlin.

Bishop, C. M. (2006). *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA.

Blum, C. and Roli, A. (2003). Metaheuristics in combinatorial optimization: Overview and conceptual comparison. *ACM Computing Surveys*, 35 (3):268–308.

Boyd, S. and Vandenberghe, L. (2004). *Convex Optimization*. Cambridge University Press.

Brodie, J., Daubechies, I., Mol, C. D., Giannone, D., and Loris, I. (2009). Sparse and stable markowitz portfolios. *Proceedings of the National Academy of Sciences*, 106(30):12267–12272.

Bryant, D. (2003). A classification of consensus methods for phylogenies. In Janowitz, M., Lapointe, F.-J., McMorris, F., Mirkin, B., and Roberts, F., editors, *Bioconsensus*, DIMACS, pages 163–184. AMS.

Buckley, I. and Korn, R. (1998). Optimal index tracking under transaction costs and impulse control. *International Journal of Theoretical and Applied Finance*, 1(3):315–330.

Cadima, J. and Jolliffe, I. (1995). Loadings and correlations in the interpretation of principal components. *Applied Statistics*, (22):203–214.

Carvalho, C. M., Polson, N. G., and Scott, J. G. (2009). Handling sparsity via the horseshoe. *Journal of Machine Learning Research W&CP*, 5:73–80.

Chang, T. J., Meade, N., Beasley, J. E., and Sharaiha, Y. M. (2000). Heuristics for cardinality constrained portfolio optimisation. *Computers and Operations Research*, 27:1271–1302.

Cormen, T. H., Leiserson, C. H., and Rivest, R. L. (1990). *Introduction to Algorithms*. The MIT Press.

Cotta, C. and Troya, J. M. (2003). Information processing in transmitting recombination. *Applied Mathematics Letters*, 16:2003.

Crama, Y. and Schyns, M. (1999). Simulated annealing for complex portfolio selection problems. Technical report, Groupe d'Etude des Mathematiques du Management et de l'Economie 9911, Universie de Liege.

Dantzig, G. (1998). *Linear programming and extensions*. Princeton University Press.

d'Aspremont, A., Bach, F., and Ghaoui, L. E. (2008a). Optimal solutions for sparse principal component analysis. *Journal of Machine Learning Research*, 9:1269–1294.

d'Aspremont, A., Ghaoui, L. E., Jordan, M., and Lanckriet, G. (2004). A direct formulation for sparse PCA using semidefinite programming. In *Advances in Neural Information Processing Systems (NIPS)*, Vancouver, BC, December 2004.

d'Aspremont, A., Ghaoui, L. E., Jordan, M., and Lanckriet, G. (2007). A direct formulation for sparse PCA using semidefinite programming. *SIAM Review*, 49(3):434–448.

d'Aspremont, A., Ghaoui, L. E., Jordan, M., and Lanckriet, G. (2008b). MATLAB code for DSPCA. http://www.princeton.edu/~{}aspremon/DSPCA.htm.

Dawkins, R. (1976). *The Selfish Gene*. Oxford University Press.

Day, W., Johnson, D., and Sankoff, D. (1986). The computational complexity of inferring rooted phylogenies by parsimony. *Mathematical Biosciences*, 88:33–42.

DeMiguel, V., Garlappi, L., Nogales, F. J., and Uppal, R. (2009a). A generalized approach to portfolio optimization: Improving performance by constraining portfolio norms. *Management Science*, 55(5):798–812.

DeMiguel, V., Garlappi, L., and Uppal, R. (2009b). Optimal versus naive diversification: How inneficient is the 1/n portfolio strategy? *Review of Financial Studies*, 5(22):1915–1953.

DeMiguel, V. and Nogales, F. J. (2009). Portfolio selection with robust estimation. *Operations Research*, 57(3):560–577.

DiGaspero, L., DiTollo, G., Roli, A., and Schaerf, A. (2011). Hybrid metaheuristics for constrained portfolio selection problem. *Quantitative Finance*, 11(10):1473–1488.

DiGaspero, L., DiTollo, G., Schaerf, A., and Roli, A. (2007). A hybrid solver for constrained portfolio selection problems: preliminary report. In *Proceedings of Learning and Intelligent Optimization (LION2007)*.

Dorigo, M. and Gambardella, L. M. (1997). Ant colony system: a cooperative learning approach to the traveling salesman problem. *IEEE Transactions on Evolutionary Computation*, 1:53–66.

Eberhart, R. and Kennedy, J. (1995). A new optimizer using particle swarm theory. In *Proceedings of the Sixth International Symposium on Micromachine and Human Science, Nagoya, Japan*, pages 39–43.

Efron, B., Hastie, T., Johnstone, I., and Tibshirani, R. (2004). Least angle regression. *Annals of Statistics*, 32(2):407–451.

El-Abd, M. and Kamel, M. (2005). *A Taxonomy of Cooperative Search Algorithms*, pages 32–41. Springer Verlag.

Fletcher, R. (2000). *Practical Methods of Optimization*. Wiley.

Foulds, L. and Graham, R. (1982). The steiner problem in phylogeny is NP-complete. *Advances in Applied Mathematics*, 3.

George, E. I. and McCulloch, R. E. (1997). Approaches for Bayesian variable selection. *Statistica Sinica*, 7(2):339–373.

Gill, P. E., Murray, W., Saunders, M. A., and Wright, M. H. (1991). Inertia-controlling methods for general quadratic programming. *SIAM Review*, 33:1–36.

Gilli, M. and Këllezi, E. (2001a). A global optimization heuristic for portfolio choice-with VaR and expected shortfall. In *Computational Methods in Decision-Making, Economics and Finance*. Kluwer Academic Publishers.

Gilli, M. and Këllezi, E. (2001b). Threshold accepting for index tracking. *Computing in Economics and Finance*, 72.

Glover, F. (1986). Future paths for integer programming and links to artificial intelligence. *Computers and Operations Research*, 13:533–549.

Goldberg, D. (1989). *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Weasley, Reading, MA.

Han, S., Yoon, Y., and Cho, K. (2007). Inferring biomolecular interaction networks based on convex optimization. *Computational Biology and Chemistry*, 31:347–354.

Hansen, P. and Mladenovic, N. (2003). A tutorial on variable neighborhood search.

Harjunkoski, I. and Grossmann, I. E. (2002). Decomposition techniques for multi-stage scheduling problems using mixed-integer and constraint programming methods. *Comp. Chem. Engng*, 26:1533–1552.

Hershenson, M., Boyd, S., and Lee, T. (2001). Optimal design of a CMOS op-amp via geometric programming. *IEEE Transactions on Computer-Aided Design*, 20(1):1–21.

Holland, J. H. (1975). *Adaptation in Natural and Artificial Systems*. University of Michigan Press.

Holmes, S. (1999). *Phylogenies: An overview*, pages 81–119. Springer Verlag.

Huang, K. and Jane, C. (2009). A hybrid model for stock market forecasting and portfolio selection based on ARX, Grey system and RS theories. *Expert Systems with Applications*, pages 5387–5392.

Jacob, L., Obozinski, G., and Vert, J. (2009). Group lasso with overlaps and graph lasso. In *Proceedings of the 26th International Conference on Machine Learning (ICML)*.

Jagannathan, R. and Ma, T. (2003). Risk reduction in large portfolios: Why imposing the wrong constraint helps. *The Journal of Finance*, 58(4):1651–1683.

Jeffers, J. (1967). Two case studies in the application of principal components. *Applied Statistics*, (16):225–236.

Jeong, S., Hasegawa, S., Shimoyama, K., and Obayashi, S. (2009). Development and investigation of efficient GA/PSO-hybrid algorithm applicable to real world design optimization. *IEEE Computational Intelligence Magazine*, 4(3):36–44.

Jeurissen, R. and van den Berg, J. (2005). Index tracking using a hybrid genetic algorithm. In *ICSC Congress on Computational Intelligence Methods and Applications 2005*.

Jeurissen, R. and van den Berg, J. (2008). Optimized index tracking using a hybrid genetic algorithm. In *Proceedings of the IEEE World Congress on Evolutionary Computation (CEC2008)*, pages 2327–2334.

Jobst, N. J., Horniman, M. D., Lucas, C. A., and Mitra, G. (2001). Computational aspects of alternative portfolio selection models in the presence of discrete asset choice constraints. *Quantitative Finance*, 1:1–13.

Jolliffe, I., Trendafilov, N., and Uddin, M. (2003). A modified principal component techique based on the lasso. *Journal of Computational and Graphical Statistics*, (12):531–547.

Kellerer, H., Pferschy, U., and Pisinger, D. (2004). *Knapsack Problems*. Springer Verlag.

Kim, J., Hwang, I., Kim, Y., and Moon, B. (2011). Genetic approaches for graph partitioning: A survey. In *Proceedings of the 13th annual conference on Genetic and Evolutionary Computation (GECCO 2011), Dublin, Ireland*, pages 473–480.

Kim, J. and Warnow, T. (1999). Tutorial on phylogenetic tree estimation. In et. al., T. L., editor, *Proceedings of the 7th International Conference on Intelligent Systems for Molecular Biology*, pages 196–205. AAAI Press.

Kirkpatrick, S., Gelatt, C. D., and M. P. Vecchi, J. (1983). Optimization by simulated annealing. *Science*, 4598:671–679.

Krasnogor, N. and Smith, J. (2000). A memetic algorithm with self-adaptive local search: Tsp as a case study. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2000)*, pages 987–994. Morgan Kaufmann.

Ku, S. and Lee, B. (2001). A set-oriented genetic algorithm and the knapsack problem. In *Proceedings of the IEEE World Congress on Evolutionary Computation (CEC2001)*.

Ladanyi, L., Ralphs, T., Guzelsoy, M., and Mahajan, A. (2009). SYMPHONY. https://projects.coin-or.org/SYMPHONY.

Larrañaga, P. and Lozano, J. A., editors (2002). *Estimation of Distribution Algorithms: A New Tool for Evolutionary Computation*. Kluwer Academic Publishers.

Larrañaga, P., Lozano, J. A., and Bengoetxea, E. (2001). Estimation of distribution algorithms based on multivariate normal and gaussian networks. Technical report, Department of Computer Science and Artificial Intelligence, University of the Basque Country.

Ledoit, O. and Wolf, M. (2004). A well-conditioned estimator for large-dimensional covariance matrices. *J. Multivar. Anal.*, 88(2):365–411.

Leonard, T. and Hsu, J. (1992). Bayesian inference for a covariance matrix. *Annals of Statistics*, (4):1669–1696.

Lobo, M., Fazel, M., and Boyd, S. (2007). Portfolio optimization with linear and fixed transaction costs. *Annals of Operations Research, special issue on financial optimization*, 152 (1):376–394.

Lübbecke, M. E. and Desrosiers, J. (2005). Selected topics in column generation. *Operations Research*, 53(6):1007–1023.

Markowitz, H. (1952). Portfolio selection. *Journal of Finance*, 7:77–91.

Markowitz, H. (1987). *Mean-variance analysis in portfolio choice and capital markets*. Basil Blackwell.

Martello, S., Pisinger, D., and Toth, P. (1999). Dynamic programming and strong bounds for the 0-1 knapsack problem. *Management science*, 45:414–424.

Michalewicz, Z. (1996). *Genetic Algorithms + Data Structures = Evolution Programs*. Springer Verlag.

Miller, R. E. and Thatcher, J. W., editors (1972). *Reducibility among combinatorial problems*, pages 85–103. Plenum Press.

Mitchell, J. and Braun, S. (2004). Rebalancing an investment portfolio in the presence of convex transaction costs. Technical report, Department of Mathematical Sciences, Rensselaer Polytechnic Institute.

Moghaddam, B., Weiss, Y., and Avidan, S. (2005). Spectral bounds for sparse PCA. In *Advances in Neural Information Processing Systems (NIPS2005)*.

Moral-Escudero, R., Ruiz-Torrubiano, R., and Suarez, A. (2006). Selection of optimal investment portfolios with cardinality constraints. In *Proceedings of the IEEE World Congress on Evolutionary Computation*, pages 2382–2388.

Moscato, P. and Cotta, C. (2003). A gentle introduction to memetic algorithms. In *Handbook of Metaheuristics*, pages 105–144. Kluwer Academic Publishers.

Muehlenbein, H. (1998). The equation for response to selection and its use for prediction. *Evolutionary Computation*, 5:303–346.

Osman, I. H. and Laporte, G. (1996). Metaheuristics: A bibliography. *Annals of Operations Research*, 63 (5):513–623.

Padberg, M. W. and Rinaldi, G. (1991). A branch-and-cut algorithm for the solution of large scale traveling salesman problems. *SIAM Review*, 33:60–100.

Perold, A. (1984). Large-scale portfolio optimization. *Management Science*, 30(10):1143–1160.

Pirkwieser, S. and Ruiz-Torrubiano, R. (2007). Simple neighborhoods. Technical report, Institute for Computer Graphics and Algorithms, Vienna University of Technology, Vienna, Austria.

Pisinger, D. (2000). A minimal algorithm for the bounded knapsack problem. *INFORMS Journal on Computing*, 12(1):75–82.

Pisinger, D. (2005). Where are the hard knapsack problems? *Computers & Operations Research*, pages 2271–2284.

Pisinger, D. (2007). The quadratic knapsack problem - a survey. *Discrete Applied Mathematics*, 155:623–648.

Pogue, G. (1970). An extension of the markowitz portfolio selection model to include variable transactions' costs, short sales, leverage policies and taxes. *The Journal of Finance*, 25(5):1005–1027.

Poirriez, V., Yanev, N., and Andonov, R. (2009). A hybrid algorithm for the unbounded knapsack problem. *Discrete optimization*, 6:110–124.

Puchinger, J., Raidl, G., and Pferschy, U. (2010). The multidimensional knapsack problem: Structure and algorithms. *INFORMS Journal on Computing*, 22(2):250–265.

Qaurooni, D. (2011). A memetic algorithm for course timetabling. In *Proceedings of the 13th annual conference on Genetic and Evolutionary Computation (GECCO 2011), Dublin, Ireland*, pages 435–442.

Radcliffe, N. J. (1990). *Genetic Neural Networks on MIMD Computers*. PhD thesis, University of Edinburgh.

Radcliffe, N. J. (1993). Genetic set recombination. In *Foundations of Genetic Algorithms*. Morgan Kaufmann Pulishers.

Radcliffe, N. J. (1994). The algebra of genetic algorithms. *Annals of Maths and Artificial Intelligence*, pages 339–384.

Radcliffe, N. J. and George, F. (1993). A study in set recombination. In *Proceedings of the 5th International Conference on Genetic Algorithms*. Morgan Kaufmann Pulishers.

Radcliffe, N. J. and Surry, P. D. (1995). Fitness variance of formae and performance prediction. In *Foundations of Genetic Algorithms III*. Morgan Kaufmann Pulishers.

Raidl, G. R. (2006). A unified view on hybrid metaheuristics. In *Proceedings of the Hybrid Metaheuristics Workshop*, volume 4030 of *Lecture Notes in Computer Science*, pages 1–12. Springer.

Rudolf, M., Wolter, H., and Zimmermann, H. (1999). A linear model for tracking error minimization. *Journal of Banking and Finance*, 23(1):85–103.

Ruiz-Torrubiano, R., García-Moratilla, S., and Suárez, A. (2010). Optimization problems with cardinality constraints. In *Computational Intelligence in Optimization: Implementations and Applications*. Springer Verlag.

Ruiz-Torrubiano, R. and Suárez, A. (2007a). A hybrid optimization approach to index tracking. *Accepted for publication in Annals of Operations Research.*

Ruiz-Torrubiano, R. and Suárez, A. (2007b). Use of heuristic rules in evolutionary methods for the selection of optimal investment portfolios. In *Proceedings of the IEEE World Congress on Evolutionary Computation CEC 2007, September 25th-28th, Singapore.*

Ruiz-Torrubiano, R. and Suarez, A. (2010). Hybrid approaches and dimensionality reduction for portfolio selection with cardinality constraints. *IEEE Computational Intelligence Magazine*, 5(2):92–107.

Ruiz-Torrubiano, R. and Suárez, A. (2011). The transrar crossover operator for genetic algorithms with set encoding. In *Proceedings of the 13th annual conference on Genetic and Evolutionary Computation (GECCO 2011), Dublin, Ireland*, pages 489–496.

Santos, A., Nogales, F., Ruiz, E., and VanDijk, D. (2012). Optimal portfolios with minimum capital requirements. *Accepted for publication: Journal of Banking and Finance.*

Schaerf, A. (2002). Local search techniques for constrained portfolio selection problems. *Computational Economics*, 20:177–190.

Sebag, M. and Ducoulombier, A. (1998). Extending population-based incremental learning to continuous search spaces. *Lecture Notes in Computer Science*, 1498:418–427.

Seeger, M. W. (2008). Bayesian inference and optimal design for the sparse linear model. *The Journal of Machine Learning Research*, 9:759–813.

Shapcott, J. (1992). Index tracking: genetic algorithms for investment portfolio selection. Technical report, EPCC-SS92-24, Edinburgh, Parallel Computing Centre.

Simões, A. and Costa, E. (2001). An evolutionary approach to the zero/one knapsack problem: Testing ideas from biology. In *Proceedings of the Fifth International Conference on Artificial Neural Networks andGenetic Algorithms ICANNGA.*

Speer, N., Spieth, C., and Zell, A. (2004). A memetic co-clustering algorithm for gene expression profiles and biological annotation. In *Proceedings of the IEEE Congress on Evolutionary Computation 2004*, pages 1631–1638. IEEE Press.

Streichert, F. and Tamaka-Tamawaki, M. (2006). The effect of local search on the constrained portfolio selection problem. In *Proceedings of the IEEE World Congress on Evolutionary Computation (CEC2006)*, pages 2368–2374, Vancouver, Canada.

Streichert, F., Ulmer, H., and Zell, A. (2004). Evaluating a hybrid encoding and three crossover operators on the constrained portfolio selection problem. In *Proceedings of the Congress on Evolutionary Computation (CEC2004)*, volume 1, pages 932–939.

Swofford, S. (1991). *Phylogenetic analysis of DNA sequences*, pages 295–333. Oxford University Press.

Tabata, Y. and Takeda, E. (1995). Bicriteria optimization problem of designing an index fund. *Journal of the Operational Research Society*, 46(8):1023–1032.

Talbi, E. (2002). A taxonomy of hybrid metaheuristics. *Journal of Heuristics*, 8:541–564.

Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society B*, 58(1):267–268.

Tipping, M. E. (2001). Sparse Bayesian learning and the relevance vector machine. *The Journal of Machine Learning Research*, 1:211–244.

Turlach, B. A. (2005). On algorithms for solving least squares problems under an l1 penalty or an l1 constraint. In *2004 Proceedings of the Americal Statistical Association, Statistical Computing Section*, pages 2572–2577. Americal Statistical Association.

Vandenberghe, L. and Boyd, S. (1996). Semidefinite programming. *SIAM Review*, 38 (1):49–95.

Wang, J., Shan, H., Shasha, D., and Piel, W. (2003). TreeRank: A similarity measure for nearest neighbor searching in phylogenetic databases. In *Proceedings of the 15th International Conference on Scientific and Statistical Database Management, Cambridge, MA*.

Wang, J. and Zhu, J. (2009). Portfolio theory of information retrieval. In *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '09, pages 115–122, New York, NY, USA. ACM.

Wolpert, D. and Macready, W. (1997). No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*, 1:67–82.

Wu, M., Schlkopf, B., and Bakir, G. (2006). A direct method for building sparse kernel learning algorithms. *Journal of Machine Learning Research*, 7:603–624.

Yeh, W. (2006). An efficient memetic algorithm for the multi-stage supply chain network problem. *The International Journal of Advanced Manufacturing Technology*, 29:803–813.

Yoshimoto, A. (1996). The mean-variance approach to portfolio optimization subject to transaction costs. *Journal of the Operations Research Society of Japan*, 39(1):99–117.

Yu, C. and Li, H. (2000). A robust optimization model for stochastic logistic problems. *International Journal of Production Economics*, 64:385–397.

Zou, H., Hastie, T., and Tibshirani, R. (2006). Sparse principal component analysis. *Journal of Computational and Graphical Statistics*, 15 (2):265–286.