Hybrid Metaheuristics for Constrained Portfolio Selection Problems

Luca Di Gaspero[*], Andrea Schaerf
DIEGM, University of Udine, Italy
{l.digaspero | schaerf}@uniud.it

Giacomo di Tollo
Science Department, University "G. D'Annunzio", Italy
ditollo@unich.it

Andrea Roli
DEIS – Campus of Cesena, University of Bologna, Italy
andrea.roli@unibo.it

Abstract

Portfolio selection is a relevant problem arising in finance and economics. While its basic formulations can be efficiently solved through linear or quadratic programming, its more practical and realistic variants, which include various kinds of constraints and objectives, have in many cases to be tackled by heuristics. In this work, we present a hybrid technique that combines a local search metaheuristic, as *master* solver, with a quadratic programming procedure, as *slave* solver. Experimental results show that the approach is very promising as it reaches regularly the optimal solution and thus achieves results comparable with, or superior to, the state of the art solvers, including widespread commercial software tools (CPLEX 11.0.1 and MOSEK 5). Finally, the paper proposes a detailed analysis of the behavior of the technique in variour settings of the constraints, thus showing how the performances are dependent on the features of the instance.

1 Introduction

The *portfolio selection* problem consists in selecting a set of *assets*, and the share invested in each asset, that provides the investor a minimum required return and minimizes the *risk*. One of the main contributions in this problem is the seminal work by Markowitz

[*]Corresponding author. Address: via delle Scienze 208, I-33100, Udine, Italy — Phone: +39 0432 55 8242 — Fax: +39 0432 55 8251

(1952), who introduced the so-called *mean-variance* model, which assumes that all information about current and future asset prices are normal random variables that are described by their distribution parameters (mean and variance). In this model, the variance of the portfolio returns is conceived as a measure of investor's risk therefore the problem can be formulated as an optimization problem over real-valued variables with a quadratic objective function and linear constraints. In such a framework, a portfolio will exhibit the benefit of *diversification*, that is, the total portfolio risk is less than the weighted sum of the risks of the composing assets. The model satisfies the stochastic dominance criteria of *non-satiation* (investor's preference of a higher to a lower return) and *risk-aversion* (investor's preference of a lower to a higher risk).

In this paper we consider the basic objective function introduced by Markowitz, and we take into account three additional constraints: the *cardinality* constraint, which limits the number of assets; the *quantity* constraint, which fixes minimal and maximal shares of each asset included in the portfolio; and the *preassignments* that force some specific assets to be included in the portfolio. Assets are considered as perfectly divisible (i.e., they can be traded in any quantity) and non-negativity constraints are imposed (so forbidding short sales). These feature are binding for small investors, yet negligible for great and institutional fund managers.

Cardinality constraints are justified by the preference of investors to have, on the one hand, a limited number of assets to deal with and, on the other hand, a certain degree of diversification. Indeed, the Markowitz's framework seems to suggest that investors are eager to hold as many different assets as possible, as any further asset insertion in the portfolio will contribute in lowering the risk (and increasing investor utility). Instead, in practice, the administration of huge portfolios appears to be cumbersome, and there is evidence that investors prefer only a small degree of diversification (Blume and Friend, 1975; Guiso et al., 1996; Jansen and van Dijik, 2002). Furthermore, the best diversification can be obtained by choosing a small subset of assets (Maringer, 2005), so the choice of a good value for the number of assets to be included is crucial.

Quantity constraints use is twofold: ceiling constraints (i.e., upper bounds on assets shares) are introduced to avoid excessive exposure to a specific asset and in some case are imposed by law; floor constraints (lower bounds) are used to avoid the cost of administrating tiny portions of assets.

Preassignments are used to model investors preferences and/or surplus assets from previous investing periods.

For the solution of the optimization problem we devise a hybrid solution technique based on a local search metaheuristic (see, e.g, Hoos and Stützle (2005)) for selecting the assets to be included in the portfolio, which at each step resorts to a quadratic programing (QP) solver for computing the best allocation for the chosen assets. The QP procedure implements the Goldfarb-Idnani dual algorithm for strictly convex quadratic programs Goldfarb and Idnani (1983).

In this paper, we describe our hybrid approach and we report our experimental analyses regarding the comparison with previous results (on their formulation) and with exact solvers, the structure of the instances, and the influence of specific constraints. The experiments show that our solver is able to find regularly the optimal solution and compares favorably with previous results in the literature.

The paper is organized as follows: In Section 2 we present the problem formulation and in Section 3 we discuss the related work. In Section 4 we introduce the general solution techniques, and in Section 4 we present our solver detailing its components. Section 5 collects the results of our solver and shows how they compare with previous ones. Finally, in Section 6 we draw some conclusions and point out our plans for further work.

2 Problem definition

Following Markowitz (1952), we are given a set of n *assets*, $A = \{a_1, \dots, a_n\}$. Each asset a_i has an associated real-valued *expected return* (per period) r_i, and each pair of assets $\langle a_i, a_j \rangle$ has a real-valued return *covariance* σ_{ij}. The matrix $\sigma_{n \times n}$ is symmetric and the diagonal elements σ_{ii} represent the return *variance* of assets a_i. A positive value R represents the minimum required (expected) return. The values r_i and σ_{ij} are usually estimated from past data and are relative to one fixed period of time.

A portfolio is a vector of real values $X = \{x_1, \dots, x_n\}$ such that each x_i represents the fraction invested in the asset a_i. The value $\sum_{i=1}^{n} \sum_{j=1}^{n} \sigma_{ij} x_i x_j$ represents the variance of the portfolio, and is considered as the measure of the risk associated with the portfolio. Consequently, the problem is to minimize the overall variance, still ensuring the minimum required return R. The formulation of the basic (unconstrained) problem is thus the following.

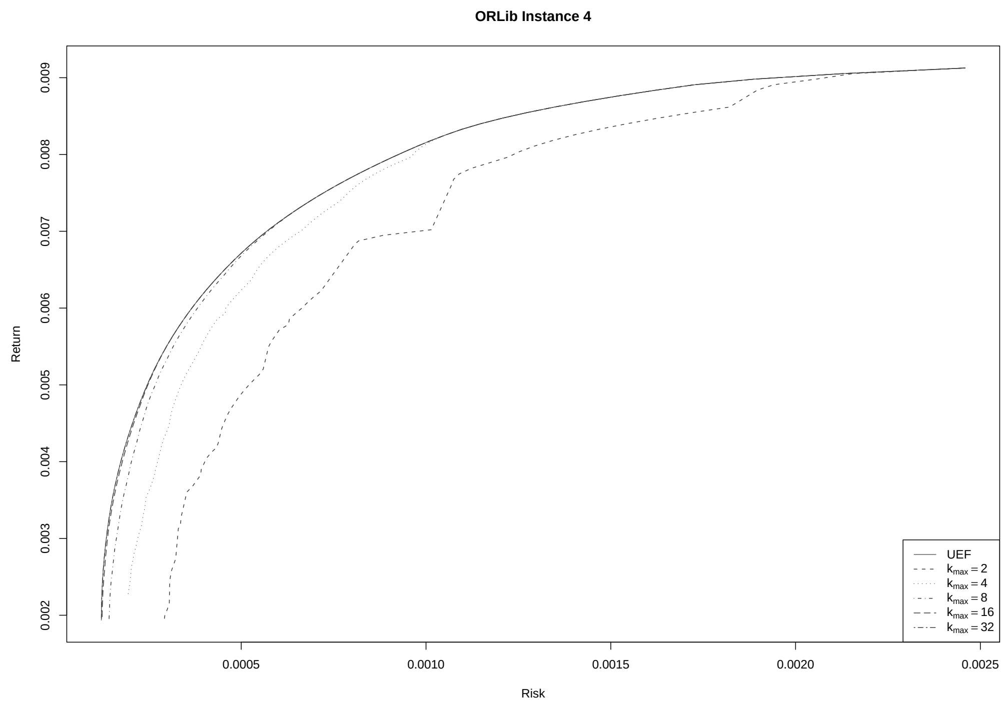


Figure 1: Unconstrained and constrained efficient frontier.

$$\begin{aligned}
\min F(X) = \quad & \sum_{i=1}^{n}\sum_{j=1}^{n}\sigma_{ij}x_i x_j \\
s.t. \quad & \sum_{i=1}^{n} r_i x_i \geq R && (1)\\
& \sum_{i=1}^{n} x_i = 1 && (2)\\
& x_i \geq 0 \qquad (i = 1, \dots, n) && (3)
\end{aligned}$$

This is a quadratic programming problem, and nowadays it can be solved optimally, for real instances of typical size, using available tools.

Since the minimum required return R can be considered as a parameter of the problem, by solving the problem as a function of R, ranging over a finite and discretized domain, we obtain the so-called *unconstrained (Pareto) efficient frontier* (UEF), that gives for each return the minimum associated risk. One of the advantages of the Markowitz model is that this Pareto front can be drawn without explicit knowledge about the investor exact utility as long as investors are assumed to be rational and risk-averse.

The UEF for one of the benchmark instances employed in this study is provided in Figure 1 (the lowest black solid line).

Although the classical Markowitz's model is extremely useful from the theoretical point of view, dealing with real-world financial markets imposes some additional constraints that are going to be considered in this work. In order to model them, we add to the formulation a vector of n binary decision variables Z such that if asset i is in the solution then $z_i = 1$.

The constraints relating Z and X variables are formulated as follows:

$$x_i \leq z_i \quad (i = 1, \ldots, n) \tag{4}$$

Cardinality constraint: The number of assets that compose the portfolio is bounded: we give two values k_{min} and k_{max} (with $1 \leq k_{min} \leq k_{max} \leq n$) such that:

$$k_{min} \leq \sum_{i=1}^{n} z_i \leq k_{max} \tag{5}$$

Quantity constraints: The quantity of each asset i that is included in the portfolio is limited within a given interval: we give a minimum ϵ_i and a maximum δ_i for each asset i, such that either $z_i = 0$ (and $x_i = 0$) or $\epsilon_i \leq x_i \leq \delta_i$. This is expressed in linear constraints as follows.

$$\epsilon_i z_i \leq x_i \leq \delta_i z_i \quad (i = 1, \ldots, n) \tag{6}$$

Preassignments: Certain assets are preassigned in the portfolio: we give a binary vector P (of size n) such that $p_i = 1$ if and only if a_i is preassigned in the portfolio and we require that

$$z_i \geq p_i \quad (i = 1, \ldots, n) \tag{7}$$

Notice that preassignments and minimum cardinality constraints are meaningful only in presence of a minimum quantity $\epsilon_i > 0$, otherwise they can be satisfied by assigning $z_i = 1$ and $x_i = 0$.

Notice also that some constraints can be implied by other ones. For example, if the number of preassigned assets $\sum_{i=1}^{n} p_i$ is greater than k_{min}, then k_{min} is implicitly set to this number. Similarly, if the minimum quantity is $\epsilon_i = 0.1$ for $i = 1, \ldots, n$, then k_{max} is implicitly set to 10.

We call CEF the analogous of the UEF for the constrained problem. In Figure 1 we plot the CEF found by our solver for the values $\epsilon_i = 0.01$, $\delta_i = 1$ (for $i = 1, \ldots, n$), $k_{min} = 1$, k_{max} varying from 4 to 12, and no preassignments. For higher values of k_{max}

the cardinality constraint reduces its effect and the curve is almost indistinguishable from the UEF, indeed the distance[1] among the CEF and the UEF becomes smaller than 10^{-3} for the instance at hand.

It is common opinion that Constraints (5) and (6) make it computationally expensive to solve real-world instances of the problem with proof of optimality (see, e.g., Jobst et al. (2001)). Indeed, just adding the maximum cardinality constraints the underlying decision problem becomes NP-complete, as proven by Bienstock (1996). Therefore, either simplified models are considered, such as formulations with linear objective function (Mansini et al., 2003; Mansini and Speranza, 2005), or approximate methods are applied, like in the present work.

3 Related Work

In this section, we review the papers that used a similar solution technique for this or similar formulations of the portfolio selection problem. For a comprehensive overview of the formulations presented in the literature we forward the interested reader to di Tollo and Roli (2006).

The use of local search techniques for constrained Markowitz model has been proposed by several authors, namely Chang et al. (2000), Schaerf (2002), Crama and Schyns (2003), Armañanzas and Lozano (2005), Moral-Escudero et al. (2006), and Fernández and Gómez (2007).The relationship between these works and our is discussed in details in Section 5.3, in which we compare the results.

Notice however that we deal with a more general problem w.r.t. the cited papers, since they do not consider the possibility to specify a minimum number of assets and to declare preassigned assets. To the best of our knowledge, preassignment constraints have never been considered in previous solution techniques for this formulation of the problem, although they have been discussed informally by Chang et al. (2000).

Moving to different formulations, the first contributions on this subject appearing in the literature are due to Arnone et al. (1993), Loraschi and Tettamanzi (1996), and Loraschi et al. (1995), who present evolutionary techniques to tackle the unconstrained formulation. This formulation is considered also by Rolland (1996), who solves it using tabu search.

[1]Measured by what we call average percentage loss, introduced in Section 5.1.

Local search techniques for the constrained case has been proposed by several authors, including Gilli and Këllezi (2002). Among the population-based methods developed for tackling the constrained formulation, we mention Streichert et al. (2004), in which the cardinality constrained variant is considered, and memetic algorithm approaches introduced by Gomez et al. (2006), Kellerer and Maringer (2003), and Maringer and Winker (2003). These strategies, by being inherently effective in diversifying the search, exhibit good performance especially in multi-objective formulations, as shown by the family of Multi-Objective Evolutionary Algorithms Dioşan (2005); Ong et al. (2005); Streichert et al. (2004). Finally, Ant Colony Optimization has also been applied to portfolio problems modeled with the cardinality constraint by Armañanzas and Lozano (2005) and Maringer (2001).

4 A Hybrid Local Search Approach for Portfolio Selection

Our solver is based on a combination of local search and quadratic programming techniques. Local search acts as a master search method that uses QP as subordinate solver whose goal is to solve to optimality the subproblems assigned by the master. In this section, we first briefly introduce local search and QP, respectively, then we describe the hybrid approach.

4.1 Local Search

Local search is a family of general-purpose techniques for search and optimization problems. These techniques are *non-exhaustive* in the sense that they do not guarantee to find an optimal solution, but they explore a search space until a specific stop criterion is satisfied.

Given an instance p of a problem P, we associate a *search space* S with it. Each element $s \in S$ corresponds to a potential solution of p, and is called a *state* of p. Local search relies on a function $\mathcal{N}$ (depending on the structure of P), which assigns to each $s \in S$ its *neighborhood* $\mathcal{N}(s) \subseteq S$. Each state $s' \in \mathcal{N}(s)$ is called a *neighbor* of s.

A local search algorithm starts from an initial state s_0 (which can be obtained with some other technique or generated randomly) and enters a loop that *navigates* the search space, stepping from a state s_i to one of its neighbors s_{i+1}.

Algorithm 1 Hill climbing pseudo-code.

```
1: $s \leftarrow$ GenerateInitialSolution()
2: stop $\leftarrow$ **false**
3: **while** ($\neg$stop) **do**
4:    $s' \leftarrow$ ChooseNeighbor($s, \mathcal{N}(s)$)
5:    **if** AcceptNeighbor($s', f(s'), s, f(s)$) **then**
6:       $s \leftarrow s'$
7:    **else**
8:       stop $\leftarrow$ **true**
9:    **end if**
10: **end while**
```

The neighborhood of a state is usually described in an "intensional" fashion, i.e., in terms of atomic local changes, called *moves*, that can be applied upon it.

Several search control strategies can be defined upon this framework, according to the criteria used for the selection of the move and for stopping the search. However, in all techniques, the search is driven by a *cost function* f that estimates the quality of each state and, without loss of generality, it has to be minimized.

For constraint satisfaction problems, f generally accounts for the number of violated constraints, whereas for optimization problems it takes into account also the objective function of the problem.

The most basic local search technique is *Hill Climbing*. Hill Climbing is actually not a single local search technique, but rather a family of techniques based on the idea of performing only moves that improve (or leave unchanged, i.e. *sideways* moves) the value of the cost function f. The high-level general Hill climbing algorithm is reported in Algorithm 1. The reader who is interested in a more detailed presentation can refer to Hoos and Stützle (2005).

The way a move is selected (line 4) and the fact whether sideways moves are accepted or not (line 5) characterize the different Hill Climbing strategies. The so-called *Steepest Descent (SD)* strategy relies on the exhaustive exploration of the neighborhood and the selection of the neighbor that has the minimal value of f (breaking ties at random). The SD strategy stops as soon as no improving move is available, i.e., a local minimum has been reached (strict or not).

First descent (FD) is like SD, but interrupts the exploration of the neighborhood at the first improving move and performs it. Like SD, it stops in the first local minimum.

4.2 Quadratic Programming Solver

In order to tackle the Quadratic Program we adopt an implementation of the Goldfarb and Idnani (1983) dual-active-set method, which works for positive definite quadratic programs. The method starts from an unconstrained solution of the QP, which is both a dual feasible point and a primal optimal point for a subset of constraints of the original problem (i.e., the empty set). The algorithm iterates until primal feasibility (i.e., dual optimality) is achieved, while maintaining the primal optimality (i.e., dual feasibility) of intermediate subproblems. These subproblems are identified by an active-set, that contains the constraints that are satisfied as equalities by the current solution estimate.

Even though active-set methods are not acknowledged as the state-of-the-art methods for Quadratic Programming, yet for moderate scale problems (such as the QP at hand, and the ones coming from the hybrid local search procedure) these methods have performances at the same level of more sophisticated techniques, especially on dense matrices such as in the case at hand.

4.3 Hybrid Approach for Portfolio Selection

Our master solver is based on local search, which works on the space induced by the vector Z only. For computing the actual quantities X, it invokes the QP (slave) solver, using as the input assets for it only those such that $z_i = 1$ in the current state. A flowchart representing the control flow of the hybrid method is depicted in Figure 2.

In order to apply local search techniques we need to define the search space, the cost function, the neighborhood structures, and the selection rule for the initial solution.

4.3.1 Search Space and Cost Function

The search space S is composed of the all 2^n possible configurations of Z, with the exception of assignments that do not satisfy Constraints (5) and (7). These constraints are therefore implicitly enforced by the local search solver by excluding them from the search space. On the contrary, states that violate Constraints (1), (2), (3), or (6) are included, and these constraints are passed to the QP solver that handles them explicitly.

The QP solver receives as input only those assets included in the state under consideration, and it produces the assignment of values to the corresponding x_i variables. For all assets a_i that are not included in the state we obviously set $x_i = 0$, according

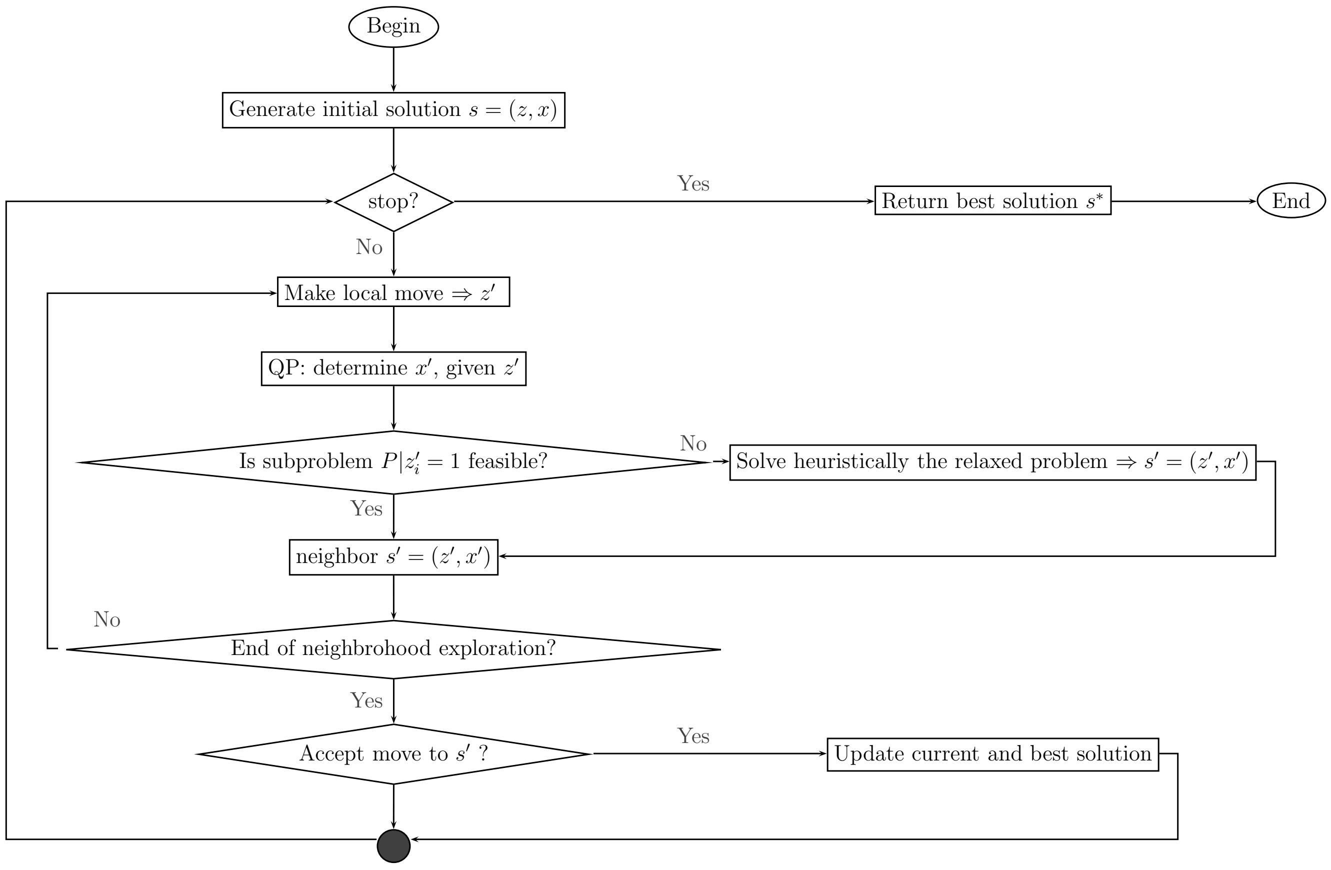

Begin
Generate initial solution $s = (z, x)$
stop?
Yes
Return best solution s^*
End
No
Make local move $\Rightarrow z'$
QP: determine x', given z'
Is subproblem $P|z'_i = 1$ feasible?
No
Solve heuristically the relaxed problem $\Rightarrow s' = (z', x')$
Yes
neighbor $s' = (z', x')$
No
End of neighbrohood exploration?
Yes
Accept move to s' ?
Yes
Update current and best solution

Figure 2: Hybrid solving method flowchart.

to Constraints (4). In addition, the QP solver also returns the computed risk F for the solution produced, which represents the cost of the state.

If the QP solver is unable to produce a feasible solution it returns the special value $F = +\infty$, and the values for x_i produced are not meaningful. In this case, we relax Constraint (1) and we build the configuration, using only the assets included in s, that gives the highest return without violating the other constraints. This construction is done by a greedy algorithm that sorts the assets by the expected return and assigns the maximum quantity to each asset in turn, as long as the sum is smaller than 1 (similarly to the fractional knapsack problem).

In the latter case, the cost is the degree of violation of Constraint (1) multiplied by a suitably large constant (that ensures that return-related costs are always bigger than risk-related ones).

The general algorithmic structure of the hybrid method is the same as the one described in Algorithm 1. The part that is changed is the neighbor evaluation, performed inside the function ChooseNeighbor() (line 4 of Algorithm 1). Its pseudo-code is reported

Algorithm 2 SD ChooseNeighbor pseudo-code.

```
1: QP denotes the Quadratic programming solver
2: SolveRelaxedP() denotes the heuristic procedure that solves the relaxed problem
3: $(F, x)$ denotes objective function value and assignment to continuous variables returned by QP() or SolveRelaxedP()
4: **Input:** current solution $s = (z, x)$
5: **Output:** best neighbor $s^* = (z^*, x^*)$
6: $F^* \leftarrow +\infty$
7: for all $z' \in \mathcal{N}(z) \wedge z'$ is feasible do
8:    $(F', x') \leftarrow$QP$(P|z_i = 1)$
9:    if $F' == +\infty$ then {infeasible subproblem}
10:       $(F', x') \leftarrow$ SolveRelaxedP$(P|z_i = 1)$
11:   end if
12:   $s' \leftarrow (z', x')$
13:   if $F' < F^*$ then {update best neighbor}
14:       $s^* \leftarrow s'$
15:       $F^* \leftarrow F'$
16:   end if
17: end for
```

in Algorithm 2 in the case of Steepest descent.

4.3.2 Neighborhood Structure

The neighborhood relation we propose is based on addition, deletion and replacement of an asset. A move m is identified by a pair $\langle i, j \rangle$, where a_i is the asset to be added and a_j is the asset to be deleted ($i, j \in \{1, \ldots, n\}$, $i \neq j$). The value of i can also be 0, meaning that no asset is added. Analogously for j, if $j = 0$ it means that no asset is deleted.

Notice that not all pairs $m = \langle i, j \rangle$, with $i, j \in \{0, 1, \ldots, n\}$, correspond to a feasible move, since some values are meaningless, e.g. inserting an asset already present or setting both i and j to zero (*null move*). In details, the move $m = \langle i, j \rangle$ is *infeasible* in a given state s if one of the following conditions holds (where $|s|$ denotes the cardinality of s: the number of h's such that $z_h = 1$ in s):

- $i = 0$ and $j = 0$ (null move)
- $i \neq 0$ and $z_i = 1$ (adding an asset already present)
- $j \neq 0$ and $z_j = 0$ (deleting an asset not present)
- $i = 0$, $j \neq 0$, and $|s| = k_{min}$ (removing an asset when at the minimum)

- $j = 0$, $i \neq 0$, and $|s| = k_{max}$ (adding an asset when at the maximum)
- $p_i = 1$ (removing a preassigned asset)

Infeasible moves do not contribute to the actual neighborhood, in the sense that they are not generated and evaluated by the local search procedure.

4.3.3 Initial Solution Construction

For the initial solution, we use three different strategies, that are employed at different stages of the search (as explained in Section 4.3.4). For all three, we ensure that Constraints (5) and (7) are always satisfied.

RandomCard: We draw at random a number k (between k_{min} and k_{max}), and we insert k randomly selected assets.

MaxReturn: We build the portfolio that produces the maximum possible return, independently of the risk (using the greedy algorithm mentioned above).

PreviousPoint: We use the final solution of the previously computed point of the efficient frontier.

4.3.4 Local Search Techniques

We implemented the two mentioned local search techniques, namely SD, and FD. In order to improve the robustness of the solver, for all metaheuristics, we make two runs for each value of R: one using the RandomCard initial solution construction, and the other using the PreviousPoint one. For the very first point of the frontier (highest requested return and no previous point available) we use instead the MaxReturn construction.

5 Experimental Analysis and Comparisons

In this section, we first present the benchmark instances and the settings of our solvers. In the following subsections, we show the comparison with all the previous works that use the same formulation or a simpler one (with less constraint types). We conclude showing a search space analysis that tries to explain the behavior of our solvers on the proposed instances.

5.1 Benchmark Instances

We experimented our techniques on two datasets obtained from real stock markets and used in previous works. The first is a group of five instances taken from the repository `ORlib` available at the URL `http://people.brunel.ac.uk/~mastjjb/jeb/info.html`. These instances have been proposed by Chang et al. (2000) and have been studied also by Schaerf (2002), Armañanzas and Lozano (2005), Moral-Escudero et al. (2006), and Fernández and Gómez (2007). According to Chang et al. (2000), these instances refer to well known stock indexes and they were extracted from *DataStream* weekly price data from March 1992 to September 1997. Only stocks for which all the price values along the time window were available has been included in the instances.

The second group of three instances have been provided to us by M. Schyns and are used in Crama and Schyns (2003). Also these instances were obtained from US stock market DataStream weekly prices from 6 January 1988 to 9 April 1997. The stocks included in the instances were drawn at random from a subpopulation of major stocks.

For the first group, a discretized UEF composed of 100 equally distributed values for the minimum required return R is provided along with the data. For the second group, we compute the discretized UEF ourselves using the QP solver with all assets available and no additional constraints. Table 1 illustrates for all instances the original market, the average variance of the UEF, and the minimum eigenvalue of the covariance matrix (which is a measure of the problem conditioning). The CEF for some settings of instance 4 has already been shown in Figure 1.

As in previous works, we evaluate the quality of our solutions employing an aggregate indicator that measures the deviation of the CEF found by the algorithms w.r.t. the UEF on the whole set of frontier points. We call this measure *average percentage loss* (apl) and we define it as follows: let R_l be the minimum required return, $V(R_l)$ and $V_U(R_l)$ the values of the function F returned by the solver and the risk on the UEF, respectively, and $l = 1, \ldots, p$ where p is the number of points of the frontier; the apl is equal to $\frac{100}{p} \sum_{l=1}^{p} (V(R_l) - V_U(R_l))/V_U(R_l)$.

5.2 Experimental Setting of the Solvers

Experiments were performed on a Pentium 4 (3.2 GHz) processor running Linux; the SD, and FD metaheuristics have been coded in C++ exploiting the framework EASY-LOCAL++ Di Gaspero and Schaerf (2003), the QP solver has also been coded in C++

Table 1: The benchmark instances.

`ORlib` dataset

Inst.	Origin	assets	UEF	min λ_σ
1	Hong Kong (Hang Seng)	31	$1.55936{\cdot}10^{-3}$	$2.2648{\cdot}10^{-4}$
2	Germany (DAX 100)	85	$0.412213{\cdot}10^{-3}$	$8.1830{\cdot}10^{-5}$
3	UK (FTSE 100)	89	$0.454259{\cdot}10^{-3}$	$5.9073{\cdot}10^{-5}$
4	USA (S&P 100)	98	$0.502038{\cdot}10^{-3}$	$8.0857{\cdot}10^{-5}$
5	Japan (NIKKEI)	225	$0.458285{\cdot}10^{-3}$	$6.0542{\cdot}10^{-6}$

Crama and Schyns's dataset

Inst.	Origin	assets	UEF	min λ_σ
S1	USA (DataStream)	20	4.812528	5.7846
S2	USA (DataStream)	30	8.892189	4.3539
S3	USA (DataStream)	151	8.64933	0.83626

and is made publicly available from one of the authors' website[2]. The executables were obtained using the GNU C/C++ compiler (v. 4.0.1).

Concerning the algorithms setting, SD and FD have no parameter to be set.

5.3 Comparison with Previous Results

Portfolio optimization is nowadays amongst the most studied topics in finance, and a wide range of models have been proposed to take care of the problem: several variable definition, objective functions, constraint set, benchmarks and heuristic techniques have been proposed. For this reason, a fair comparison amongst papers cannot be performed, and, due to different formulations employed by the authors, the only papers we can compare with are those of Schaerf (2002) and Moral-Escudero et al. (2006), who employ the same set of constraints on the `ORlib` instances, and with Crama and Schyns (2003) who deal with a slightly different setting and with a novel set of instances.

Concerning Chang et al. (2000), as already pointed out in Schaerf (2002), even though they work on the `ORlib` instances (and with the same constraints), a fair comparison with their solutions is not possible because the problem is solved by taking points along the frontier that are not homogeneously distributed. For exactly the same reason, it is not possible to compare with Fernández and Gómez (2007).

Armañanzas and Lozano (2005) work on a variant of the problem for which the values k_{min} and k_{max} coincide (i.e., $k_{min} = k_{max} = K$) on the `ORlib` instances. However, due to

[2]`http://www.diegm.uniud.it/digaspero/`

Table 2: Comparison of results with (1) Schaerf and (2) Moral-Escudero et al.

	FD+QP		SD+QP		TS$^{(1)}$		GA+QP$^{(2)}$	
Inst.	min apl	time	min apl	time	min apl	time	min apl	time
1	0.00366	1.5s	0.00321	3.1s	0.00409	251s	0.00321	415.1s
2	2.66104	9.6s	2.53139	14.1s	2.53617	531s	2.53180	552.7s
3	2.00146	10.1s	1.92146	16.1s	1.92597	583s	1.92150	886.3s
4	4.77157	11.2s	4.69371	18.8s	4.69816	713s	4.69507	1163.7s
5	0.24176	25.3s	0.20219	45.9s	0.20258	1603s	0.20198	1465.8s

what we believe is an error in the implementation of their solution methods[3] they obtain a set of points that are infeasible w.r.t. Constraint (2). In details, they assign to the assets i for which $z_i = 1$ chosen by their ACO algorithm the quantity $x_i = (\delta_i - \epsilon_i)/K$, therefore since they set $\epsilon_i = 0.001, \delta_i = 1$ for all $i = 1, \ldots, n$, they obtain $\sum_{i=1}^{n} x_i = 0.999$ instead of 1. For this reason we could not compare our solvers with Armañanzas and Lozano (2005).

Comparison with Schaerf and Moral-Escudero et al. Schaerf (2002) also uses local search, but as a *monolithic* solver exploring a search space composed of both continuous and discrete variables; whereas in our approach the local search focuses on the discrete variables. Moral-Escudero et al. (2006), instead, use like us a hybrid solver, although they make use of genetic algorithms instead of local search for the determination of the discrete variables.

For this comparison, we set the constraint values exactly as in the previous papers: $\epsilon_i = 0.01$ and $\delta_i = 1$ for $i = 1, \ldots, n$, and $k_{max} = 10$ for all instances. Minimum cardinality is not considered in the cited works, and therefore we set $k_{min} = 1$ (i.e., no limitation). Preassignments are not considered too, and thus we set $p_i = 0$ (for all i) in our solver.

Table 2 shows best results and running times obtained by our three solvers in comparison with previous work. Since Moral-Escudero et al. (2006) report only the best outcomes of their solvers, in order to fairly compare with them we have to present the results as the *minimum* apl w.r.t. the UEF found by the algorithm (without any statistical indicator). However, for SD+QP the standard deviation is close to zero and all runs obtain approximately the same results.

[3]We found the error in our analysis of the data kindly provided to us by J. Lozano.

Table 3: Comparison of results with (3) Crama and Schyns.

	FD+QP		SD+QP		SA$^{(3)}$	
Inst.	apl	time	apl	time	apl	time
S1	0.72 (0.094)	0.3s	0.35 (0.0)	1.4s	1.13 (0.13)	3.2s
S2	1.79 (0.22)	0.5s	1.48 (0.0)	3.1s	3.46 (0.17)	5.4s
S3	10.50 (0.51)	10.2s	8.87 (0.003)	124.3s	16.12 (0.43)	30.1s

The results of our solvers are the best CEFs found in 30 trials of the algorithm on each instance and the running times reported are those of the best trial, exactly as Moral-Escudero et al. (2006). Running times of Schaerf (2002) are obtained re-running Schaerf's software on our machine, those of Moral-Escudero et al. are taken from their paper, and are obtained using a PC having about the same performances.

Table 2 shows that we obtain results superior to Schaerf (2002) both in terms of risk and running times. This suggests that the hybrid solver outperforms monolithic local search ones.

Regarding Moral-Escudero et al. (2006), we obtain with SD+QP results very similar to their best solver. We believe that the results are the very same, and the differences are due only to different roundings of the real values. It is worth noticing however that we obtain them in a much shorter time.

Before drawing definitive conclusions about possible differences in performance w.r.t. Moral-Escudero et al. (2006) we would need other (more difficult) instances and a more principled statistical comparison, based on the whole distributions of quality of solution returned and execution time.

Comparison with Crama and Schyns. Since the results of Crama and Schyns (2003) are presented in the paper in graphical form and make use of a slightly different cost function (i.e., they consider the standard deviation instead of the variance as the risk measure) we re-run their solver[4] on the three instances employed in their experimentation employing the same parameter setting reported in their paper. The constraints set in this experiment are as follows: $k_{min} = 1$, $k_{max} = 10$, $\epsilon_i = 0$, and $\delta_i = 0.25$.

In Table 3 we present the outcome of this comparison. For each algorithm we report in three columns the average and the standard deviation (in parentheses) of the apl w.r.t.

[4]The executable was kindly provided to us by M. Schyns.

the UEF, and the average time spent by the algorithm. The data was collected by running 30 times each algorithm on each instance and computing the whole CEF.

From the table it is clear that, in terms of solution quality, the family of our solvers outperforms the SA approach of Crama and Schyns. Looking at the times, we can see that SA, in general, exhibits shorter running times than our hybrid SD+QP approach. This can be explained by the strategy employed by both our algorithms that thoroughly explore the full neighborhood of each solution whereas the SA randomly picks out only some neighbors, thus saving time in the evaluation of the cost function. Moreover, the slave QP procedure is more time-consuming than the solution evaluation carried out by Crama and Schyns, however it allows us a higher accuracy on the assignment of the assets.

5.4 Efficient frontiers and Portfolio Composition

In Figure 3 we plot the efficient frontiers found by our SD+QP solver on the constrained problem for different values of k_{max} (letting $\epsilon_i = 0.01$ and $\delta_i = 1$ as in previous experiments). In particular we draw the solutions for $k_{max} = 2, 4, 8, 16$ and 32 on the ORLib Instances 2–5.

It is possible to observe that for the instances at hand, the constrained frontier is quite smooth and very close to the unconstrained one apart for the case $k_{max} = 2$, which also features a more rough frontier plot. This can be explained by the fact that in this setting the choice of the two assets to be included in the solution is quite crucial and it seems to make the local search space quite rugged, prompting for more sophisticated solution techniques.

Looking at the solutions from another point of view, in Figure 4 we illustrate the composition of the portfolio obtained under various settings on a representative instance (namely ORLib instance 4). For each return value we plot the histogram of the shares invested in each asset (identified by a different gray level) in the non-dominated solution, i.e., each bar of the histogram shows the percentage of the shares composing the portfolio. On the x-axis on the top of each plot, it is also reported the corresponding portfolio cardinality, i.e., the number of non-zero shares. The settings of the experiment are reported below each plot.

Figure 4(a) illustrates the portfolio composition in the case of the unconstrained problem. The general trend that can be observed is portfolio fragmentation decreases

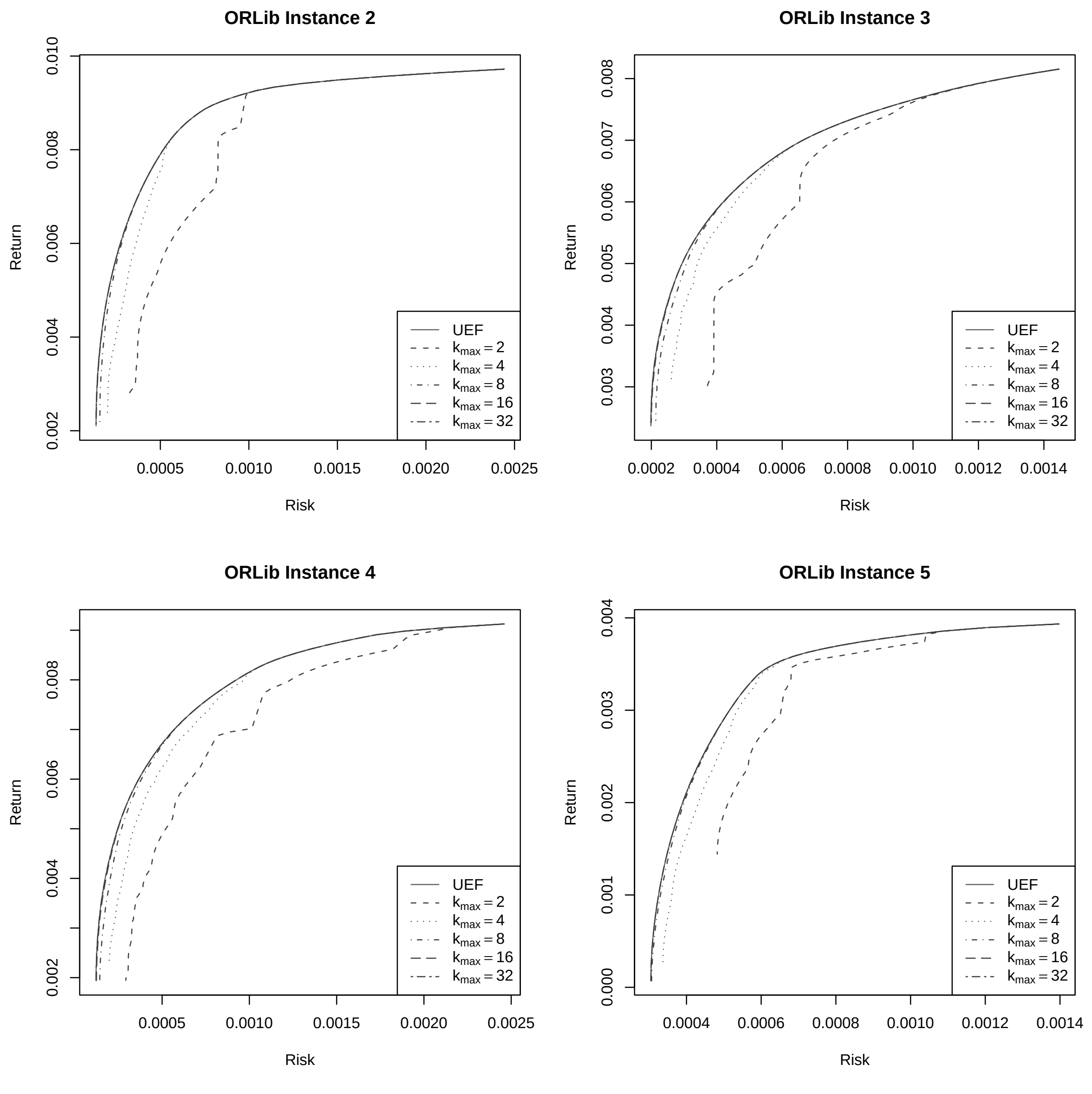
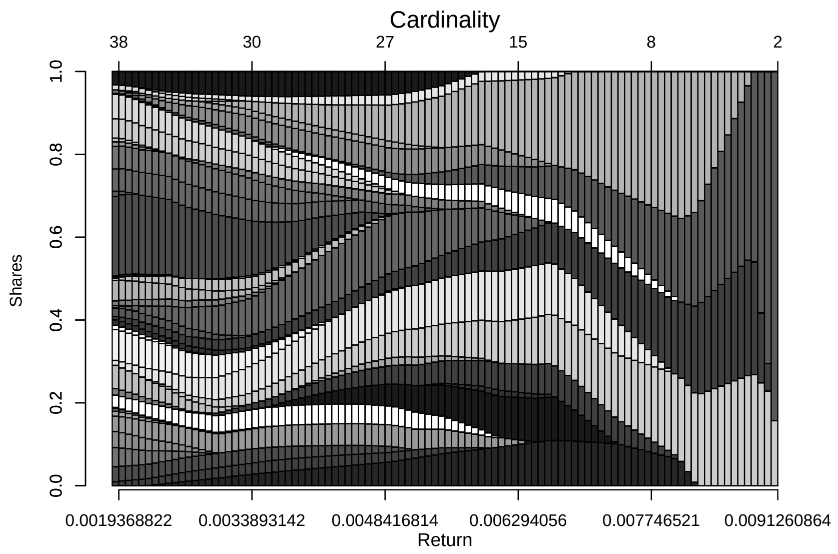
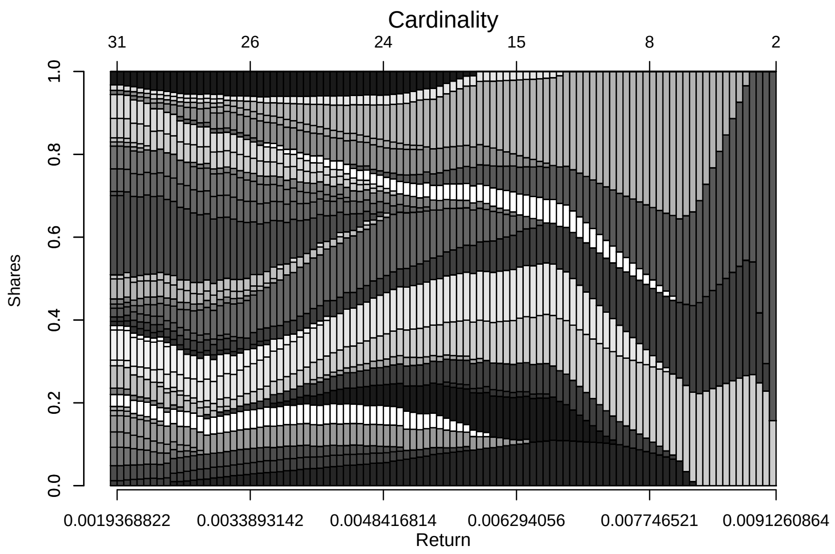


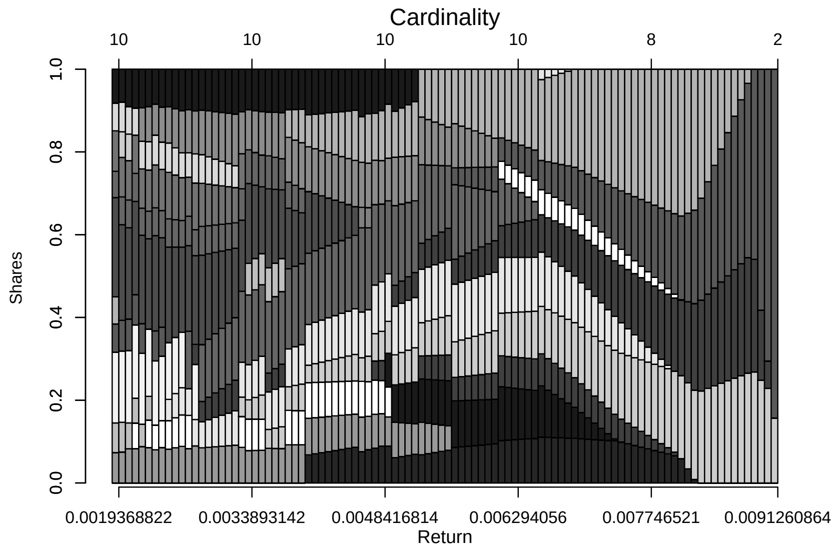
Figure 3: Unconstrained and Constrained Efficient Frontiers (for different values of k_{max}).



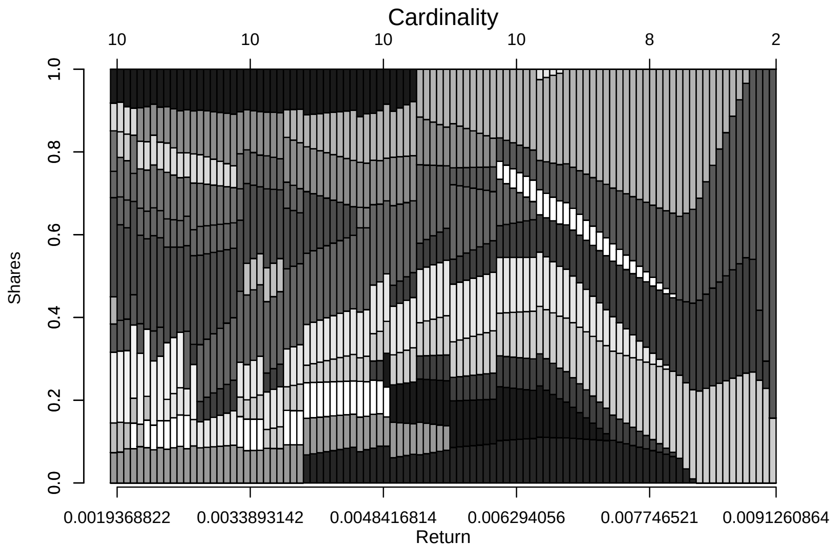
(a) UEF: $k_{min} = 1$, $k_{max} = +\infty$, $\varepsilon_i = 0$, $\delta_i = 1$



(b) $k_{min} = 1$, $k_{max} = +\infty$, $\varepsilon_i = 0.01$, $\delta_i = 1$



(c) $k_{min} = 1$, $k_{max} = 10$, $\varepsilon_i = 0$, $\delta_i = 1$



(d)$k_{min} = 1$, $k_{max} = 10$, $\varepsilon_i = 0.01$, $\delta_i = 1$

Figure 4: Analysis of the portfolio composition for ORLib Instance 4.

from low to high return values. This result reflects the fact that when the overall return to be guaranteed is high, there are few asset combinations among which the financial operator can choose. On the other extreme, the combinations of assets providing a small overall return are abundant, thus risk is minimized by increasing diversification.

Figure 4(b) illustrates the case in which only constraints on minimum and maximum quantity are imposed. The difference between the unconstrained case is subtle, yet remarkable. First of all, contours are less smooth because assets are included only at a minimum quantity, thus when a new asset enters the portfolio, this corresponds to a step in the related contour. The second observation is that the upper bound on quantity has a negligible effect also for the highest return values, because risk minimization usually requires anyway the inclusion of more than one asset in the portfolio.

The effect of constraining the maximal cardinality k_{max} is illustrated in Figure 4(c). As expected, an upper bound on the maximum asset cardinality causes a sharp segmentation of the portfolios, making also apparent that the combinatoric part of the problem becomes more relevant. Finally, Figure 4(d) shows the combined effect of quantity and maximum cardinality constraints. The difference with the previous case is rather limited, because the cardinality constraint dominates the feasibility component of the problem.

5.5 Results with preassignment constraints

In order to evaluate the impact of preassigment constraints on solution quality, for each benchmark instance we proceed as follows: we fix in turn one of the assets as preassigned and we compute the resulting CEF (imposing the other constraints as in the previous experiments, i.e., $\epsilon_i = 0.01$ and $\delta_i = 1$ for $i = 1, \ldots, n$, no cardinality constraints is set). This way, for each instance consisting of n assets, we obtain a family of n CEFs, one for each preassigned asset.

Intuitively, imposing preassignment constraints generally worsen the solution quality measured with apl, unless the preassigned assets belong to an optimal solution for all the values of the required return R. Moreover, the magnitude of this worsening effect might depend on the asset features. Specifically, when a low-return asset is preassigned the performances should deteriorate significantly, whereas when a high return one is preassigned the deterioration is lower.

As an example, in Figure 5 we report the results obtained on the instance S1 by imposing preassignment constraints on the two assets for which the return values reach

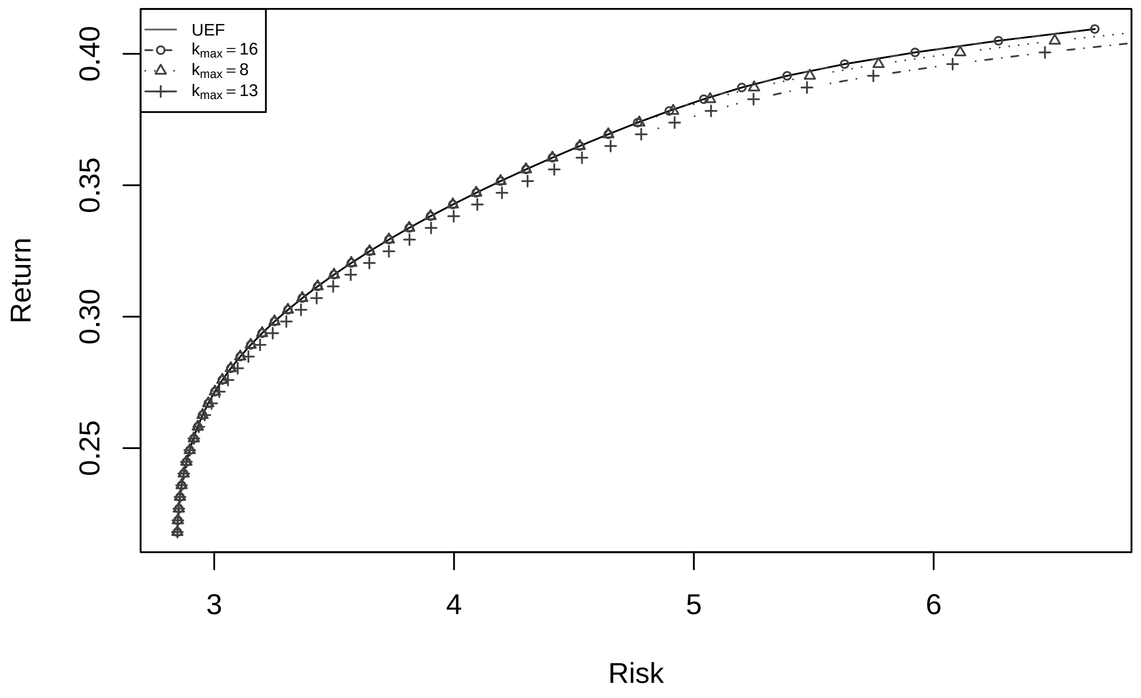
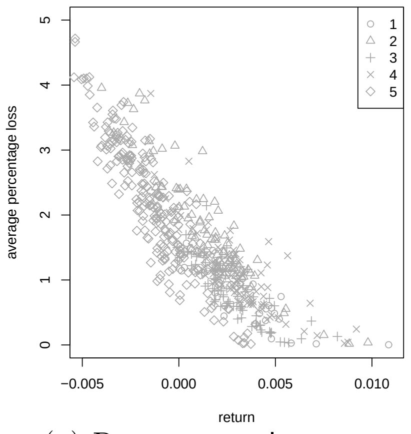


Figure 5: UEF and preassignment constrained CEFs for extremal return values.

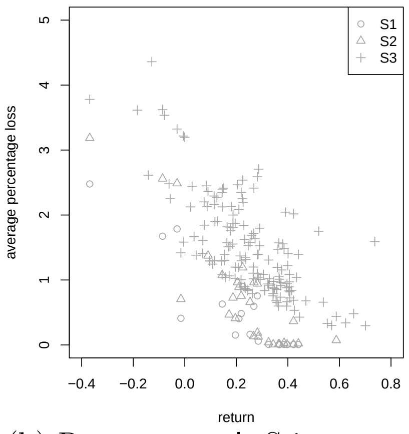
the extremal values and on an asset whose return is the closest to 0 (i.e., $r_{13} = \min\{r_i\} = -0.368822$ and $r_{16} = \max\{r_i\} = 0.440583$ and $r_8 = -0.0150255$). The picture shows that, as intuitively expected, the CEF obtained preassigning asset a_{16} strictly dominates the one obtained preassigning a_{13}. The CEF for a_8 preassigned lies between the other two.

The general behavior of our SD+QP solver for these experiments is reported in Figure 6. For each asset a_i we plot its return r_i (a–b) or the variance σ_{ii} (c–d) on the x-axis and on the y-axis the apl obtained by preassigning a_i. Notice that the x-scale is different in the two graphs for the `ORlib` and the Crama and Schyns's instances, because of the different nature of the two benchmarks and the different way to compute assets statistics.

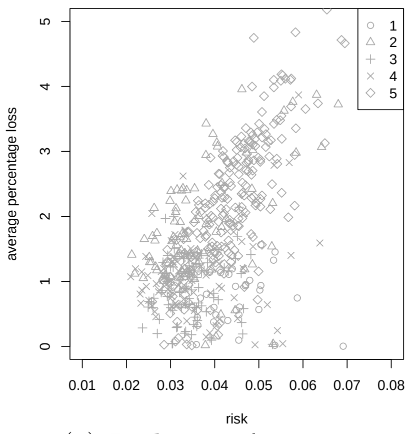
Figures 6(a–b) confirm our claim about the relationship between return of the preassigned asset and solution quality. They show that there is a good inverse correlation between apl and r_i, which is confirmed by the numerical values of Pearson's correlation reported in Table 4. Conversely, apart for a few instances, there is no perceivable trend



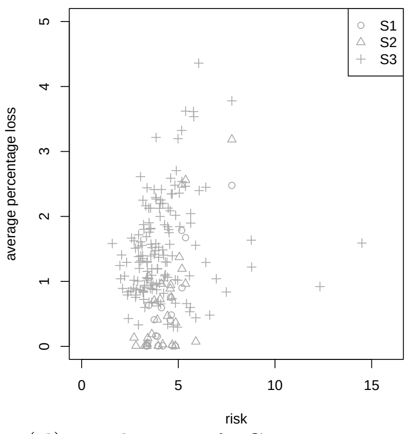
(a) Return vs apl: `ORlib` instances



(b) Return vs apl: S instances



(c) Risk vs apl: `ORlib` instances



(d) Risk vs apl: S instances

Figure 6: Return/risk vs apl for one preassigned asset.

Table 4: Pearson's correlation between preassigned asset return/risk and the resulting apl.

Inst.	$\rho(\mathsf{apl}, r_i)$	$\rho(\mathsf{apl}, \sigma_{ii})$
1	-0.7907505	0.06618641
2	-0.9107568	0.4910591
3	-0.8249417	0.09394818
4	-0.8138779	0.2904020
5	-0.9280059	0.8029789
S1	-0.896169	0.825982
S2	-0.8816018	0.6919496
S3	-0.73096	0.1938812

between the variance of the preassigned asset σ_{ii} and apl as it can be seen from Figures 6(c–d) and from the values in the correlation table. This depends on the situation whether in the specific instance there are suitable assets that have a negative covariance w.r.t. the preassigned one and also have sufficiently high return.

5.6 Comparison with a family of exact solvers

For the purpose of evaluating the effectiveness of our techniques w.r.t. the state-of-the-art Mixed Integer Non-linear Programming (MINLP) solvers we encode in AMPL Fourer et al. (2002) the problem formulation. The AMPL model is then solved using CPLEX 11.0.1 and MOSEK 5. We experimented the MINLP solvers with two different settings: (i) *cold restarts*, i.e. the MINLP solver is started from an empty initial solution, and (ii) *warm restarts*, that is the solution computed for the previous return point is provided as initial solution. Not surprisingly warm restarts performed slightly better and in the following we report only the results of this setting.

We run the MINLP solvers on all instances of the ORLib benchmark, with the same constraint setting employed in Section 5.3. For each return point of the UEF we compute the optimal risk value with the exact solver and we record the running time. For the SD+QP solver we repeat the same measures for 10 runs on each point and we take the average as the performance indicator.

For all instances, the optimal risk values computed by the MINLP solvers are exactly the same as those found by our solver (apart the rounding errors for reals), thus proving that our results are indeed optimal.

Concerning the running times, instead, there are noticeable differences between our

Table 5: Running times for the computation of the whole CEF on ORLib instances 1–4.

Instance	avg(SD+QP)	CPLEX 11	MOSEK 5
1	3.1s	2.1s	15.8s
2	14.7s	397.1s	5.0s
3	18.0s	890.7s	1,903.3s
4	20.9s	169,461.0s	239,178.4s

hybrid approach and the performances of the MINLP solvers. In Figure 7 we report the running times of CPLEX, MOSEK and the SD+QP algorithm against the return value for the first 4 instances of the ORLib benchmark. The scale of the graphs is semi-logarithmic (on the y-axis) in orderd to enhance also small differences of performances.

From the results we notice that the running times of the three solvers are comparable only for the points for which Constraints (5) are not binding (with the clear advantage of the proof of optimality for the MINLP solvers). However, for the points in which Constraints (5) are binding (the left part of the UEF, where the required return value is low) the situation changes dramatically and both CPLEX and MOSEK running times scale very poorly with instance size. As an example, CPLEX employs more than 2 hours for the computation of the solution value for one single return point on Instance 4. The performances of our SD+QP solver, instead, are very stable and they scale well with respect to both instance size and constraints strength.

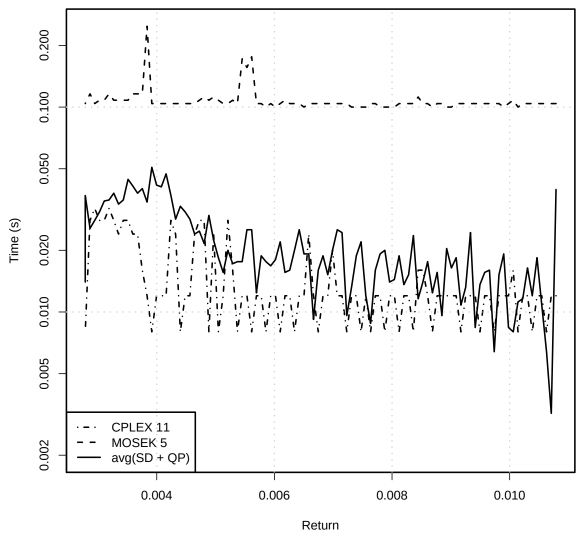
The total running times for the computation of the whole CEF are shown in Table 5; for SD+QP we report the average of the total running times for the 10 runs. The table cofirms that SD+QP outperforms CPLEX and MOSEK in the computation of the whole frontier on all instances except of Instance 1, which is the smaller one.

In conclusions, solving the problem with current state-of-the-art MINLP solvers is to be preferred for the easier cases for which they are able to reach the certified optimal solution, but it is impractical for the bigger and more constrained ones.

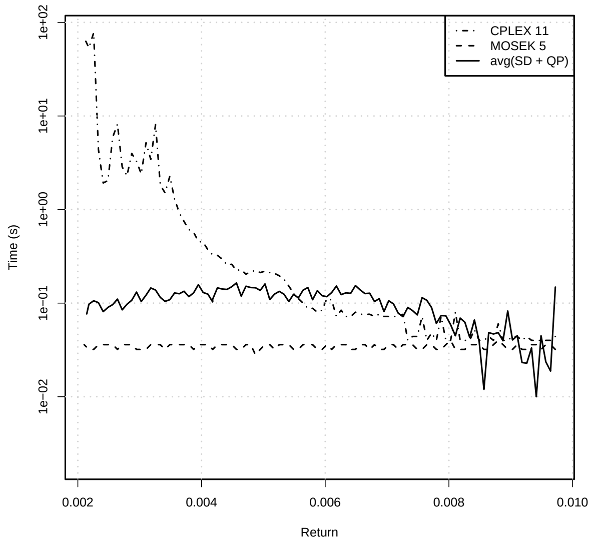
A better and *ad hoc* tuning of the MINLP solvers might help improve their running time, however this is out of the scope of the present work.

6 Conclusions and Future Work

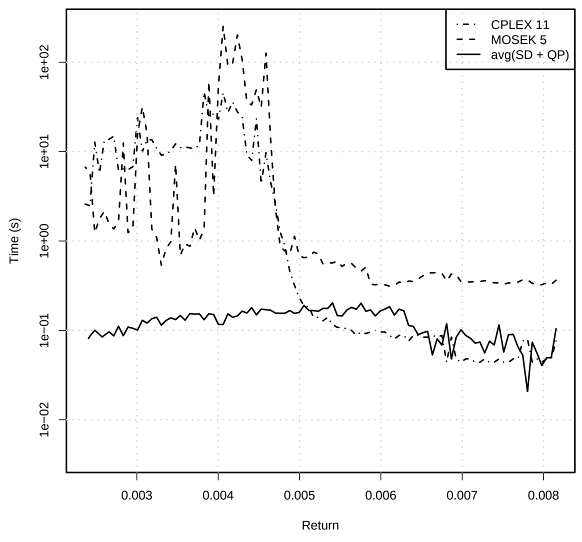
In this work, we have extended the constrained Markowitz model for the portfolio selection problem by adding minimum cardinality constraints and preassignments. We have



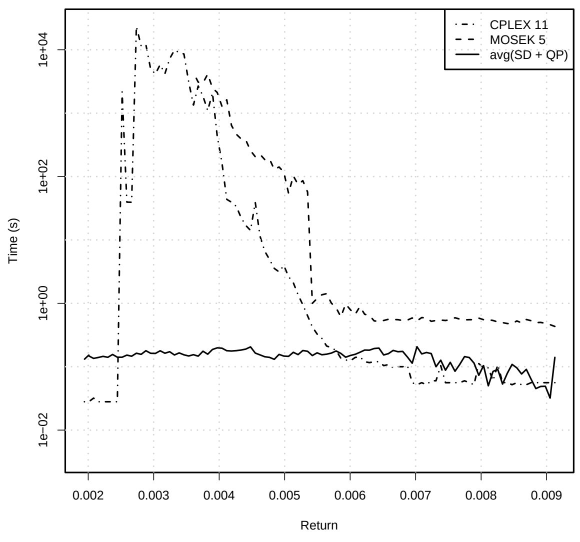
(a) ORLib Instance 1



(b) ORLib Instance 2



(c) ORLib Instance 3



(d) ORLib Instance 4

Figure 7: Comparison of the SD+QP solver against a MINLP-encoding of the problem solved by CPLEX and MOSEK solvers.

compared our results with those in the literature for the less constrained problem formulation (no minimum cardinality, no preassigned). Finally, we have proposed a detailed analysis of the search space of the benchmark instances and the behaviour of metaheuristic solvers for this problem.

The experimental analysis has highlighted that, in our hybrid setting, basic metaheuristic techniques are adequate to obtain good performances for the available instances. In fact, the experiments show that our solver finds the optimal solution and in thus is comparable with (or superior to) the state of the art.

In the future, we plan to adapt this approach to tackle other formulations, such as the fully-discrete formulation that is particularly interesting for some investors. This formulation enables us to take into account aspects of real-world finance, such as transaction lots. To this extent, instances including minimum lots will be investigated, since assets generally cannot be purchased in any quantity and the amount of money to be invested in a single asset must be a multiple of a given minimum lot Mansini and Speranza (1999).

We also aim at identifying difficult real instances and verify whether more sophisticated local search metaheuristics, such as TS, could improve on the results of the simple SD strategy.

We are furthermore aimed in exploring and verify economic phenomena that arise from different settings of the parameters (diversification in small portfolios, investor preferences, etc.) and comparing several risk measures on the same instances, to show analogies and differences in portfolios obtained optimizing different objective functions.

Acknowledgements

We thank Jose Lozano, Renata Mansini, Michaël Schyns, and Rubén Ruiz-Torrubiano for helpful clarification about their work. We are also grateful to Franca Rinaldi for her assistance for the use of CPLEX.

References

Armañanzas, R. and Lozano, J. (2005). A multiobjective approach to the portfolio optimization problem. In *Proceedings of the 2005 IEEE Congress on Evolutionary Computation (CEC 2005)*, volume 2, pages 1388–1395. IEEE Press.

Arnone, S., Loraschi, A., and Tettamanzi, A. (1993). A genetic approach to portfolio selection. *Neural Network World – International Journal on Neural and Mass-Parallel Computing and Information Systems*, 3(6):597–604.

Bienstock, D. (1996). Computational study of a family of mixed-integer quadratic programming problems. *Mathematical Programming*, 74:121–140.

Blume, M. and Friend, I. (1975). The asset structure of individual portfolios and some implications for utility functions. *The Journal of Finance*, 30(2):585–603.

Chang, T.-J., Meade, N., Beasley, J. E., and Sharaiha, Y. M. (2000). Heuristics for cardinality constrained portfolio optimisation. *Computers & Operations Research*, 27(13):1271–1302.

Crama, Y. and Schyns, M. (2003). Simulated annealing for complex portfolio selection problems. *European Journal of Operational Research*, 150:546–571.

Di Gaspero, L. and Schaerf, A. (2003). EasyLocal++: An object-oriented framework for flexible design of local search algorithms. *Software—Practice & Experience*, 33(8):733–765.

di Tollo, G. and Roli, A. (2006). Metaheuristics for the portfolio selection problem. Technical Report R-2006-005, Dipartimento di Scienze, Università "G. D'Annunzio" Chieti–Pescara. Accepted for publication on *International Journal of Operations Research*. Available from `http://www.sci.unich.it/~tecrep/2006/R-2006-005.pdf`.

Dioşan, L. (2005). A multi-objective evolutionary approach to the portfolio optimization problem. In *Proceedings of the International Conference on Computational Intelligence for Modelling, Control and Automation (CIMCA 2005)*, pages 183–188. IEEE Press.

Fernández, A. and Gómez, S. (2007). Portfolio selection using neural networks. *Computers & Operations Research*, 34:1177–1191.

Fourer, R., Gay, D. M., and Kernighan, B. W. (2002). *AMPL: A Modeling Language for Mathematical Programming*. Duxbury Press/Brooks/Cole Publishing Company.

Gilli, M. and Këllezi, E. (2002). A global optimization heuristic for portfolio choice with VaR and expected shortfall. In Kontoghiorghes, E. J., Rustem, B., and Siokos, S., editors, *Computational Methods in Decision-making, Economics and Finance*, Applied Optimization Series, pages 167–183. Kluwer Academic Publishers.

Goldfarb, D. and Idnani, A. (1983). A numerically stable dual method for solving strictly convex quadratic programs. *Mathematical Programming*, 27:1–33.

Gomez, M., Flores, C., and Osorio, M. (2006). Hybrid search for cardinality constrained portfolio optimization. In *Proceedings of the 8th annual conference on Genetic and Evolutionary Computation,*, pages 1865–1866. ACM Press.

Guiso, L., Jappelli, T., and Terlizzese, D. (1996). Income risk, borrowing constraints and portfolio choice. *American Economic Review*, 86(1):158–172.

Hoos, H. and Stützle, T. (2005). *Stochastic Local Search Foundations and Applications.* Morgan Kaufmann Publishers, San Francisco, CA (USA).

Jansen, R. and van Dijik, R. (2002). Optimal benchmark tracking with small portfolios. *The Journal of Portfolio Management*, 28(2):9–22.

Jobst, N., Horniman, M., Lucas, C., and Mitra, G. (2001). Computational aspects of alternative portfolio selection models in the presence of discrete asset choice constraints. *Quantitative Finance*, 1:1–13.

Kellerer, H. and Maringer, D. (2003). Optimization of cardinality constrained portfolios with a hybrid local search algorithm. *OR Spectrum*, 25(4):481–495.

Loraschi, A. and Tettamanzi, A. (1996). An evolutionary algorithm for portfolio selection within a downside risk framework. In Dunis, C., editor, *Forecasting Financial Markets*, Series in Financial Economics and Quantitative Analysis, pages 275–285. John Wiley & Sons, Chichester, UK.

Loraschi, A., Tettamanzi, A., Tomassini, M., and Verda, P. (1995). Distributed genetic algorithms with an application to portfolio selection problems. In Pearson, D. W., Steele, N. C., and Albrecht, R. F., editors, *Proceedings of the International Conference on Artificial Neural Networks and Genetic Algorithms*, pages 384–387. Springer-Verlag.

Mansini, R., Ogryczak, W., and Speranza, M. (2003). LP solvable models for portfolio optimization a classification and computational comparison. *IMA Journal of Management Mathematics*, 14:187–220.

Mansini, R. and Speranza, M. (1999). Heuristic algorithms for the portfolio selection problem with minimum transaction lots. *European Journal of Operational Research*, 114:219–233.

Mansini, R. and Speranza, M. (2005). An exact approach for portfolio selection with transaction costs and rounds. *IIE Transactions*, 37:919–929.

Maringer, D. (2001). Optimizing portfolios with ant systems. In *Proceedings of the International ICSC congress on computational intelligence: methods and applications (CIMA 2001)*, pages 288–294. ISCS Academic Press.

Maringer, D. (2005). *Portfolio Management with heuristic optimization*. Springer.

Maringer, D. and Winker, P. (2003). Portfolio optimization under different risk constraints with modified memetic algorithms. Technical Report 2003–005E, University of Erfurt, Faculty of Economics, Law and Social Sciences.

Markowitz, H. (1952). Portfolio selection. *Journal of Finance*, 7(1):77–91.

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

Ong, C., Huang, J., and Tzeng, G. (2005). A novel hybrid model for portfolio selection. *Applied Mathematics and Computation*, 169:1195–1210.

Rolland, E. (1996). A tabu search method for constrained real-number search: applications to portfolio selection. Technical report, The A. Gary Anderson Graduate School of Management, University of California, Riverside, CA, USA.

Schaerf, A. (2002). Local search techniques for constrained portfolio selection problems. *Computational Economics*, 20(3):177–190.

Streichert, F., Ulmer, H., and Zell, A. (2004). Comparing discrete and continuous genotypes on the constrained portfolio selection problem. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2004)*, volume 3103 of *Lecture Notes in Computer Science*, pages 1239–1250, Seattle, Washington, USA. Springer-Verlag.