Local Search Algorithms for Portfolio Selection: Search Space and Correlation Analysis

Giacomo di Tollo and Andrea Roli

Abstract Modern Portfolio Theory dates back from the fifties, and quantitative approaches to solve optimization problems stemming from this field have been proposed ever since. We propose a metaheuristic approach for the Portfolio Selection Problem that combines local search and Quadratic Programming, and we compare our approach with an exact solver. Search space and correlation analysis are performed to analyse the algorithm's performance, showing that metaheuristics can be efficiently used to determine optimal portfolio allocation.

1 Introduction

Modern Portfolio Theory dates back to the 1950s and concerns wealth allocation over assets: the investor has to decide which asset to invest in and by how much. Many optimization problem have been formulated to express this principle, and the main example is to minimize a risk measure for a given minimum required target return. Variance of portfolio's return was used as risk measure in the seminal work by Markowitz [25] and is still the most used, even though there exists a wide literature about risk measures to be implemented.

Portfolio Selection Problem (PSP) can be viewed as an optimisation problem, defined in terms of three objects: variables, objective, and constraints. Every object has to be instantiated by a choice in a set of possible choices, the combination of which induces a specific formulation (model) of the problem, and different optimisation results. For instance, as observed by di Tollo and Roli [8], two main choices are

G. di Tollo (✉)
Dipartimento di Economia, Universitá Ca' Foscari, Cannaregio 873,
30121 Venice, Italy
e-mail: giacomo.ditollo@unive.it

A. Roli
Dipartimento di Informatica e Ingegneria,
Alma Mater Studiorum – Universitá di Bologna,
Campus of Cesena, Via Venezia 52, 47521 Cesena, Italy
e-mail: andrea.roli@unibo.it

© Springer International Publishing Switzerland 2016

S. Fidanova (ed.), *Recent Advances in Computational Optimization*,
Studies in Computational Intelligence 655, DOI 10.1007/978-3-319-40132-4_2

possible for variable domains: continuous [15, 28, 29, 31] and integer [22, 30]. Choosing continuous variables is a very 'natural' option and leads to a representation independent of the actual budget, while integer values (ranging between zero and the maximum available budget, or equal to the number of 'rounds') makes it possible to add constraints taking into account actual budget, minimum lots and to tackle other objective functions to capture specific features of the problem at hand. As for the different results, the integer formulation is more suitable to explain the behaviour of rational operators such small investors, whose activity is strongly influenced by integer constraint [23].

In addition, the same representation can be modelled by means of different formulations, e.g., by adding auxiliary variables [21], symmetry breaking [27] or redundant [32] constraints, which may provide beneficial effects on, or on the contrary harm, the efficiency of the search algorithms yet preserving the possibility of finding an optimal solution.

In this work we investigate how the use of different formulations for the very same problem can lead to different behaviours of the algorithm used. We address this question by solving the PSP by means of metaheuristic techniques [4, 8], which are general problem-solving strategies conceived as high level strategies that coordinate the behaviour of lower level heuristics. Although most metaheuristics can not return a proof of optimality of the solution found, they represent a good compromise between solution quality and computational effort. Through the use of metaheuristic, and using the paradigm of separation between model and algorithm [17], we show that different formulations affect algorithm performance and we study the reasons of this phenomenon.

The paper will start by recalling Portfolio Theory in Sect. 2, before introducing the concept of metaheuristics in Sect. 3. Then we will introduce a metaheuristic approach for the Portfolio Selection Problem in Sect. 4. In Sect. 5 will briefly present the principles of the search space analysis we perform. Search Space Analysis is applied to instances of PSP and results are discussed in Sect. 6. Finally, we conclude in Sect. 7.

2 Portfolio Selection Basis

We associate to each asset belonging to a set A of n *assets* ($A = \{a_1, \ldots, a_n\}$) a real-valued *expected return* r_i, and the corresponding return variance σ_i. We furthermore associate, to each pair of assets $\langle a_i, a_j \rangle$, a real-valued return *covariance* σ_{ij}. We are furthermore given a value r_e representing the minimum required return.

In this context, a portfolio is defined as the n-sized real vector $X = \{x_1, \ldots, x_n\}$ in which x_i represents the relative amount invested in asset a_i. For each portfolio we can define its variance as $\sum_{i=1}^{n} \sum_{j=1}^{n} \sigma_{ij} x_i x_j$ and its return as $\sum_{i=1}^{n} r_i x_i$. In the original formulation [25], PSP is formulated as the minimization of portfolio variance, imposing that the portfolio's return must be not smaller than r_e, leading to the following optimisation problem:

$$\min \sum_{i=1}^{n} \sum_{j=1}^{n} \sigma_{ij} x_i x_j, \tag{1}$$

$$s.t. \sum_{i=1}^{n} r_i x_i \geq r_e, \tag{2}$$

$$\sum_{i=1}^{n} x_i = 1, \tag{3}$$

$$x_i \geq 0 \qquad (i = 1, \ldots, n). \tag{4}$$

The aforecited return constrained is introduced in constraint (2); constraint (3) is referred to as *budget constraint*, meaning that all the capital must be invested; constraint (4) imposes that variables have to be non-negative (i.e., short sales are not allowed).

If we define a finite set of values for r_e and solve the problem for all defined r_e values, we obtain the *Unconstrained Efficient Frontier* (UEF), in which the minimum risk value is associated to each r_e.

This formulation may be improved to grasp financial market features, by introducing a binary variable Z for each asset ($z_i = 1$ if asset i is on the portfolio, 0 otherwise). Additional constraints which can be added to the basic formulation are:

- **Cardinality constraint**, used either to impose an upper bound k to the cardinality of assets in the portfolio

$$\sum_{i=1}^{n} z_i \leq k, \tag{5}$$

  or to force the resulting portfolio to contain exactly k assets:

$$\sum_{i=1}^{n} z_i = k_{max}. \tag{6}$$

  This constraint is important for practitioners in order to reduce the portfolio management costs.
- **Floor and ceiling constraints**, used to set, for each asset, the minimum (ε_i) and maximum (δ_i) quantity allowed to be held in the portfolio

$$\varepsilon_i z_i \leq x_i \leq \delta_i z_i. \tag{7}$$

  Those constraints are used to ensure diversification and to avoid tiny portions of assets in the portfolios, which would make their management difficult and lead to unnecessary transaction costs.

- **Preassignments**. This constraint is used to express subjective preferences: we want certain specific assets to be held in the portfolio, by determining a n-sized binary vector P (i.e., $p_i = 1$ if a_i has to be held in the portfolio) and imposing the following:

$$z_i \geq p_i \quad (i = 1, \ldots, n). \tag{8}$$

3 Metaheuristics

As stated in the Introduction, in this work we are solving the PSP by using metaheuristics [4], which can be defined as high-level strategies that coordinate the action of low-level algorithms (heuristics) in order to find near-optimal solutions for combinatorial optimization problem. They are used when it is impossible to find the certified optimum solution in a reasonable amount of time, and their features can be outlined as follows:

- They are used to explore the search space and to determine principles to guide the action of subordinated heuristics.
- Their level of complexity ranges from a simple escape-mechanism to complex populations procedures.
- They are stochastic, hence escape and restart procedures have to be devised in the experimental phase.
- The concepts they are built upon allow an abstract descriptions, that is useful to design hybrid procedures.
- They are not problem-specific, but additional components may be used to exploit the structure of the problem or knowledge acquired during the search process.
- They may make use of problem-specific knowledge in the form of heuristics that are controlled by the upper level strategy.

The main paradigm metaheuristics are build upon is the *intensification-diversification* paradigm, meaning that they should incorporate a mechanism to balance the exploration of promising regions of the search landscape (intensification) and the identification of new areas in the search landscape (diversification). The way of implementing this balance is different depending on the specific metaheuristic used. A completed description is out of the scope of this paper, and we forward the interested reader to Hoos and Stuetzle [18].

4 Our Approach for Portfolio Choice

We are using the solver introduced by di Tollo et al. [7, 9] to tackle a constrained PSP, in which the Markowitz' variance minimisation in a continuous formulation is enhanced by adding constraints (4), (6) and (7), leading to the following formulation:

$$\min \sum_{i=1}^{n} \sum_{j=1}^{n} \sigma_{ij} x_i x_j, \tag{9}$$

subject to

$$\sum_{i=1}^{n} r_i x_i \geq r_e, \tag{10}$$

$$\sum_{i=1}^{n} x_i = 1, \tag{11}$$

$$x_i \geq 0 \quad i = 1 \ldots n, \tag{12}$$

$$k_{min} \leq \sum_{i=1}^{n} z_i \leq k_{max}, \tag{13}$$

$$\varepsilon_i z_i \leq x_i \leq \delta_i z_i, \tag{14}$$

$$x_i \leq z_i \quad i = 1 \ldots n. \tag{15}$$

where k_{min} and k_{max} are respectively lower and upper bounds on cardinality. This problem formulation contains two classes of decision variables: integer (i.e., Z) and continuous (i.e., X). Hence, it is possible to devise an hybrid procedure in which each variable class is tackled by a different component. Starting from this principle, we have devised a master–slave decomposition, in which a metaheuristic procedure is used in order to determine, for each search step, assets contained in the portfolio (Z). Once the assets contained in the portfolio are decided, the corresponding continuous X values can be determined with proof of optimality. Hence at each step, after having selected which assets to be taken into account, we are resorting to a the Goldfarb–Idnani algorithm for quadratic programming (QP) [16] to determine their optimum value. The stopping criterion and escape mechanism depend on the metaheuristic used, which will be detailed in what follows.

As explained in Sect. 6, this master–slave decomposition has a dramatic impact on the metaheuristic performance due to the different structure determined by this formulation, in which the basin of attraction are greater than the ones determined by a monolithic approach based on the same metaheuristic approaches. In what follows we are outlining the components of our metaheuristic approach.

- **Search space** Since the *master* metaheuristic component takes into account the Z variables only, the search space S is composed of the 2^n portfolios that are feasible w.r.t cardinality and pre-assignment constraints, while other constraints are directly ensured by the *slave* QP procedure. If the QP procedure does not succeed in finding a feasible portfolio, a greedy procedure is used to find the portfolio with maximum return and minimum constraint violations.

- **Cost function** In our approach the cost function corresponds to the objective function of the problem σ^2, and is computed, at each step of the search process, by the *slave* QP procedure.
- **Neighborhood relations** As in di Tollo et al. [9], we are using three neighborhood relations in which the neighbor portfolio are generated by *adding*, *deleting* or *replacing* one asset: the neighbor is created by defining the asset pair $\langle i, j \rangle (i \neq j)$, inserting asset i, and deleting asset j. Addition is implemented by setting $j = 0$; deletion is implemented by $i = 0$.
- **Initial solution** The initial solution must be generated to create a configuration of Z. Since the we aim to generate an approximation of the unconstrained efficient frontier, we are devising three different procedures for generating the starting portfolio, which are used w.r.t. different r_e values: MaxReturn (in which the starting portfolio corresponds to the maximum return portfolio, without constraints on the risk); RandomCard (in which cardinality and assets are randomly generated); and WarmRestart (in which the starting portfolio corresponds to the optimal solution found for the previous r_e value). MaxReturn is used when setting the highest r_e value (i.e., first computed value); for all other r_e values both RandomCard and WarmRestart have been used.

4.1 Solution Techniques

As specific metaheuristics for the *master* procedure, we have used Steepest Descent (SD), First Descent (FD) and Tabu Search (TS). SD and FD are considered as the most simple metaheuristic strategies, since they accept the candidate solution only when its cost function is better than the current one, otherwise the search stops. They differ to each other in the neighborhood exploration, since in SD all neighbors are generated and the best one is compared to the current solution, while in FD the first better solution found is selected as current one. TS enhances this schema by selecting, as the new current solution, the best one amongst the neighborhood, and using an additional memory (Tabu list) in which forbidden states (i.e., former solutions) are stored, so that they cannot be generated as neighbors. In our implementation, we have used a dynamic-sized tabu list, in which solutions are put in the Tabu list for a randomly generated period of time. The length range of the Tabu list has been determined by using F-Race [3], and has been set to [3, 10].

The three metaheuristics components have been coded in C++ by Luca Di Gaspero and Andrea Schaerf and are available upon request.

As for the *slave* Quadratic programming procedure, we have used the Goldfarb and Idnani dual set method [16] to determine the optimal X values corresponding to Z values computed by the *master* metaheuristic component. This method has been coded in C++ by Luca Di Gaspero: it is available upon request, and has achieved good performances when matrices at hand are dense.

To sum up, the *master* metaheuristic component determines the actual configuration of Z variables (i.e., point of the search space), the *slave* QP procedure computes

the cost of the determined configuration, which is accepted (or not) depending on the mechanism embedded in FD, SD or TS.

4.2 Benchmark Instances

We have used instances from the repository `ORlib` (http://people.brunel.ac.uk/~mastjjb/jeb/info.html) and instances used in Crama and Schyns [6], which have been kindly provided to us by the authors. The UEF for the `ORlib` instances is provided in the aforementioned website; the UEF for instances from Crama and Schyns [6] has been generated by us by using our *slave* QP procedure. In both cases, the resulting UEF consists of 100 portfolios corresponding to 100 equally distributed r_e values. Benchmarks' main features are highlighted in Table 1.

By measuring the distance of the obtained frontier (CEF) from the UEF we obtain the *average percentage loss*, which is an indicator of the solution quality and which is defined as:

$$\mathsf{apl} = \frac{100}{p} \sum_{l=1}^{p} (V(r_e) - V_U(r_e))/V_U(r_e) \tag{16}$$

in which r_e is the minimum required return, p is the frontier cardinality, $V(r_e)$ and $V_U(r_e)$ are the values of the function F returned by the solver and the risk on the UEF.

4.3 Experimental Analysis

Our experiments have been run on a computer equipped with a Pentium 4 (3.2 GHz), and in what follows we are showing results obtained on both instance classes. In

Table 1 Our instances

`ORlib` dataset				Crama and Schyns dataset			
ID	Country	Assets	AVG(UEF)risk	ID	Country	Assets	AVG(UEF)risk
1	Hong Kong (Hang Seng)	31	1.55936×10^{-3}	S1	USA (DataStream)	20	4.812528
2	Germany (DAX 100)	85	0.412213×10^{-3}	S2	USA (DataStream)	30	8.892189
3	UK (FTSE 100)	89	0.454259×10^{-3}	S3	USA (DataStream)	151	8.64933
4	USA (S&P 100)	98	0.502038×10^{-3}				
5	Japan (NIKKEI)	225	0.458285×10^{-3}				

Table 2 Results over ORlib instances

Inst.	FD+QP		SD+QP		TS+QP		TS [29]		GA+QP [26]	
	Min apl	Time	Min apl	Time	Min apl	Time	Min apl	Time	Min apl	Time
1	0.00366	1.5	0.00321	3.1	0.00321	29.1	0.00409	251	0.00321	415.1
2	2.66104	9.6	2.53139	14.1	2.53139	100.9	2.53617	531	2.53180	552.7
3	2.00146	10.1	1.92146	16.1	1.92133	114.4	1.92597	583	1.92150	886.3
4	4.77157	11.2	4.69371	18.8	4.69371	130.5	4.69816	713	4.69507	1163.7
5	0.24176	25.3	0.20219	45.9	0.20210	361.8	0.20258	1603	0.20198	1465.8

Table 3 Results over Crama and Schyns instances

Inst.	FD+QP			SD+QP			TS+QP			SA [6]		
	apl		Time	apl		Time	apl		Time	apl		Time
S1	0.72	0.094	0.3	0.35	0.0	1.4	0.35	0.0	4.6	1.13	0.13	3.2
S2	1.79	0.22	0.5	1.48	0.0	3.1	1.48	0.0	8.5	3.46	0.17	5.4
S3	10.50	0.51	10.2	8.87	0.003	53.3	8.87	0.0003	124.3	16.12	0.43	30.1

order to assess the quality of our approach, in the following tables we also report results obtained by other works tackling the same instances. Table 2 reports results over ORlibinstances, showing that our approach outperforms the metaheuristic approach by Schaerf [29], and compares favourably with Moral-Escudero et al. [26].

Table 3 compares our results with the one by Crama and Schyns [6]: solutions found by our hybrid approach have better quality than the ones found by SA [6], but running times are higher, due to our QP procedure and to our complete neighbourhood exploration, which are not implemented by Crama and Schyns.

We have also compared our approach with Mixed Integer Non-linear Programming (MINLP) solvers, by encoding the problem in AMPL [14] and solving it using CPLEX 11.0.1 and MOSEK 5. We have run the MINLP solvers over ORLib instances, and compared their results with SD+QP (10 runs), obtaining the same solutions in the three approaches, hence showing that our approach is able to find the optimal solution in a low computational time. Computational times for SD+QP and for the MINLP solvers are reported in Table 4 and in Fig. 1. We can notice that for big-sized instances exact solvers require higher computation time to generate points in which cardinality constraints are binding (i.e., left part of the frontier). Our approach instead scales very well w.r.t. size and provides results which are comparable.

We can conclude this section by observing that SD+QP provides as satisfactory results as the more complex TS+QP. Since Tabu Search is conceived to better explore the search space, this can be considered rather surprising. The next sections will enlighten us about this phenomenon.

Table 4 Computational times over ORLib instances 1–4, SD+QP and MINLP

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

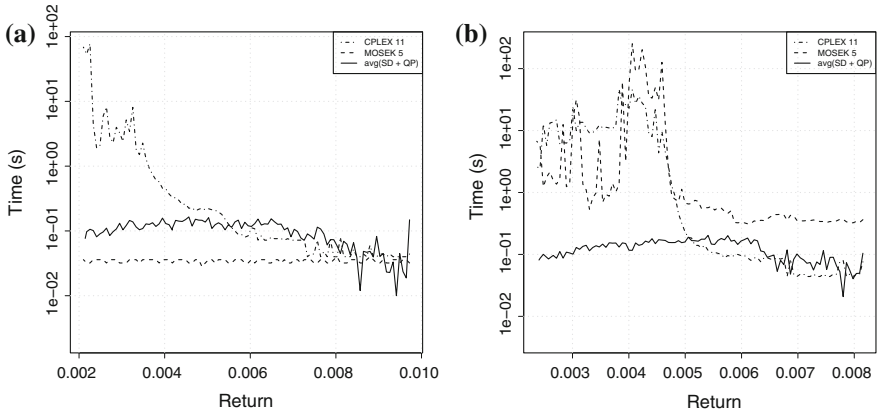

Fig. 1 Computational time: comparison between SD+QP and MINLP approaches over ORLib Instances. **a** Instance 2. **b** Instance 3

5 Search Space Analysis

The search process executed by a metaheuristic method can be viewed as a probabilistic walk over a discrete space, which in turn can be modelled as a graph: the vertices (usually named 'nodes' in this case) of the graph correspond to candidate solutions to the problem, while edges denote the possibility of locally transforming a solution into the other by means of the application of a local move. Therefore, algorithm behaviour depends heavily on the properties of this search space. A principled and detailed illustration of the most relevant techniques for search space analysis can be found in the book by Hoos and Stützle [18].

In this work we focus on a specific and informative feature of the search space, the *basin of attraction* (BOA), defined in the following.

Definition Given a deterministic algorithm $\mathscr{A}$, the basin of attraction $\mathscr{B}(\mathscr{A}|s)$ of a point s, is defined as the set of states that, taken as initial states, give origin to trajectories that include point s.

Let S^* be the set of global optima: for each $s \in S^*$ there exist a basin of attraction, and their union $I^* = \bigcup_{i \in S^*} \mathscr{B}(\mathscr{A}|i)$ contains the states that, taken as a starting solution, would have the search provide a certified global optimum. Hence, if we use a randomly chosen state as a starting solution, the ratio $|I^*|/|S|$ would measure

the probability to find an optimal solution. As a generalization, we are defining a probabilistic basin of attraction as follows:

Definition Given a stochastic algorithm $\mathscr{A}$, the basin of attraction $\mathscr{B}(\mathscr{A}|s;\, p^*)$ of a point s, is defined as the set of states that, taken as initial states, give origin to trajectories that include point s *with probability* $p \geq p^*$. Accordingly, the union of the BOA of global optima is defined as $I^*(p) = \bigcup_{i \in S^*} \mathscr{B}(\mathscr{A}|i;\, p)$. It is clear that that $\mathscr{B}(\mathscr{A}|s)$ is a special case for $\mathscr{B}(\mathscr{A}|s;\, p^*)$, hence in what follows we are using $\mathscr{B}(s;\, p^*)$ instead of $\mathscr{B}(\mathscr{A}|s;\, p^*)$, without loss of generalization. When $p^* = 1$ we want to find solutions belonging to trajectories that ends in s. Notice that $\mathscr{B}(s;\, p_1) \subseteq \mathscr{B}(s;\, p_2)$ when $p_1 > p_2$.

Topology and structure of the search space have a dramatic impact on the effectiveness of a metaheuristic, and since the aim is to reach an optimal solution, the need of an analysis of BOA features arises.Note that our definition of basins of attraction enables both a complete/analytical study—when probabilities can be deducted from the search strategy features—and a statistical/empirical analysis (e.g., by sampling).

In our metaheuristic model, we define BOAs as sets of search graph nodes. For this definition to be valid for any state of the search graph [2], we are relaxing the requirement that the goal state is an attractor. Therefore, the BOA also depends on the particular termination condition of the algorithm. In the following examples, we will suppose to end the execution as soon as a stagnation condition is detected, i.e., when no improvements are found after a maximum number of steps.

6 Search Space Analysis for Portfolio Selection Problem

When solving an optimisation problem, a sound modelling and development phase should be based on the separation between the model and the algorithm: this stems from constraint programming, and several tools foster this approach (i.e., Comet [17]). In this way, it is possible to draw information about the structure of the optimisation problem, and this knowledge can be used, for instance, for the choice of the algorithm to be used. Up to the author's knowledge, literature about portfolio selection by metaheuristics has hardly dealt with this aspect, though some attempts have been made to study the problem structure. For instance, Maringer and Winker [24] draw some conclusion about the objective function landscape by using a memetic algorithm which embeds, in turn, Simulated Annealing (SA) [20] and Threshold Acceptance (TA) [11]. They compare the use of SA and TA inside the memetic algorithm dealing with different objective functions: Value-at-Risk(*Var*) and Expected Shortfall (*ES*) [8]. Their results indicates that TA is suitable when using *VaR*, while SA performs best when using *ES*. An analysis of the search space is made to understand this phenomenon.

Other works compare different algorithms on the same instance to understand which algorithm perform best, and in what portion of the frontier. Amongst them, Crama and Schyns [6] introduce three different Simulated Annealing strategies,

showing that there is no clear dominance among them. Armañanzas and Lozano [1] introduces Ant Colony Optimisation (ACO) [10], refining solutions with a greedy search, comparing results with Simulated Annealing and Iterative Improvement, and showing that ACO and SA performances greatly depends on the expected return (see Sect. 2). A common way of tackling this analysis is to run the different algorithms, and then to pool the obtained solutions. After this phase, the dominated solutions are deleted and it is possible to understand which algorithm performs best w.r.t. a given part of the frontier [5, 13].

The main shortcoming of these approaches is that they identify which algorithm performs well in a given portion of the frontier, without explaining the motivation beneath this behaviour. Hence, an additional effort has to be made to understand the model and how it can affect the algorithm performance. In this section, we are aimed in comparing different formulations for the PSP and in understanding how the structure of the problem affects the algorithm's performances through Search Space Analysis.

When using a metaheuristic, search space analysis represents an effective tool to assess the algorithm performances and the instance hardness. In what follows we are discussing results obtained over real instances and over hard-handmade instances in order to outline the connections between search space analysis and algorithm performances.

Analysis for Real Instances

We define five equally distributed r_e values, referred to as R_i ($i = 1 \ldots 5$) and we analyse the search space corresponding to each r_i over the five `ORlib` instances in order to assess the local minima distribution, that is an indicator of the search space ruggedness. This concept is important since it has been shown that there exists a negative correlation between ruggedness and metaheuristic performances [18]. We have implemented and run a deterministic version of SD (referred to as SD_{det}) to estimate the number of minima of an instance of the problem discussed in Sect. 4, which combines continuous variables x with integer variables z. As for the constraints, we have set both a minimum (k_{min}) or a maximum (k_{max}) bound on cardinality in order to understand the differences arising when using a maximum or strict cardinality constraint. As for determining the initial states, we have resorted either to complete enumeration (if the instance at hand is small) or to uniform sampling.

Results are shown in Table 5, where we report the number of the different local minima found by 30 runs of SD_{det}. Dashed entries mean that no feasible solution exists.

Results indicate that instances at hand show a small number of local minima and only one global minimum. This clearly indicates a situation in which the search landscape is rather smooth, and explains why different strategies such TS and FD/SD lead to similar optimization results: since local optimum are few and far between, there is no need of using complex strategies or escape mechanisms, since the probability of meeting a trajectory leading to one of the optima are quite high. We recall that those values have been found by using a deterministic version of SD, and their inverse

Table 5 Instance 4, number of minima found

k_{min}, k_{max}	$R_1 = 0.00912$	$R_2 = 0.00738$	$R_3 = 0.00556$	$R_4 = 0.00375$	$R_5 = 0.00193$
1,3	1	1	1	1	1
1,6	1	1	1	5	1
1,10	1	1	1	1	3
3,3	1	1	3	5	3
6,6	–	1	1	2	1
10,10	–	1	1	3	2

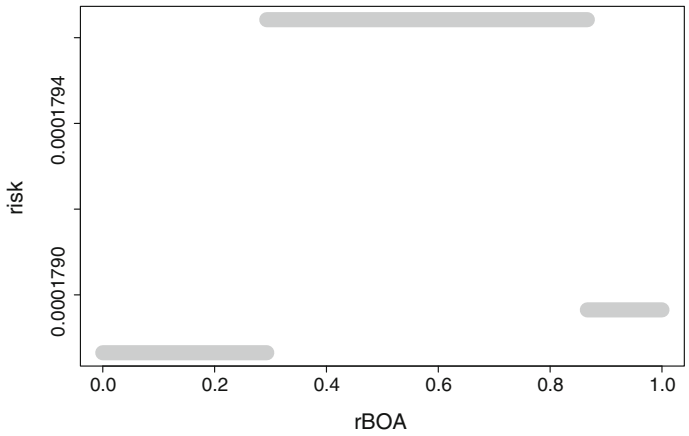

Fig. 2 Instance 4: BOA analysis. $k_{min} = 1$, $k_{max} = 10$, $R = 0.00375$

represents an upper bound on the probability to reach the certified optimum when using the stochastic SD and TS defined in Sect. 4.1.

We conclude that when using our formulation, global minima have a quite large BOA. A pictorial view of an example of this is provided in Fig. 2, where segments length corresponds to *rBOA* (i.e., ratio between size of *BOA(s)* and search space size) and their y-value corresponds to the minimum found: global minima *rBOA* ranges from 30 to 60 %.

Search space autocorrelation

A further analysis of the search space with respect to the study of BOA is the estimation of the *autocorrelation* of the search landscape [19]. This measure estimate the extent to which a local move from a solution leads to a destination state with similar objective function value. Smooth landscape, where it is easy to move towards better solutions, are characterized by a high autocorrelation value; conversely, low autocorrelation values are typical of landscape in which the search if often trapped in local optima or anyway very loosely guided towards good solution just by exploring the

Table 6 Autocorrelation of lag $k = 1, \ldots 10$ estimated for the three instances

Lag	Instance 2	Instance 3	Instance 4
1	0.99	0.95	0.91
2	0.98	0.90	0.90
3	0.97	0.86	0.87
4	0.97	0.81	0.86
5	0.96	0.76	0.77
6	0.95	0.70	0.75
7	0.94	0.65	0.73
8	0.93	0.61	0.72
9	0.93	0.55	0.69
10	0.92	0.50	0.57

neighbor of incumbent solutions. The autocorrelation of the search space may help elucidating the differences among algorithm performance across different instances, providing further bits of information besides the BOA analysis.

The autocorrelation of a series $G = (g_1, \ldots, g_m)$ of objective function values is computed as

$$r = \frac{\sum_{k=1}^{m-1} (g_k - \overline{g}) \cdot (g_{k+1} - \overline{g})}{\sum_{k=1}^{m} (g_k - \overline{g})^2},$$

where $\overline{g}$ is the average value of the series. This definition refers to the autocorrelation of length one, i.e., that corresponding to series generated by sampling neighbouring states at distance 1 in the search space. In general, we can consider the autocorrelation of lag k.

We performed a random walk of 1000 steps over the search space in the case of the three instances considered in this study and computed the autocorrelation of lag $k = 1, 2, \ldots 10$. In this way we can estimate with more precision the hardness of each instance. Results are shown in Table 6. As we can observe, the autocorrelation value for $k = 1$ is quite high for all the three instances, confirming the fact that in general the instances are quite easy for this combination of model and algorithm, as already observed in the case of BOA analysis. However, some differences can be observed considering the autocorrelation decay when the lag increases: instance 2 is by far the one with the highest autocorrelation, while we observe that instance 3 has a faster decrease with respect to the other two instances, making search slightly more difficult in case that the initial solution is not in the BOA of the optimal solution.

In conclusion, we can confirm that the three instances are considerably easy for local search. In the next paragraph we will show that the same problem, modeled in a different way, leads to different basin of attractions.

Monolithic Search Basin of Attraction

In the previous paragraph we have shown that, when using our problem formulation, the BOAs of local optima are quite big, making the search landscape smooth and the problem easy to be tackled by our hybrid solver. BOAs depend on the search strategy used and on the problem formulation, and this can be shown by running a different strategy, i.e., a monolithic one, on the same problem instances. We have used a SD based on a variant of Threshold Accepting [12], in which only a variable class is considered, i.e., w variables corresponding to actual asset weights. The desired outcome of this problem is the same as the previously introduced one, but they are represented in a different way. In the following we explain the main features of this metaheuristic approach:

- **Search Space** The *master–slave* decomposition is not used anymore, and a state is represented by a sequence $W = w_1 \dots w_n$ such that w_b corresponds to the relative amount invested in asset b. Furthermore, the portfolio has to be feasible w.r.t. cardinality, budget, floor and ceiling constraints.
- **Neighborhood relations** A given amount ($step$) is transferred from asset a to another b, no matter if b is already in the portfolio or not. If this leads one asset value to be smaller than ε_i, its value is set to ε_i. If the move consists in decreasing the value of an asset being set to ε_i, its value is set to 0.
- **Initial solution** The initial solution has to be feasible w.r.t. cardinality, budget, floor and ceiling constraints and is always created from scratch.
- **Cost Function** As for the cost function we are using a penalty approach, hence it is given by adding the degree of constraints violations to the portfolio risk.
- **Local Search Strategies** SD that explores the space of w variables.

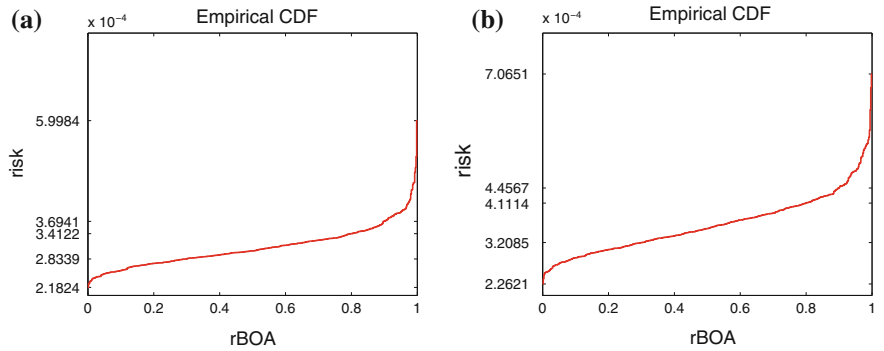

Fig. 3 Two `ORlib` instances: Monolithic BOA analysis with different constraints. **a** Instance 4: $k_{min} = 1$, $k_{max} = 10$, $R = 0.00375$. **b** Instance 4: $k_{min} = 1$, $k_{max} = 6$, $R = 0.00193$

$$\sigma = \left(\begin{array}{cc|cc|cc|ccc|cc} 1 & -1 & 0 & 0 & 0 & & \cdots & & & & 0 \\ -1 & 1 & 0 & 0 & 0 & & \cdots & & & & 0 \\ \hline 0 & 0 & 1 & -0.9 & 0 & 0 & 0 & \cdots & & & \vdots \\ 0 & 0 & -0.9 & 1 & 0 & 0 & 0 & \cdots & & & \vdots \\ \hline \vdots & & 0 & 0 & 1 & -0.9 & 0 & \cdots & & & \vdots \\ \vdots & & 0 & 0 & -0.9 & 1 & 0 & \cdots & & & \vdots \\ \hline \vdots & & \ddots & & 0 & 0 & \ddots & & & & \vdots \\ \vdots & & & \ddots & \vdots & \vdots & & \ddots & & & \vdots \\ \vdots & & & & \cdots & & & & 1 & 0 & 0 \\ \hline \vdots & & & & \cdots & & & & 0 & 1 & -0.9 \\ 0 & & & & \cdots & & & & 0 & -0.9 & 1 \end{array}\right) \tag{6}$$

Results about BOAs analysis for this approach are shown in Fig. 3. Even from visual inspection only, it turns out that the number of local minima is dramatically higher than the one corresponding to the *master–slave* approach; furthermore basin of attraction are tiny, and the certified optimum has not been found.

Analysis for artificial instances

In the previous paragraph we have shown that, for the PSP we are solving, instances at hand are easy to solve, since our *master–slave* decomposition leads to search spaces with a small number of local optima with huge BOAs. Hence, there is no need for complex approaches and escape mechanisms, and this explains why simple metaheuristics performances are comparable with more sophisticated one such TS. Furthermore, preliminary analysis have suggested us that this is a common feature in financial market related instances: this could be considered as a good point for practitioners, but makes impossible to test the robustness of our approach, in which we have developed TS+QP in order to tackle more difficult instances. Hence, we have designed an artificial hand-made instance featuring a huge number of minima with tiny BOAs, containing an even number n of assets i, in which $r_i = 1 \forall i$ and whose covariance matrix is depicted here above.

It is easy to see that for every r_e the best portfolio contains the first two assets only, but also that portfolios consisting of assets i (odd) and $i + 1$ only are local optima, since all their neighbors feature higher risk.

It can be shown show that it is necessary to visit a portfolio s having $z_1 = 1$ or $z_2 = 1$ to reach the global optimum s^*. Furthermore, portfolios containing an odd asset i ($i > 1$) whose $z_i = 1$ and $z_{i+1} = 1$ will never entry in a trajectory in which this couple would be removed. Hence, $\mathscr{B}(s^*)$ contains all portfolios featuring $z_1 = 1$ or $z_2 = 1$, and in which there is no i odd and > 1 such that $z_i = 1$ and $z_{i+1} = 1$. In this case, $rBOA(s^*)$ is inversely proportional to n.

By running our master–slave approach over this instance ($\varepsilon_i = 0.01$ and $\delta_i = 1$ for $i = 1 \ldots n$) we have remarked that TS+QP easily find a solution comparable to that provided by CPLEX, while SD and FD performances are greatly affected by the starting solution (and anyhow much poorer than TS+QP).

It has to be noticed that such an instance could be hardly found over real markets, even its presence is not forbidden by structural properties, but when tackling it the need of larger neighborhoods arises. Anyhow, no matter the neighborhood size, it is always possible to devise artificial instances whose minima are composed by subsets that have to be moved jointly.

From the Search Space Analysis conducted in this section, we may conclude that different formulations (hybrid vs continuous only) lead to different Basin of Attraction analysis on the instances at hand. This turns into different algorithm behaviours. The formulation that leads to a smooth search landscape (hybrid) can be tackled by algorithms with weak diversification capabilities (i.e., SD in the proposed hybrid formulation), whilst these algorithms are to be replaced by more sophisticated ones when the search landscape becomes rugged (see the behaviour of SD in the monolithic version). The artificial instance places itself in the middle of these phenomena, as it provides room for the use of more complex strategies (i.e., TS) in the hybrid case, due to the neighbor moves used which make the search to get stuck in the first local optimum found, but when embedded in the continuous only formulation doesn't provide different performances from the real instances.

7 Conclusion

In this work we have used a metaheuristic approach to study the impact of different formulations on the Portfolio Selection algorithm's behaviour, and we have devised a methodology to understand the root of the different behaviours (search space analysis through BOA analysis). To this aim we have compared an approach based on a master–slave decomposition with a monolithic approach. Results have shown that the search space defined by the monolithic approach is quite rugged and need an algorithm featuring an escape mechanism to be solved efficiently, whilst the hybrid approach leads to a smoother search landscape to be explored efficiently also by simpler algorithms such SD.

References

1. Armañanzas, R., Lozano, J.A.: A multiobjective approach to the portfolio optimization problem. In: Proceedings of the 2005 IEEE Congress on Evolutionary Computation, vol. 2, pp. 1388–1395 (2005)
2. Roli, A.: A note on a model of local search. Technical Report TR/IRIDIA/2004/23.01, IRIDIA, ULB, Belgium (2004)
3. Birattari, M., Stützle, T., Paquete, L., Varrentrapp, K.: A racing algorithm for configuring metaheuristics. In: Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2002), pp. 11–18. Morgan Kaufmann Publishers (2002)

4. Blum, C., Roli, A.: Metaheuristics in combinatorial optimization: overview and conceptual comparison. ACM Comput. Surv. **35**(3), 268–308 (2003)
5. Chang, T.J., Meade, N., Beasley, J.E., Sharaiha, Y.M.: Heuristics for cardinality constrained portfolio optimisation. Comput. Oper. Res. **27**(13), 1271–1302 (2000)
6. Crama, Y., Schyns, M.: Simulated annealing for complex portfolio selection problems. Eur. J. Oper. Res. **150**, 546–571 (2003)
7. Di Gaspero, L., di Tollo, G., Roli, A., Schaerf, A.: Hybrid local search for constrained financial portfolio selection problems. In: Proceedings of Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems, pp, 44–58 (2007)
8. di Tollo, G., Roli, A.: Metaheuristics for the portfolio selection problem. Int. J. Oper. Res. **5**(1), 443–458 (2008)
9. di Tollo, G., Stützle, T., Birattari, M.: A metaheuristic multi-criteria optimisation approach to portfolio selection. J. Appl. Oper. Res. **6**(4), 222–242 (2014)
10. Dorigo, M., Gambardella, L.M., Middendorf, M., Stützle, T. (eds.): Special Section on Ant Colony Optimization. IEEE Trans. Evol. Comput. **6**(4), 317–365 (2002)
11. Dueck, G., Scheuer, T.: Threshold accepting: a general purpose optimization algorithm appearing superior to simulated annealing. J. Comput. Phys. **90**(1), 161–175 (1990)
12. Dueck, G., Winker, P.: New concepts and algorithms for portfolio choice. Appl. Stoch. Models Data Anal. **8**, 159–178 (1992)
13. Fernandez, A., Gomez, S.: Portfolio selection using neural networks. Comput. Oper. Res. **34**, 1177–1191 (2007)
14. Fourer, R., Gay, D.M., Kernighan, B.W.: AMPL: A Modeling Language for Mathematical Programming. Duxbury Press/Brooks/Cole Publishing Company, Pacific Grove (2002)
15. Di Gaspero, L., di Tollo, G., Roli, A., Schaerf, A.: Hybrid metaheuristics for constrained portfolio selection problems. Quant. Financ. **11**(10), 1473–1487 (2011)
16. Goldfarb, D., Idnani, A.: A numerically stable dual method for solving strictly convex quadratic programs. Math. Program. **27**, 1–33 (1983)
17. Van Hentenryck, P., Michel, L.: Constraint-Based Local Search. The MIT Press, Cambridge (2005)
18. Hoos, H., Stützle, T.: Stochastic Local Search Foundations and Applications. Morgan Kaufmann Publishers, Burlington (2005)
19. Hordijk, W.: A measure of landscapes. Evol. Comput. **4**, 335–360 (1996)
20. Kirkpatrick, S., Gelatt, C.D., Vecchi, M.P.: Optimization by simulated annealing. Science **220**(4598), 671–680 (1983)
21. Mansini, R., Ogryczak, W., Speranza, M.G.: LP solvable models for portfolio optimization: a classification and computational comparison. IMA J. Manag. Math. **14**(3), 187–220 (2003)
22. Mansini, R., Speranza, M.G.: Heuristic algorithms for the portfolio selection problem with minimum transaction lots. Eur. J. Oper. Res. **114**(2), 219–233 (1999)
23. Maringer, D.: Portfolio Management with Heuristic Optimization. Springer, Heidelberg (2005)
24. Maringer, D., Winker, P.: Portfolio optimization under different risk constraints with modified memetic algorithms. Technical Report 2003-005E, University of Erfurt, Faculty of Economics, Law and Social Sciences (2003)
25. Markowitz, H.: Portfolio selection. J. Financ. **7**(1), 77–91 (1952)
26. Moral-Escudero, R., Ruiz-Torrubiano, R., Suárez, A.: Selection of optimal investment with cardinality constraints. In: Proceedings of the IEEE World Congress on Evolutionary Computation, pp. 2382–2388 (2006)
27. Prestwich, S., Roli, A.: Symmetry breaking and local search spaces. In: Proceedings of the 2nd International Conference on Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems, pp. 273–287 (2005)
28. Rolland, E.: A tabu search method for constrained real number search:applications to portfolio selection. Technical report, Department of Accounting and Management Information Systems, Ohio State University, Columbus. U.S.A. (1997)
29. Schaerf, A.: Local search techniques for constrained portfolio selection problems. Comput. Econ. **20**(3), 177–190 (2002)

30. Speranza, M.G.: A heuristic algorithm for a portfolio optimization model applied to the Milan stock market. Comput. Oper. Res. **23**(5), 433–441 (1996)
31. Streichert, F., Ulmer, H., Zell, A.: Comparing discrete and continuous genotypes on the constrained portfolio selection problem. In: Proceedings of Genetic and Evolutionary Computation Conference. LNCS, vol. 3103, pp. 1239–1250 (2004)
32. Yokoo, M.: Why adding more constraints makes a problem easier for hill-climbing algorithms: Analyzing landscapes of CSPs. In: Proceedings of the Third Conference on Principles and Practice of Constraint Programming, pp. 356–370 (1997)

http://www.springer.com/978-3-319-40131-7

Recent Advances in Computational Optimization
Results of the Workshop on Computational Optimization
WCO 2015
Fidanova, S. (Ed.)
2016, X, 303 p. 102 illus., 55 illus. in color., Hardcover
ISBN: 978-3-319-40131-7