Simulated annealing for complex portfolio selection problems

Y. Crama [a], M. Schyns [a,b,*]

[a] *University of Liege, School of Business Administration, Bd. du Rectorat 7 (B31), Liege 4000, Belgium*
[b] *University of Namur, Department of Business Administration, Rpt. de la Vierge 8, Namur 5000, Belgium*

Abstract

This paper describes the application of a simulated annealing approach to the solution of a complex portfolio selection model. The model is a mixed integer quadratic programming problem which arises when Markowitz' classical mean–variance model is enriched with additional realistic constraints. Exact optimization algorithms run into difficulties in this framework and this motivates the investigation of heuristic techniques. Computational experiments indicate that the approach is promising for this class of problems.
© 2003 Elsevier B.V. All rights reserved.

Keywords: Finance; Simulated annealing; Metaheuristics; Portfolio selection; Quadratic programming

1. Introduction

Markowitz' mean–variance model of portfolio selection is one of the best known models in finance. In its basic form, this model requires to determine the composition of a portfolio of assets which minimizes risk while achieving a predetermined level of expected return. The pioneering role played by this model in the development of modern portfolio theory is unanimously recognized (see e.g. [7] for a brief historical account).

From a practical point of view, however, the Markowitz model may often be considered too basic, as it ignores many of the constraints faced by real-world investors: trading limitations, size of the portfolio, etc. Including such constraints in the formulation results in a nonlinear mixed integer programming problem which is considerably more difficult to solve than the original model. Several researchers have attempted to attack this problem by a variety of techniques (decomposition, cutting planes, interior point methods, . . .), but there appears to be room for much improvement on this front. In particular, exact solution methods fail to solve large-scale instances of the problem. Therefore, in this paper, we investigate the ability of the simulated annealing metaheuristic (SA) to deliver high-quality solutions for the mean–variance model enriched by additional constraints.

The remainder of this paper is organized in six sections. Section 2 introduces the portfolio selection model that we want to solve. Section 3 sums up the basic structure of simulated annealing algorithms. Section 4 contains a detailed description of our algorithm. Here, we make an attempt to underline the difficulties

---

* Corresponding author. Tel.: +32-81-724877; fax: +32-81-724840.
E-mail addresses: y.crama@ulg.ac.be (Y. Crama), michael.schyns@fundp.ac.be (M. Schyns).

encountered when tailoring the SA metaheuristic to the problem at hand. Notice, in particular, that our model involves continuous as well as discrete variables, contrary to most applications of simulated annealing. Also, the constraints are of various types and cannot be handled in a uniform way. In Section 5, we discuss some details of the implementation. Section 6 reports on computational experiments carried out on a sample of 151 US stocks. Finally, the last section contains a summary of our work and some conclusions.

2. Portfolio selection issues

2.1. Generalities

In order to handle portfolio selection problems in a formal framework, three types of questions (at least) must be explicitly addressed.

1. Data modelling, in particular the behavior of asset returns.
2. The choice of the optimization model, including
   - the nature of the objective function;
   - the constraints faced by the investor.
3. The choice of the optimization technique.

Although our work focuses mostly on the third step, we briefly discuss the whole approach since all the steps are interconnected to some extent.

The first requirement is to understand the nature of the data and to be able to correctly represent them. Markowitz' model (described in the next section) assumes for instance that the asset returns follow a multivariate normal distribution. In particular, the first two moments of the distribution suffice to describe completely the distribution of the asset returns and the characteristics of the different portfolios. Real markets often exhibit more intricacies, with distributions of returns depending on moments of higher-order (skewness, kurtosis, etc.), and distribution parameters varying over time. Analyzing and modelling such complex financial data is a whole subject in itself, which we do not tackle here explicitly. We rather adopt the classical assumptions of the mean–variance approach, where (pointwise estimates of) the expected returns and the variance–covariance matrix are supposed to provide a satisfactory description of the asset returns. Also, we do not address the origin of the numerical data. Note that some authors rely for instance on factorial models of the asset returns, and take advantage of the properties of such models to improve the efficiency of the optimization techniques (see e.g. [2,22]). By contrast, the techniques that we develop here do not depend on any specific properties of the data, so that some changes of the model (especially of the objective function) can be performed while preserving our main conclusions.

When building an optimization model of portfolio selection, a second requirement consists in identifying the objective of the investor and the constraints that he is facing. As far as the objective goes, the quality of the portfolio could be measured using a wide variety of utility functions. Following again Markowitz' model, we assume here that the investor is risk averse and wants to minimize the variance of the investment portfolio subject to the expected level of final wealth. It should be noted, however, that this assumption does not play a crucial role in our algorithmic developments, and that the objective could be replaced by a more general utility function without much impact on the optimization techniques that we propose.

As far as the constraints of the model go, we are especially interested in two types of complex constraints limiting the number of assets included in the portfolio (thus reflecting some behavioral or institutional restrictions faced by the investor), and the minimal quantities which can be traded when rebalancing an existing portfolio (thus reflecting individual or market restrictions). This topic is covered in more detail in Section 2.2.2.

The final ingredient of a portfolio selection method is an algorithmic technique for the optimization of the chosen model. This is the main topic of the present paper. In view of the complexity of our model (due, to a large extent, to the constraints mentioned in the previous paragraph), and to the large size of realistic problem instances, we have chosen to work with a simulated annealing metaheuristic. An in-depth study has been performed to optimize the speed and the quality of the algorithmic process, and to analyze the impact of various parameter choices.

In the remainder of this section, we return in more detail to the description of the model, and we briefly survey previous work on this and related models.

2.2. The optimization model

2.2.1. The Markowitz mean–variance model

The problem of optimally selecting a portfolio among n assets was formulated by Markowitz in 1952 as a constrained quadratic minimization problem (see [11,19,20]). In this model, each asset is characterized by a return varying randomly with time. The risk of each asset is measured by the variance of its return. If each component x_i of the n-vector x represents the proportion of an investor's wealth allocated to asset i, then the total return of the portfolio is given by the scalar product of x by the vector of individual asset returns. Therefore, if $R = (R_1, \ldots, R_n)$ denotes the n-vector of expected returns of the assets and C the $n \times n$ covariance matrix of the returns, we obtain the mean portfolio return by the expression $\sum_{i=1}^{n} R_i x_i$ and its level of risk by $\sum_{i=1}^{n} \sum_{j=1}^{n} C_{ij} x_i x_j$.

Markowitz assumes that the aim of the investor is to design a portfolio which minimizes risk while achieving a predetermined expected return, say $R_{\exp}$. Mathematically, the problem can be formulated as follows for any value of $R_{\exp}$:

$$
\begin{aligned}
\min \quad & \sum_{i=1}^{n} \sum_{j=1}^{n} C_{ij} x_i x_j \\
\text{s.t.} \quad & \sum_{i=1}^{n} R_i x_i = R_{\exp}, \\
& \sum_{i=1}^{n} x_i = 1, \\
& x_i \geqslant 0 \quad \text{for } i = 1, \ldots, n.
\end{aligned}
\tag{1}
$$

The first constraint expresses the requirement placed on expected return. The second constraint, called *budget constraint*, requires that 100% of the budget be invested in the portfolio. The nonnegativity constraints express that no short sales are allowed.

The set of optimal solutions of the Markowitz model, parametrized over all possible values of $R_{\exp}$, constitutes the *mean–variance frontier* of the portfolio selection problem. It is usually displayed as a curve in the plane where the ordinate is the expected portfolio return and the abscissa is its standard deviation.

If the goal is to draw the whole frontier, an alternative form of the model can also be used where the constraint defining the required expected return is removed and a new weighted term representing the portfolio return is included in the objective function. Hereafter, we shall use the initial formulation involving only the variance of the portfolio in the objective function.

2.2.2. Extensions of the basic model

In spite of its theoretical interest, the basic mean–variance model is often too simplistic to represent the complexity of real-world portfolio selection problems in an adequate fashion. In order to enrich the model, we need to introduce more realistic constraints. The present section discusses some of them.

Consider the following portfolio selection model (similar to a model described by Perold [22]):

Model (PS):

$$
\begin{array}{lll}
\min & \sum_{i=1}^{n} \sum_{j=1}^{n} C_{ij} x_i x_j & \text{objective function} \\
\text{s.t.} & \sum_{i=1}^{n} R_i x_i = R_{\exp} & \text{return contraint} \\
& \sum_{i=1}^{n} x_i = 1 & \text{budget constraint} \\
& \underline{x}_i \leqslant x_i \leqslant \bar{x}_i \ (1 \leqslant i \leqslant n) & \text{floor and ceiling constraints} \\
& \max(x_i - x_i^{(0)}, 0) \leqslant \bar{B}_i \ (1 \leqslant i \leqslant n) & \text{turnover (purchase) constraints} \\
& \max(x_i^{(0)} - x_i, 0) \leqslant \bar{S}_i \ (1 \leqslant i \leqslant n) & \text{turnover (sale) constraints} \\
& x_i = x_i^{(0)} \text{ or } x_i \geqslant (x_i^{(0)} + \underline{B}_i) & \\
& \quad \text{or } x_i \leqslant (x_i^{(0)} - \underline{S}_i) \ (1 \leqslant i \leqslant n) & \text{trading constraints} \\
& |\{i \in \{1, \ldots, n\} : x_i \neq 0\}| \leqslant N & \text{maximum number of assets}
\end{array}
$$

- *Return and budget constraints*. These two constraints have already been encountered in the basic model.
- *Floor and ceiling constraints*. These constraints define lower and upper limits on the proportion of each asset which can be held in the portfolio. They may model institutional restrictions on the composition of the portfolio. They may also rule out negligible holdings of asset in the portfolio, thus making its control easier. Notice that the floor constraints generalize the nonnegativity constraints imposed in the original model.
- *Turnover constraints*. These constraints impose upper bounds on the variation of the holdings from one period to the next. Here, $x_i^{(0)}$ denotes the weight of asset i in the initial portfolio, $\bar{B}_i$ denotes the maximum purchase and $\bar{S}_i$ denotes the maximum sale of asset i during the current period $(1 \leqslant i \leqslant n)$. Notice that such limitations could also be modelled, indirectly, by incorporating transaction costs (taxes, commissions, unliquidity effects, ...) in the objective function or in the constraints.
- *Trading constraints*. Lower limits on the variations of the holdings can also be imposed in order to reflect the fact that, typically, an investor may not be able, or may not want, to modify the portfolio by buying or selling tiny quantities of assets. A first reason may be that the contracts must bear on significant volumes. Another reason may be the existence of relatively high fixed costs linked to the transactions. These constraints are disjunctive in nature: for each asset i, either the holdings are not changed, or a minimal quantity $\underline{B}_i$ must be bought, or a minimal quantity $\underline{S}_i$ must be sold.
- *Maximum number of assets*. This constraint limits to N the number of assets included in the portfolio, e.g. in order to facilitate its management.

2.3. Solution approaches

The complexity of solving portfolio selection problems is very much related to the type of constraints that they involve.

The simplest situation is obtained when the nonnegativity constraints are omitted from the basic model (1) (thus allowing short sales; see e.g. [11,16,19]). In this case, a closed-form solution is easily obtained by classical Lagrangian methods and various approaches have been proposed to increase the speed of resolution for the computation of the whole mean–variance frontier or the computation of a specific portfolio combined with an investment at the risk-free interest rate [16].

The problem becomes more complex when nonnegativity constraints are added to the formulation, as in the Markowitz model (1). The resulting *quadratic programming* problem, however, can still be solved efficiently by specialized algorithms such as Wolfe's adaptation of the simplex method [26]. The same technique allows to handle arbitrary linear constraints, like the floor and ceiling constraints or the turnover constraints. Notice, however, that even in this framework, the problem becomes increasingly hard to manage and to solve as the number of assets increases. As a consequence, ad hoc methods have been developed to take advantage of the sparsity or of the special structure of the covariance matrix, (e.g. when factor models of returns are postulated; see [2,22]).

When the model involves constraints on minimal trading quantities or on the maximum number of assets in the portfolio, as in model (PS), then we enter the field of mixed integer nonlinear programming and classical algorithms are typically unable to deliver the optimal value of the problem. (Actually, very few commercial packages are even able to handle this class of problems.)

Several researchers took up this challenge, for various versions of the problem. Perold [22], whose work is most often cited in this context, included a broad class of constraints in his model, but did not place any limitation on the number of assets in the portfolio. His optimization approach is explicitly restricted to the consideration of factorial models, which, while reducing the number of decision variables, lead to other numerical and statistical difficulties. Moreover, some authors criticize the results obtained when his model is applied to certain types of markets.

Several other researchers have investigated variants of model (PS) involving only a subset of the constraints. This is the case for instance of Dembo et al. [10] (with network flow models), Konno and Yamazaki [15] (with an absolute deviation approach to the measure of risk, embedded in linear programming models), Takehara [24] (with an interior point algorithm) and Bienstock [2] (with a 'branch and cut' approach). Dahl et al. [8], Takehara [24] and Hamza and Janssen [12] discuss some of this work.

Few authors seem to have investigated the application of local search metaheuristics for the solution of portfolio selection problems. Catanas [3] has investigated some of the theoretical properties of a neighborhood structure in this framework. Loraschi et al. [17] proposed a genetic algorithm approach. Chang et al.'s work [5] is closest to ours (and was carried out concurrently). These authors have experimented with a variety of metaheuristics, including simulated annealing, on model (PS) *without* trading and turnover constraints. As we shall see in Section 6, the trading constraints actually turned out to be the hardest to handle in our experiments, and they motivated much of the sophisticated machinery described in Section 4. We also work directly with the return constraint in equality form, rather than incorporating it as a Lagrangian term in the objective function. This allows us to avoid some of the difficulties linked to the fact that, as explained in [5], the efficient frontier cannot possibly be mapped entirely in the Lagrangian approach, due to its discontinuity. In this sense, our work can be viewed as complementary to [5].

We propose to investigate the solution of the complete model (PS) presented in Section 2.2.2 by a simulated annealing algorithm. Our goal is to develop an approach which, while giving up claims to optimality, would display some *robustness* with respect to various criteria, including

- quality of solutions;
- speed;
- ease of addition of new constraints;
- ease of modification of the objective function (e.g. when incorporating higher moments than the variance, or when considering alternative risk criteria like the semi–variance).

In the next section, we review the basic principles and terminology of the simulated annealing metaheuristic.

3. Simulated annealing

Detailed discussions of simulated annealing can be found in van Laarhoven and Aarts [25], Aarts and Lenstra [1] or in the survey by Pirlot [23]. We only give here a very brief presentation of the method.

Simulated annealing is a generic name for a class of optimization heuristics that perform a stochastic neighborhood search of the solution space. The major advantage of SA over classical local search methods is its ability to avoid getting trapped in local minima while searching for a global minimum. The underlying idea of the heuristic arises from an analogy with certain thermodynamical processes (cooling of a melted solid). Kirkpatrick et al. [14] and Černý [4] pioneered its use for combinatorial problems. For a generic problem of the form

$$\min \quad F(x) \quad \text{s.t.} \quad x \in X,$$

the basic principle of the SA heuristic can be described as follows. Starting from a current solution x, another solution y is generated by taking a stochastic step in some neighborhood of x. If this new proposal improves the value of the objective function, then y replaces x as the new current solution. Otherwise, the new solution y is accepted with a probability that decreases with the magnitude of the deterioration and in the course of iterations. (Notice the difference with classical descent approaches, where only improving moves are allowed and the algorithm may end up quickly in a local optimum.)

More precisely, the generic simulated annealing algorithm performs the following steps.

- Choose an initial solution $x^{(0)}$ and compute the value of the objective function $F(x^{(0)})$. Initialize the incumbent solution (i.e. the best available solution), denoted by (x^*, F^*), as: $(x^*, F^*) \leftarrow (x^{(0)}, F(x^{(0)}))$.
- Until a stopping criterion is fulfilled and for n starting from 0, do:
  - Draw a solution x at random in the neighborhood $V(x^{(n)})$ of $x^{(n)}$.
  - If $F(x) \leqslant F(x^{(n)})$ then $x^{(n+1)} \leftarrow x$ and if $F(x) \leqslant F^*$ then $(x^*, F^*) \leftarrow (x, F(x))$.
  - If $F(x) > F(x^{(n)})$ then draw a number p at random in [0,1] and if $p \leqslant p(n, x, x^{(n)})$ then $x^{(n+1)} \leftarrow x$ else $x^{(n+1)} \leftarrow x^{(n)}$.

The function $p(n, x, x^{(n)})$ is often taken to be a Boltzmann function inspired from thermodynamics models:

$$p(n, x, x^{(n)}) = \exp\left(-\frac{1}{T_n} \Delta F_n\right), \tag{2}$$

where $\Delta F_n = F(x) - F(x^{(n)})$ and T_n is the *temperature* at step n, that is a nonincreasing function of the iteration counter n. In so-called *geometric cooling schedules*, the temperature is kept unchanged during each successive *stage*, where a stage consists of a constant number L of consecutive iterations. After each stage, the temperature is multiplied by a constant factor $\alpha \in (0, 1)$.

Due to the generality of the concepts that it involves, SA can be applied to a wide range of optimization problems. In particular, no specific requirements need to be imposed on the objective function (derivability, convexity, . . .) nor on the solution space. Moreover, it can be shown that the metaheuristic converges asymptotically to a global minimum [25].

From a practical point of view, the approach often yields excellent solutions to hard optimization problems. Surveys and descriptions of applications can be found in van Laarhoven and Aarts [25], Osman and Laporte [21] or Aarts and Lenstra [1].

Most of the original applications of simulated annealing have been made to problems of a combinatorial nature, where the notions of 'step' or 'neighbor' usually find a natural interpretation. Due to the success of simulated annealing in this framework, several researchers have attempted to extend the approach to continuous minimization problems (see [6,9,25,27]). However, few practical applications appear

in the literature. A short list can be found in the previous references, in particular in Osman and Laporte [21].

We are especially interested in these extensions, since portfolio selection typically involves a mix of continuous and discrete variables (see Section 2). One of the aims of our work, therefore, is to gain a better understanding of the difficulties encountered when applying simulated annealing to mixed integer nonlinear optimization problems and to carry out an exploratory investigation of the potentialities offered by SA in this framework.

4. Simulated annealing for portfolio selection

4.1. Generalities: How to handle constraints . . .

In order to apply the SA algorithm to problem (PS), we have to undertake an important tailoring work. Two notions have to be defined in priority, i.e. those of *solution* (or encoding thereof) and *neighborhood*.

We simply encode a solution of (PS) as an *n*-dimensional vector x, where each variable x_i represents the holdings of asset i in the portfolio. The quality of a solution is measured by the variance of the portfolio, that is $x^t C x$.

Now, how do we handle the constraints, that is, how do we make sure that the final solution produced by the SA algorithm satisfies all the constraints of (PS)?

The first and most obvious approach enforces feasibility throughout *all* iterations of the SA algorithm and forbids the consideration of any solution violating the constraints. This implies that the neighborhood of a current solution must entirely consist of feasible solutions. A second approach, by contrast, allows the consideration of infeasible solutions but adds a penalty term to the objective function for each violated constraint: the larger the violation of the constraint, the larger the increase in the value of the objective function. A portfolio which is unacceptable for the investor must be penalized enough to be rejected by the minimization process.

The "all-feasible" vs. "penalty" debate is classical in the optimization literature. In the context of the simplex algorithm, for instance, infeasible solutions are temporarily allowed in the initial phase of the *big-M* method, while feasibility is enforced thereafter by an adequate choice of the variable which is to leave the basis at each iteration. For a discussion of this topic in the framework of local search heuristics, see e.g. the references in Pirlot [23].

Both approaches, however, are not equally convenient in all situations and much of the discussion in the next subsections will center around the "right" choice to make for each class of constraints. Before we get to this discussion, let us first line up the respective advantages and inconvenients of each approach.

When penalties are used, the magnitude of each penalty should depend on the magnitude of the violation of the corresponding constraint, but must also be scaled relatively to the variance of the portfolio. A possible expression for the penalties is

$$a \times |\text{violation}|^p, \tag{3}$$

where a and p are scaling factors. For example, the violation of the return constraint can be represented by the difference between the required portfolio return ($R_{\exp}$) and the current solution return ($R^t x$). The violation of the floor constraint for asset i can be expressed as the difference between the minimum admissible level $\underline{x}_i$ and the current holdings x_i, when this difference is positive.

The first inconvenient of this method is that it searches a solution space whose size may be considerably larger than the size of the feasible region. This process may require many iterations and prohibitive computation time.

The second inconvenient stems from the scaling factors: it may be difficult to define adequate values for a and p. If these values are too small, then the penalties do not play their expected role and the final solution may be infeasible. On the other hand, if a and p are too large, then the term $x^t C x$ becomes negligible with respect to the penalty; thus, small variations of x can lead to large variations of the penalty term, which mask the effect of the variance term.

Clearly, the correct choice of a and p depends on the scale of the data, i.e. on the particular instance at hand! It appears very difficult to automate this choice. For this reason, we use penalties for "soft" constraints only, and when nothing else works.

In our implementations, we have selected values for a and p as follows. First, we let $a = V/\epsilon^p$, where V is the variance of the most risky asset, that is $V = \max_{1 \leqslant i \leqslant n} C_{ii}$, and ϵ is a measure of numerical accuracy. Since the variance of any portfolio lies in the interval $[0, V]$, this choice of a guarantees that every feasible portfolio yields a better value of the objective function than any portfolio which violates a constraint by ϵ or more (but notice that smaller violations are penalized as well). The value of p can now be used to finetune the magnitude of the penalty as a function of the violation: in our experiments, we have set $p = 2$.

Let us now discuss the alternative, all-feasible approach, in which the neighborhood of the current solution may only contain solutions that satisfy the given subset of constraints. The idea that we implemented here (following some of the proposals made in the literature on stochastic global optimization) is to draw a direction at random and to take a small step in this direction away from the current solution. The important features of such a move is that both its *direction* and *length* are computed so as to respect the constraints. Moreover, the holdings of only a few assets are changed during the move, meaning that the feasible direction is chosen in a low-dimensional subspace. This simplifies computations and provides an immediate translation of the concept of "neighbor".

The main advantage of this approach is that no time is lost investigating infeasible solutions. The main disadvantage is that it is not always easy to select a neighbor in this way, so that the resulting moves may be quite contrived, their computation may be expensive and the search process may become inflexible. On the other hand, this approach seems to be the only reasonable one for certain constraints, like for example the trading constraints.

For each class of constraints, we had to ponder the advantages and disadvantages of each approach. When a constraint must be strictly satisfied or when it is possible to enforce it efficiently without penalties in the objective function, then we do so. This is the case for the constraints on budget, return and maximum number of assets. A mixed approach is used for the trading, floor, ceiling and turnover constraints.

In the next sections, we successively consider each class of constraints, starting with those that are enforced without penalties.

4.2. Budget and return constraints

4.2.1. Basic principle

The budget constraint must be strictly satisfied, since its unique goal is to norm the solution. Therefore, it is difficult to implement this constraint through penalties.

The same conclusion applies to the return constraint, albeit for different reasons. Indeed, our aim is to compute the whole mean–variance frontier. To achieve this aim, we want to let the expected portfolio return vary uniformly in its feasible range and to determine the optimal risk associated with each return. In order to obtain meaningful results, the optimal portfolio computed by the procedure should have the exact required return. In our experience, the approach relying on penalties was completely inadequate for this purpose.

In view of these comments, we decided to restrict our algorithm to the consideration of solutions that strictly satisfy the return and the budget constraints. More precisely, given a portfolio x, the neighborhood of x contains all solutions x' with the following property: there exist three assets, labeled 1, 2 and 3 without loss of generality, such that

$$\begin{cases} x_1' = x_1 - step, \\ x_2' = x_2 + step * (R_1 - R_3)/(R_2 - R_3), \\ x_3' = x_3 + step * (R_2 - R_1)/(R_2 - R_3), \\ x_i' = x_i, \quad \text{for all } i > 3, \end{cases} \tag{4}$$

where *step* is a (small) number to be further specified below. It is straightforward to check that x' satisfies the return and budget constraints when x does so. Geometrically, all neighbors x' of the form (4) lie on a line passing through x and whose direction is defined by the intersection of the 3-dimensional subspace associated to assets 1, 2 and 3 with the two hyperplanes associated to the budget constraint and the return constraint, respectively. Thus, the choice of three assets determines the direction of the move, while the value of *step* determines its amplitude.

Observe that, in order to start the local search procedure, it is easy to compute an initial solution which satisfies the budget and return constraints. Indeed, if x denotes an arbitrary portfolio and min (resp. max) is the subscript of the asset with minimum (resp. maximum) expected return, then a feasible solution is obtained upon replacing $x_{\min}$ and $x_{\max}$ by $x_{\min}'$ and $x_{\max}'$, where

$$\begin{cases} x_{\min}' = [R_{\exp} - \sum_{i \neq \min,\max}^{n} x_i R_i - (x_{\min} + x_{\max}) R_{\max}]/(R_{\min} - R_{\max}), \\ x_{\max}' = x_{\min} + x_{\max} - x_{\min}'. \end{cases} \tag{5}$$

The resulting solution may violate some of the additional constraints of the problem (trading, turnover, etc.) and penalties will need to be introduced in order to cope with this difficulty. This point will be discussed in sections to come.

4.2.2. Direction of moves

Choosing a neighbor of x, as described by (4), involves choosing the direction of the move, i.e. choosing three assets whose holdings are to be modified. In our initial attempts, we simply drew the indices of these assets randomly and uniformly over $\{1, \ldots, n\}$. Many of the corresponding moves, however, were nonimproving, thus resulting in slow convergence of the algorithm.

We have been able to improve this situation by guiding the choice of the three assets to be modified. Observe that the assets whose return is closest to the required portfolio return have (intuitively) a higher probability to appear in the optimal portfolio than the remaining ones. (This is most obvious for portfolios with "extreme" returns: consider for example the case where we impose nonnegative holdings and we want to achieve the highest possible return, i.e. $R_{\max}$.)

To account for this phenomenon, we initially sort all the assets by nondecreasing return. For each required portfolio return $R_{\exp}$, we determine the asset whose return is closest to $R_{\exp}$ and we store its position, say q, in the sorted list. At each iteration of the SA algorithm, we choose the first asset to be modified by computing a random number normally distributed with mean q and with standard deviation large enough to cover all the list: this random number points to the position of the first asset in the ordered list. The second and third assets are then chosen uniformly at random.

4.2.3. Amplitude of moves

Let us now turn to the choice of the *step* parameter in (4). In our early attempts, *step* was fixed at a small constant value (so as to explore the solution space with high precision). The results appeared reasonably good but required extensive computation time (as compared to later implementations and to the quadratic simplex method, when this method was applicable).

In order to improve the behavior the algorithm, it is useful to realize that, even if a small value of *step* necessarily produces a small modification of the holdings of the first asset, it is more difficult to predict its

effect on the other assets (see (4)). This may result in poorly controlled moves, whose amplitude may vary erratically from one iteration to the next.

As a remedy, we chose to construct a ball around each current solution and to restrict all neighbors to lie on the surface of this ball (this is inspired by several techniques for random sampling and global optimization; see e.g. Lovász and Simonovits [18] or Zabinsky et al. [27]). The euclidean length of each move is now simply determined by the radius of the ball. Furthermore, in view of Eqs. (4), *step* is connected to the radius by the relation:

$$step = \pm \frac{radius \; * \; (R_2 - R_3)}{\sqrt{(R_2 - R_3)^2 + (R_1 - R_3)^2 + (R_2 - R_1)^2}}, \tag{6}$$

where the $\pm$ sign can be picked arbitrarily (we fix it randomly).

Now, how should we choose the radius of the ball? On the one hand, we want this value to be relatively small, so as to achieve sufficient precision. On the other hand, we can play with this parameter in order to enforce some of the constraints which have not been explicitly considered yet (floor, ceiling, trading, etc.). Therefore, we will come back to a discussion of this point in subsequent sections.

4.3. Maximum number of assets constraint

This cardinality constraint is combinatorial in nature. Moreover, a "natural" penalty approach based on measuring the extent of the violation:

$$violation = |\{i \in \{1, \ldots, n\} : x_i \neq 0\}| - N$$

(see model (PS)) does not seem appropriate to handle this constraint: indeed, all the neighbors of a solution are likely to yield the same penalty, except when an asset exceptionally appears in or disappears from the portfolio. Other types of penalties could conceivably be considered in order to circumvent the difficulty caused by this "flat landscape" (see e.g. [13,23] for a discussion of similar issues arising in graph coloring or partitioning problems). We rather elected to rely on an all-feasible approach, whereby we restrict the choice of the three assets whose holdings are to be modified, in such a way as to maintain feasibility at every iteration. Let us now proceed with a case-by-case discussion of this approach.

First, observe that the initial portfolio only involves two assets (see Section 4.2.1) and hence is always feasible with respect to the cardinality constraint (we disregard the trivial case where $N = 1$).

Now, if the current portfolio involves $N - k$ assets, with $k \geqslant 1$, then we simply make sure, as we draw the three assets to be modified, that at most k of them are not already in the current portfolio. This ensures that the new portfolio involves at most N assets.

The same logic, however, cannot be used innocuously when the current portfolio involves exactly N assets: indeed, this would lead to a rigid procedure whereby no new asset would ever be allowed into the portfolio, unless one of the current N assets disappears from the portfolio by pure chance (that is, as the result of numerical cancelations in Eqs. (4)). Therefore, in this case, we proceed as follows. We draw three assets at random, say assets 1, 2 and 3, in such a way that at most one of them is not in the current portfolio. If all three assets already are in the portfolio, then we simply determine the new neighbor as usual. Otherwise, assume for instance that assets 1 and 2 are in the current portfolio, but asset 3 is not. Then, we set the parameter *step* equal to x_1 in Eqs. (4). In this way, $x'_1 = 0$ in the neighbor solution: the move from x to x' can be viewed as substituting asset 1 by asset 3 and rebalancing the portfolio through appropriate choices of the holdings x'_2 and x'_3.

4.4. Floor, ceiling and turnover constraints

The floor, ceiling and turnover constraints are similar to each other, since each of them simply defines a minimum or maximum bound on holdings. Therefore, our program automatically converts all turnover purchase constraints into ceiling constraints and all turnover sales constraints into floor constraints.

Suppose now that we know which three assets (say, 1, 2 and 3) must be modified at the current move from solution x to solution x', and suppose that the amplitude of the move has not been determined by the cardinality constraint (see previous subsection). Then, it is easy to determine conditions on the value of *step* such that x' satisfies the floor and ceiling constraints. Indeed, combining the latter constraints with Eqs. (4) leads to the following conditions:

$$\begin{cases} x_1 - \bar{x}_1 \leqslant step \leqslant x_1 - \underline{x}_1, \\ \underline{x}_2 - x_2 \leqslant step * (R_1 - R_3)/(R_2 - R_3) \leqslant \bar{x}_2 - x_2, \\ \underline{x}_3 - x_3 \leqslant step * (R_2 - R_1)/(R_2 - R_3) \leqslant \bar{x}_3 - x_3. \end{cases} \tag{7}$$

These conditions yield a feasible interval of variation for *step* and hence (via Eq. (6)) for the radius of the ball limiting the move from x to x'. We denote by $[lb, ub]$ the feasible interval for *radius*.

Let us first assume that the interval $[lb, ub]$ is non-empty (and, for practical purposes, not too "small"). Then, different strategies are applicable. We could start a linesearch optimization process to find the optimal value of the radius in $[lb, ub]$ (i.e. the value of the radius leading to the best neighbor x'). We have not experimented with this approach and have rather implemented a simpler option. We initially pick a small positive constant ρ. If, at any iteration, ρ is an admissible value for the radius, i.e. if $\rho \in [lb, ub]$, then we set the radius of the ball equal to ρ. Otherwise, if ρ is larger than ub (resp. smaller than lb), then we set the radius equal to ub (resp. lb).

Actually, in practice, we do not work with a single value of ρ but with two values, say $\rho_1 > \rho_2$. The larger value ρ_1 is used at the beginning of the algorithm, so as to accelerate the exploration of the solution space. In a latter phase, i.e. when improving moves can no longer be found, the radius is decreased to the smaller value ρ_2 in order to facilitate convergence to a local optimum.

Let us now consider the case where the feasibility interval $[lb, ub]$ is either empty or very narrow, meaning that x is either infeasible or close to the infeasible region. In this case, we disregard conditions (7) and simply set the radius of the ball equal to ρ, thus generating an infeasible solution x'. In order to handle this and other situations where infeasible solutions arise (see e.g. the end of Section 4.2.1), we introduce a penalty term of the form $a \times |\text{violation}|^p$ in the objective function for each ceiling or floor constraint, as discussed in Section 4.1 and further specified in Table 1. Notice that the penalty approach appears to be suitable here, since limited violations of the floor, ceiling or turnover constraints can usually be tolerated in practice.

4.5. Trading constraints

The trading constraints are disjunctive: either the holdings of each asset remain at their current value $x^{(0)}$ or they are modified by a minimum admissible amount. These constraints are difficult to handle, as they disconnect the solution space into 3^n feasible subregions separated by forbidden subsets.

We use a similar approach as for the previous class of constraints. Denote by $x' = x + rd$ the neighbor of x obtained as explained in Section 4.4, where d is the direction of the move and r is the radius of the ball. If

Table 1
Penalties for floor and ceiling constraints

Ceiling	if $x_i > \bar{x}_i$, then penalty $= a(x_i - \bar{x}_i)^p$
Floor	if $x_i < \underline{x}_i$, then penalty $= a(\underline{x}_i - x_i)^p$

Table 2
Penalties for trading constraints

Purchase	$Q_{\text{purchase}} = x_i - x_i^{(0)}$; if $Q_{\text{purchase}} \in]0, \underline{B}_i[$, then
	{if $Q_{\text{purchase}} \leqslant \underline{B}_i/2$ then penalty $= a\, Q_{\text{purchase}}^p$
	else penalty $= a(\underline{B}_i - Q_{\text{purchase}})^p$}
Sale	$Q_{\text{sale}} = x_i^{(0)} - x_i$; if $Q_{\text{sale}} \in]0, \underline{S}_i[$, then
	{if $Q_{\text{sale}} \leqslant \underline{S}_i/2$ then penalty $= a\, Q_{\text{sale}}^p$
	else penalty $= a(\underline{S}_i - Q_{\text{sale}})^p$}

x' satisfies the trading constraints, then there is nothing to be done. Otherwise, we temporarily disregard the floor/ceiling constraints (which are anyway easier to enforce than the trading constraints) and we compute the smallest value t in the interval $[r, \infty)$ such that $x'' = x + td$ satisfies the trading constraints. If t is not too large (i.e., if t does not exceed a predetermined threshold), then we retain x'' as neighbor of x. On the other hand, if t is larger than the threshold, then we reject the current move and we draw three new assets to be modified.

Observe, however, that solutions which violate the trading constraints still arise in some iterations of the algorithm. For instance, the initial solution is usually infeasible, and so are the solutions which are generated when the portfolio contains exactly N assets (see the last paragraph of Section 4.3). Such infeasibilities are penalized as described in Table 2 (the parameters a and p are fixed as in Section 4.1). Observe that penalties are high at the center of the forbidden zones and decrease in the direction of admissible boundaries (associated with no trading or with minimum sales/purchases). Therefore, starting from a forbidden portfolio, the process tends to favor moves toward feasible regions.

4.6. Summary: Neighbor selection

We can summarize as follows the neighbor selection procedure.

Move direction

- If the current portfolio involves $N - k$ assets, with $k \geqslant 1$, then
  - select three assets, say 1, 2 and 3, at random as explained in Section 4.2.2, while ensuring that at most k of them are outside the current portfolio (see Section 4.3);
  - go to Case a.
- If the current portfolio involves N assets, then
  - select three assets, say 1, 2 and 3, at random as explained in Section 4.2.2, while ensuring that at most one of them is outside the current portfolio (see Section 4.3);
  - if all three selected assets are in the current portfolio, then go to Case a; else go to Case b.

Step length
Case a

- Let d be the direction of the move as defined by Eqs. (4) (with the sign of *step* fixed at random). Compute the feasible interval for the radius of the move, say $[lb, ub]$, and compute the radius r as explained in Section 4.4.
- If $x + rd$ satisfies the trading constraint, then
  - $x + rd$ is the selected neighbor; if necessary, compute penalties for the violation of the floor and ceiling constraints as in Table 1;
  - else, try to extend the move to $x + td$, as explained in Section 4.5; if t is not too large, then

    – $x + td$ is the selected neighbor; if necessary, compute penalties for the violation of the floor and ceiling constraints as in Table 1;
    – else, discard direction d and select a new direction.

Case b

- Let assets 1 and 2 be in the current portfolio and asset 3 be outside. In Eqs. (4), set the parameter *step* equal to x_1, set $x'_1 = 0$ and compute the corresponding values of x'_2 and x'_3.
- If necessary, compute penalties for the violation of the floor, ceiling and trading constraints as in Tables 1 and 2.

5. Cooling schedule, stopping criterion and intensification

5.1. Cooling schedule and stopping criterion

In our implementation of simulated annealing, we have adopted the geometric cooling schedule defined in Section 3. In order to describe more completely this cooling schedule, we need to specify the value of the parameters T_0 (the *initial temperature*), L and α.

Following the recommendations of many authors (see e.g. [1,13,23]), we set the initial temperature T_0 in such a way that, during the first cooling stage (first L steps), the probability of acceptance of a move is roughly equal to a predetermined, relatively high value χ_0 (in our numerical tests, $\chi_0 = 0.8$). In order to achieve this goal, we proceed as follows. In a preliminary phase, the SA algorithm is run for L steps without rejecting any moves. The average increase of the objective function over this phase, say Δ, is computed and T_0 is set equal to

$$T_0 = \frac{-\Delta}{\ln \chi_0} \tag{8}$$

(see Eq. (2)).

After L moves, the temperature is decreased according to the scheme $T_{k+1} = \alpha T_k$. We use here the standard value $\alpha = 0.95$.

The fundamental trade-offs involved in the determination of the stage length L are well-known, but difficult to quantify precisely. A large value of L allows to explore the solution space thoroughly, but results in long execution times. Some studies (see [13,23]) suggest to select a value of L roughly equal to the neighborhood size. In our algorithm, this rule leads to the value $L \approx \binom{n}{3}$, which appeared excessively large in our computational tests. Therefore, we eventually settled for values of L of the same order of magnitude as n (e.g. we let $L = 300$ when $n = 150$). We elaborate on this topic in the next section.

In this first, basic version, the algorithm terminates if no moves are accepted during a given number S of successive stages. In our experiments, we used $S = 5$.

5.2. Intensification

We have experimented with several ways of improving the quality of the solutions computed by the SA algorithm (at the cost of its running time). In all these attempts, the underlying strategy simply consists in running several times the algorithm described above; in this framework, we call *cycle* each execution of the basic algorithm. The main difference between the various strategies is found in the initialization process of each cycle. Namely, we have tried to favor the exploration of certain regions of the solution space by restarting different cycles from "promising" solutions encountered in previous cycles. Such *intensification* strategies have proved successful in earlier implementations of local search metaheuristics.

Strategy 1. In this naive strategy, we run several (say M) cycles successively and independently of each other, always from the same initial solution. The random nature of each cycle implies that this strategy may perform better than the single-start version.

Strategy 2. This second strategy is very similar to Strategy 1, except that each cycle uses, as initial solution, the best solution found within the previous cycle. For all but the first cycle, the initial temperature is defined by formula (8) with $\chi_0 = 0.3$ (the idea is to set the initial temperature relatively low, so as to preserve the features of the start solution). Also, for all but the last cycle, the stopping criterion is slightly relaxed: namely, each intermediate cycle terminates after S' stages without accepted moves, where $S' < S$. In our experiments, $S' = 2$.

Strategy 3. In the first cycle, we constitute a list P of promising solutions, where a promising solution may be

(a) either the best solution found during each stage,
(b) or the best solution found in any stage where the objective function has dropped significantly (a drop is significant if it exceeds the average decrease of the objective function during the previous stage).

Next, we perform $|P|$ additional cycles, where each cycle starts from one of the solutions in P. For these additional cycles, the initial temperature is computed with $\chi_0 = 0.3$ and the cycle terminates after $S' = 2$ stages without accepted moves.

In the next section, we will compare the results produced by the basic strategy (Strategy 0), Strategy 1, Strategy 2 and Strategy 3b. In order to allow meaningful comparisons between the three multi-start strategies, we restrict the number of cycles performed by Strategy 3b by setting an upper-bound ($M - 1$) on the size of the list P. That is, after completion of the first cycle, we discard solutions from P by applying the following two rules in succession:

- if several solutions in P imply the same trades (i.e., all these solutions recommend to buy or to sell *exactly* the same securities), then we only keep one of these solutions; the rationale is here that our algorithm is rather good at finding the best solution complying with any given trading rules, so that all of these solutions yield 'equivalent' starting points;
- we only keep the best ($M - 1$) solutions in P.

6. Computational experiments

6.1. Environment and data

The algorithms described above have been implemented in standard C (ANSI C) and run on a PC Pentium 450 MHz under Windows 98. A graphical interface was developed with Borland C++ Builder. All computation times mentioned in coming sections are approximate real times, not CPU times. Unless otherwise stated, the parameter settings for the basic SA algorithm are defined as follows:

- stage size: $L = 2n$;
- stopping criterion: terminate when no moves are accepted for $S = 5$ consecutive stages;
- ball radius: $\rho_1 = 0.005$, decreased to $\rho_2 = 0.001$ as soon as fewer than 10 steps are accepted during a whole stage.

For the sake of constructing realistic problem instances, we have used financial data extracted from the DataStream database. We have retrieved the weekly prices of $n = 151$ US stocks covering different traditional sectors for 484 weeks, from 6 January 1988 to 9 April 1997, in order to estimate their mean returns and covariance matrix. The stocks were drawn at random from a subpopulation involving mostly major stocks. (Note that our goal was not to draw any conclusions regarding the firms, or the stock market, or even the composition of optimal portfolios, but only to test the computational performance of the algorithms.) These data have been used to generate several instances of model (PS) involving different subsets of constraints.

For each instance, we have approximately computed the mean-variance frontier by letting the expected portfolio return (R_{exp}) vary from −0.3688% to 0.737% by steps of 0.01% (110 portfolios). Linear interpolation is used to graph intermediate values. In each graph, the ordinate represents the expected portfolio return (expressed in basis points) and the abscissa represents the standard deviation of return.

We now discuss different instances in increasing order of complexity.

6.2. The Markowitz mean–variance model

As a first base case, we have used the simulated annealing (SA) algorithm to solve instances of the Markowitz mean–variance model (see Section 2.2.1) without nonnegativity constraints. Since these instances can easily be solved to optimality by Lagrangian techniques, we are able to check the quality of the solutions obtained by the SA algorithm. Our algorithm finds the exact optimal risk for all values of the expected return. The SA algorithm requires 2 or 3 seconds per portfolio of 151 securities with the standard parameter settings.

The mean–risk frontier for this instance is plotted in Fig. 1. It will also be displayed in all subsequent figures, in order to provide a comparison with the frontiers obtained for constrained problems.
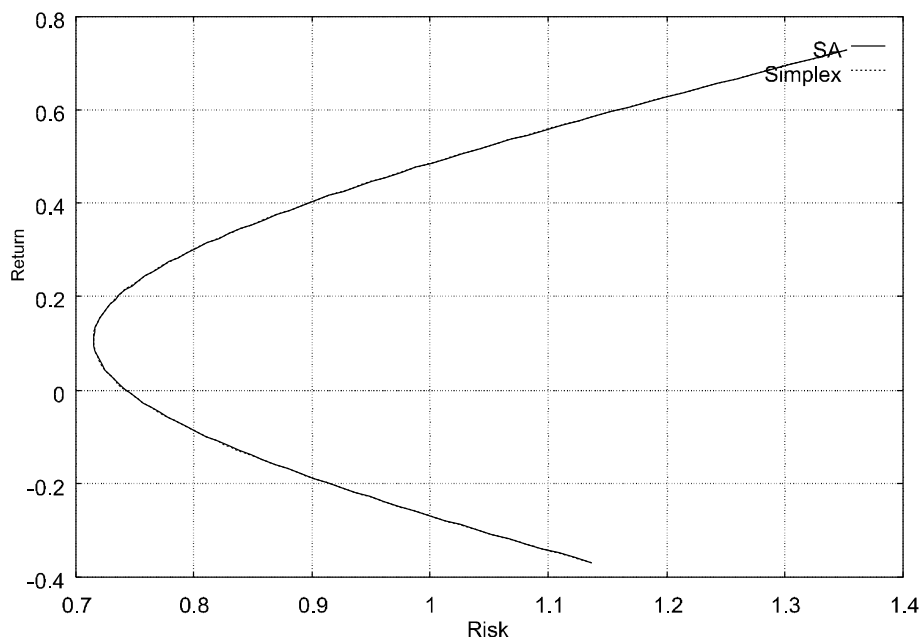
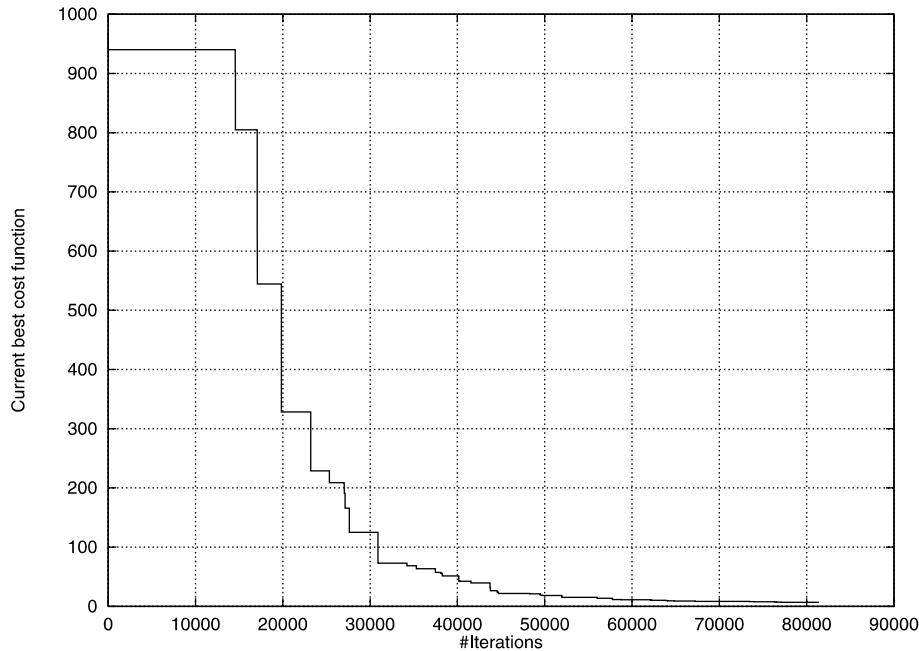Fig. 1. Mean–risk frontier with short sales.

Fig. 2. Evolution of the portfolio variance.

For a particular value of the target return, Fig. 2 illustrates the evolution of the portfolio variance in the course of iterations.

6.3. Floor, ceiling and turnover constraints

We solved several instances involving floor, ceiling and turnover constraints. The first instance (Fig. 3) imposes nonnegativity constraints on all assets (no short sales). The second one (Fig. 4) adds more restrictions on minimal and maximal holdings allowed: $\underline{x}_i = 0$ and $\bar{x}_i = 0.2$ for each security. The hypotheses and results are more completely described next to each figure.

Here again, the exact optimal solution can be computed efficiently (e.g. using Wolfe's quadratic simplex algorithm [26]) and can be used to validate the results delivered by the SA algorithm. The quality of the heuristic solutions is usually extremely good. Slight deviations from optimality are only observed for extreme portfolio returns. Moreover, the solutions always satisfy all the constraints (penalties vanish). Run times are short and competitive with those of the quadratic simplex method (less than 4 seconds per portfolio).

6.4. Trading constraints

When the model only involves floor, ceiling and turnover constraints, the mean–variance frontiers are smooth curves. When we introduce trading constraints, however, sharp discontinuities may arise. This is vividly illustrated by Fig. 5. Here, we have selected three securities and we have plotted all (mean return, risk)-pairs corresponding to feasible portfolios of these three securities. Observe that disconnected regions appear. (Similar observations are made by Chang et al. in [5].) Fig. 6 shows the outcome provided by the
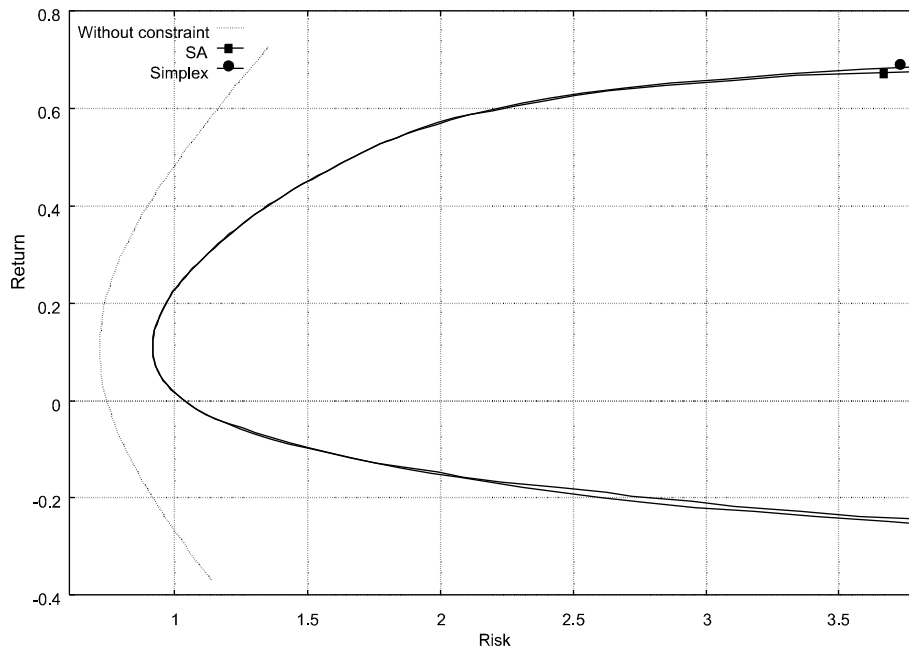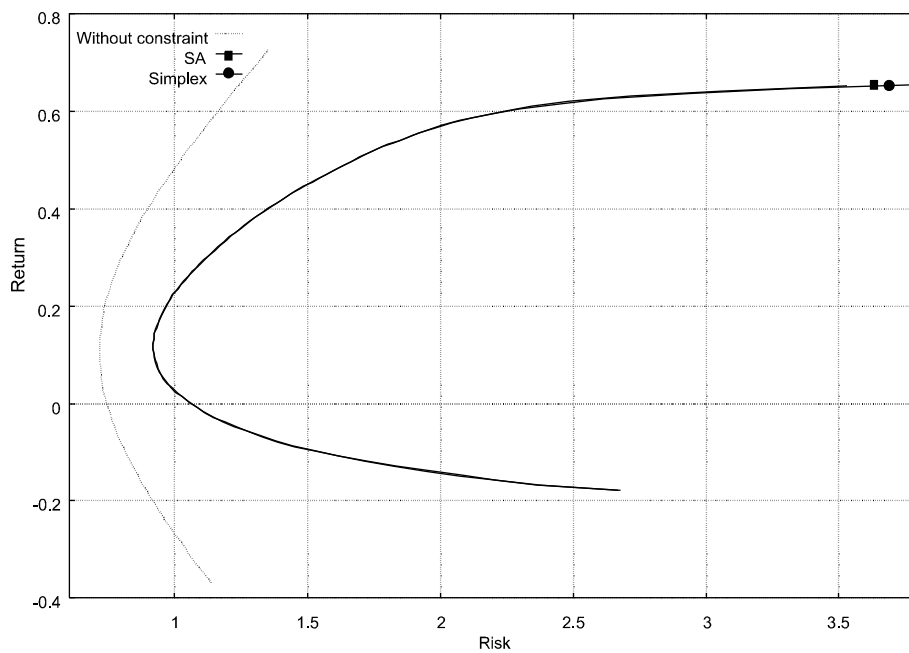
Fig. 3. Stop if no accepted move for $5L$ iterations; #equities n, 151; #portfolios, 110; $L = 2n$; $\underline{x}_i = 0 \ \forall i$; $\bar{x}_i = 1 \ \forall i$; $\rho_1 = 0.005$; $\rho_2 = 0.001$; $\rho_1 \rightarrow \rho_2 : \leqslant 10$ moves, time < 4 seconds per portfolio.
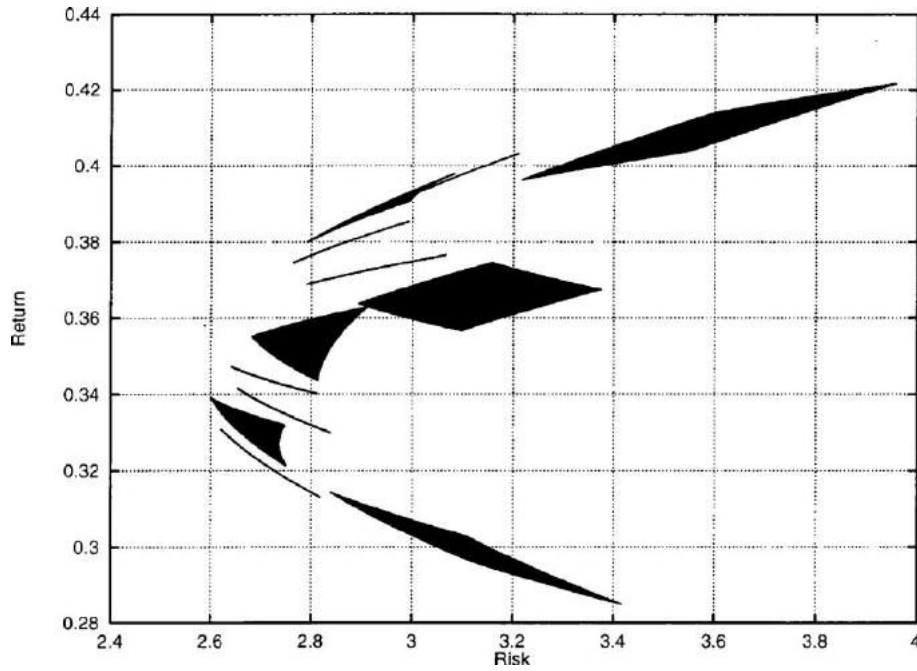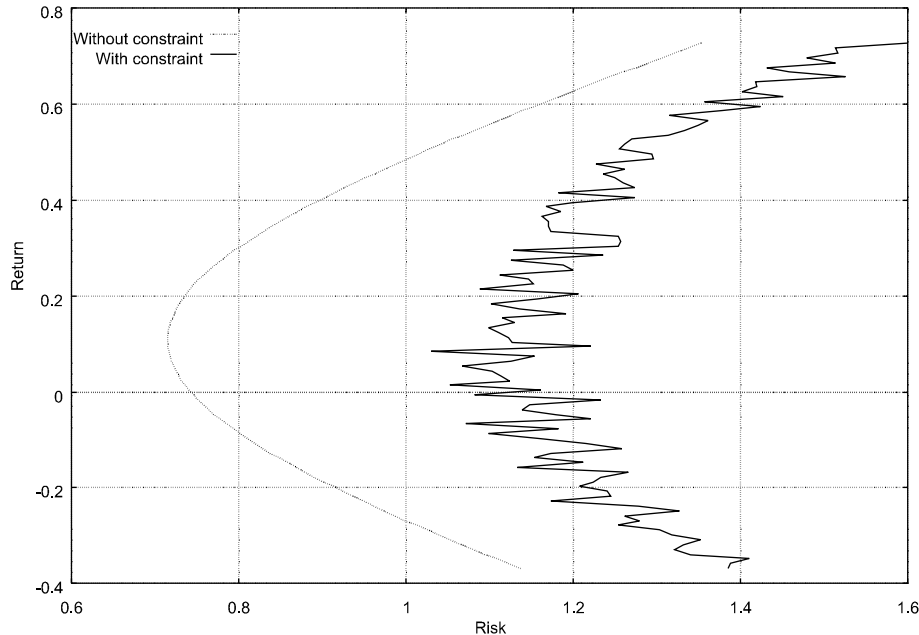

Fig. 4. Stop if no accepted move for $5L$ iterations; #equities n, 151; #portfolios, 110; $L = 2n$; $\underline{x}_i = 0 \ \forall i$; $\bar{x}_i = 0.2 \ \forall i$; $\rho_1 = 0.005$; $\rho_2 = 0.001$; $\rho_1 \rightarrow \rho_2 : \leqslant 10$ moves; time < 3 seconds per portfolio.

Fig. 5. All portfolios: #equities, 3; $x_i^{(0)} = 1/n$; $\underline{B}_i = 0.1$.
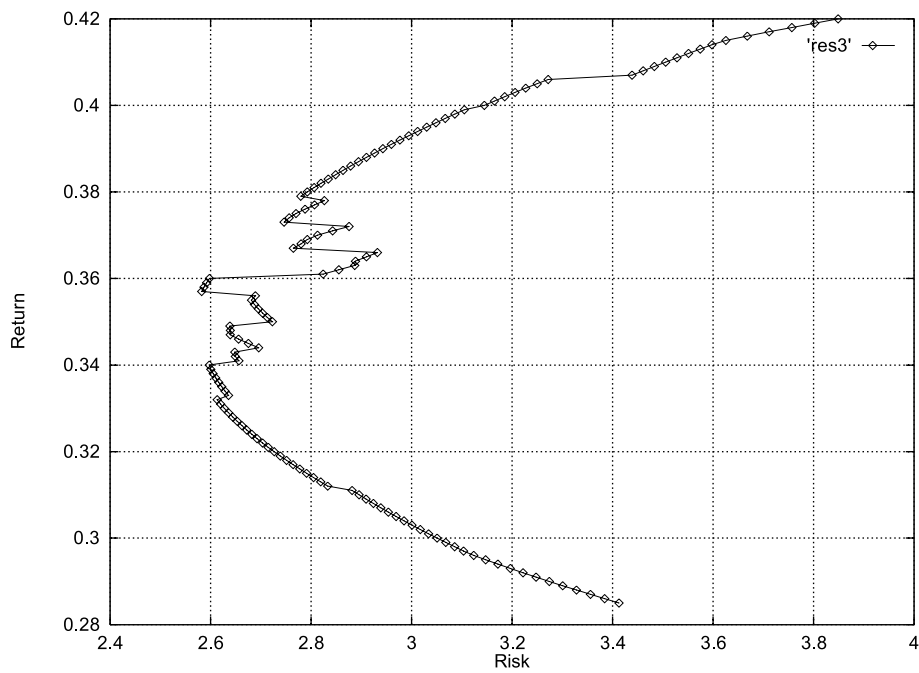
Fig. 6. SA frontier: #equities, 3; $x_i^{(0)} = 1/n$; $\underline{B}_i = 0.1$; $\underline{S}_i = 0.2$; $\underline{x}_i = 0$.

Fig. 7. Stop if no accepted move for 5L iterations; #equities n, 151; #portfolios, 110; $L = 2n$; $x_i^{(0)}|N = 20$; $\underline{B}_i = 0.05 \; \forall i$; $\underline{S}_i = 0.05 \; \forall i$; $\rho_1 = 0.005$; $\rho_2 = 0.001$; $\rho_1 \rightarrow \rho_2 : \leqslant 10$ moves; time, 10 seconds per portfolio.

simulated annealing algorithm: notice that the algorithm perfectly computes the mean–variance frontier for this small example.

When the number of securities increases, the optimization problem becomes extremely difficult to solve. Fig. 7 illustrate the results produced by the basic SA algorithm for the whole set of 151 securities, with trading constraints defined as follows:

- $\underline{B}_i = \underline{S}_i = 0.05 \; (i = 1, \ldots, n)$;
- the initial portfolio $x^{(0)}$ is the best portfolio of 20 stocks with an expected return of 0.24% (see Section 6.5 hereunder).

The computation times remain reasonable (about 10 seconds per portfolio). However, as expected, the frontier is not as smooth as in the simpler cases. The question is to know whether we succeeded in computing the actual frontier or whether the SA algorithm erred in this complex case. The simplex method cannot be used anymore to compute the optimal solutions, because of the mixed integer constraints. Therefore, we have carried out some additional experiments in order to better assess the performance of our algorithm.

First, we have used the commercial package LINGO in order to model and to solve a small instance of the problem. Indeed, LINGO allows to handle nonlinear programming problems involving both continuous and binary variables and to solve such problems to optimality by branch-and-bound. Computation times, however, increase sharply with the size of problem instances. We have therefore restricted the set of underlying assets to 30 equities, with $N = 5$. A visual comparison between the results obtained by the SA algorithm and by LINGO is provided in Fig. 8. We observe that the SA algorithm obtains near optimal solutions for all target returns, within short computational times.
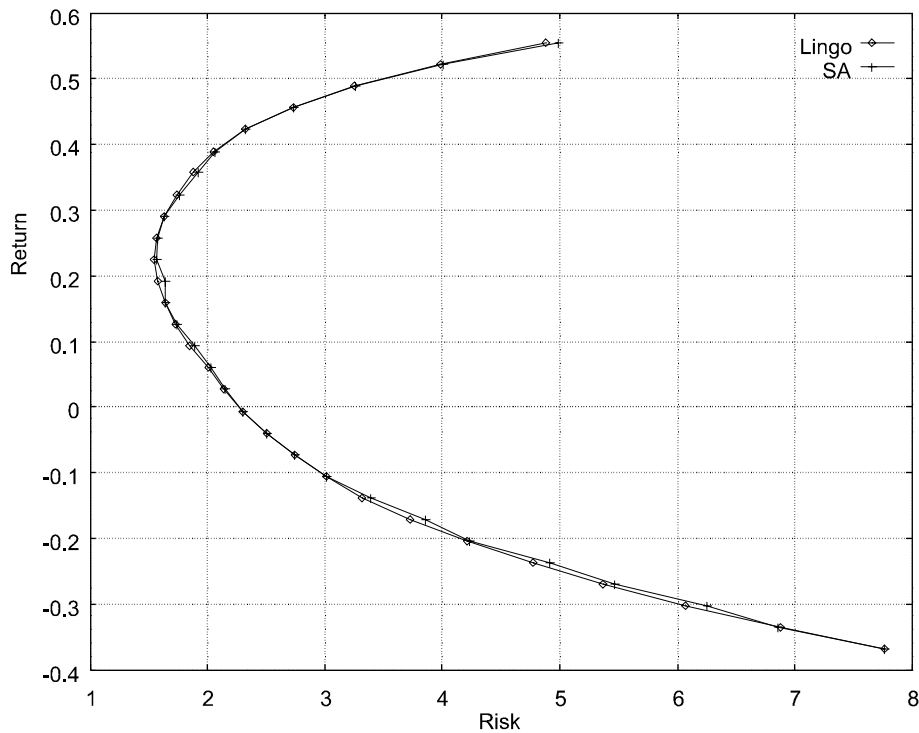
Fig. 8. Stop if no accepted move for $5L$ iterations; #equities n, 30; #portfolios, 30; $L = 3n$; $\underline{x}_i = 0$ $\forall i$; $\bar{x}_i = 1$ $\forall i$; $x_i^{(0)} | N = 5$; $\underline{B}_i = 0.1$ $\forall i$; $\rho_1 = 0.005$; $\rho_2 = 0.001$; $\rho_1 \rightarrow \rho_2 : \leqslant 10$ moves; time < 0.5 seconds per portfolio.

For another test, we have run new experiments on the full data set of 151 equities, using now the three intensification strategies described in the previous section. Globally, intensification tends to improve the results obtained by the basic algorithm (see Fig. 9). However, there is no clear dominance between the three strategies tested. This is rather disappointing, if one remembers that Strategy 1 simply consists in running several times the SA algorithm from the same initial solution. This observation suggests that increased running time, allowing for more exploration of the solution space, maybe the key element in improving the performance of the SA algorithm. (Notice that similar conclusions have been drawn by other authors working with simulated annealing algorithms; see e.g. [1] or [13].) On the other hand, restarting the process from promising solutions does not appear to help much (probably because the features of these solutions are lost in the high-temperature phase of the SA algorithm).

In order to confirm these tentative conclusions, we have run again the basic SA algorithm on the same instances without intensification, but with much larger values of the stage length L, i.e. with $L = n^2$ and $L = n^3$ (see Section 5.1). The stopping criterion is adapted for $L = n^2$ to make sure that is it at least as strict as for $L = n^3$. In this way, we ensure that the improvement obtained for $L = n^3$ is due to the increase of L and not to the stopping criterion. The results of this experiment are displayed in Fig. 10. On the average, over the whole range of target values, the standard deviation of the portfolio improves by 7% when $L = n^2$ and by 13% when $L = n^3$. The largest improvements are attained for intermediate values of the target return. It should be mentioned, however, that such improvements come at the expense of extremely long running times (about 8 hours per portfolio when $L = n^3$).
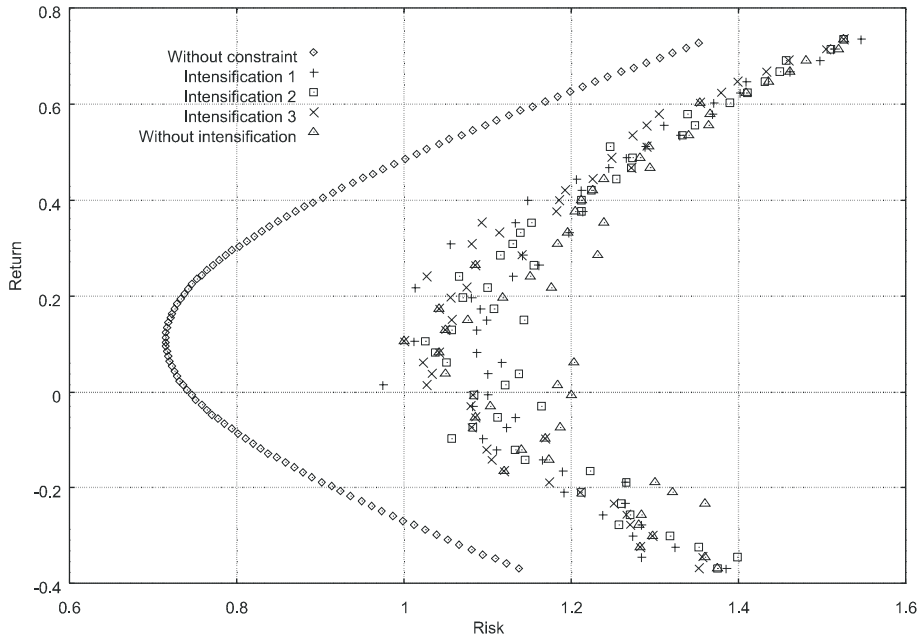
Fig. 9. Stop if no accepted move for $5L$ iterations; #equities n, 151; #portfolios, 50; $L = 2n$; $x_i^{(0)}|N = 20$; $\underline{B}_i = 0.05 \; \forall i$; $\underline{S}_i = 0.05 \; \forall i$; $\rho_1 = 0.005$; $\rho_2 = 0.001$; $\rho_1 \rightarrow \rho_2 : \; \leqslant 10$ moves; time: (Intens1) 35 seconds per portfolio; (Intens2) 34 seconds per portfolio; (Intens3) 49 seconds per portfolio.
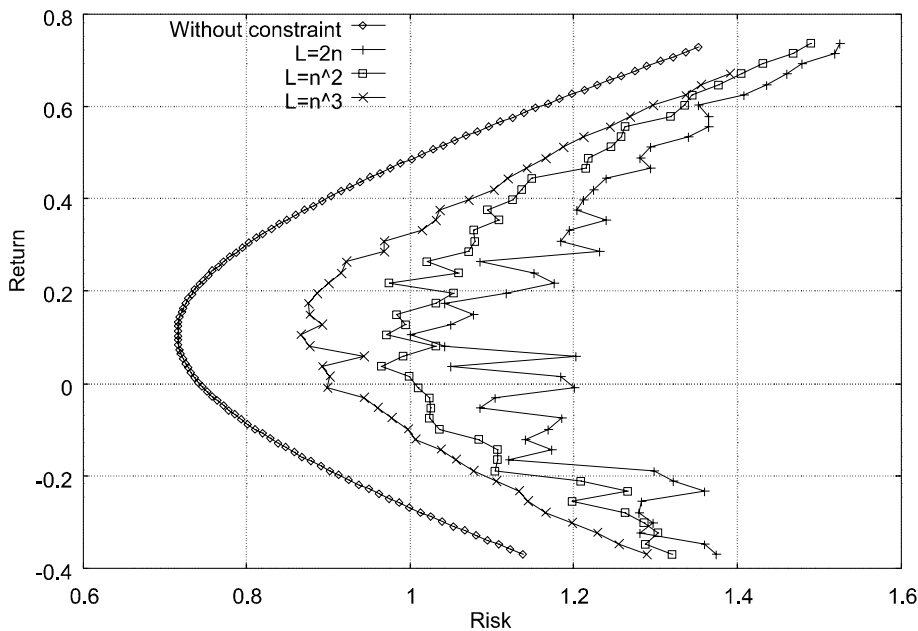

Fig. 10. $L = 2n$ or n^2 or n^3; stop if no accepted move, $5L$ iterations when $L = 2n$; $3nL$ iterations when $L = n^2$; $2L$ iterations when $L = n^3$; #equities n, 151; #portfolios, 50; $x_i^{(0)}|N = 20$; $\underline{B}_i = 0.05 \; \forall i$; $\underline{S}_i = 0.05 \; \forall i$; $\rho_1 = 0.005$; $\rho_2 = 0.001$; $\rho_1 \rightarrow \rho_2 : \; \leqslant 10$ moves; $L = 2n :$ 10 seconds per portfolio; $L = n^2 :$ 23 minutes 42 seconds per portfolio; $L = n^3 :$ 7 hours 58 minutes per portfolio.

6.5. Maximum number of securities

Let us now consider a cardinality constraint limiting the number of assets to be included in the portfolio. Fig. 11 displays the results obtained with the basic SA algorithm when we only allow $N = 20$ assets in the portfolio (without any other constraints in the model, besides the return and budget constraints).

In spite of the combinatorial nature of the cardinality restriction, the computation of the mean-variance frontier is rather efficient for this problem. The solutions obtained by the SA algorithm are always feasible (i.e. no penalties remain when the algorithm terminates). Moreover, the graph in Fig. 11 for $L = 2n$ is very smooth: this suggests that the SA algorithm may have achieved near-optimal solutions for all values of the target returns. In order to validate this hypothesis, we performed some experiments with larger stage lengths ($L = n^2$), and were only able to record minor improvements. We also ran LINGO on a subsample of 30 assets, with $N = 5$; for this smaller instance, the SA algorithm perfectly computed the whole mean–variance frontier (Fig. 12).

6.6. Complete model

Investigating each class of constraint separately was important in order to understand the behavior of the algorithm, but our final aim was to develop an approach that could handle more realistic situations where all the constraints are simultaneously imposed. Fig. 13 illustrates the results obtained by the basic SA algorithm with $L = 2n$ or $L = n^2$ for such a complex instance. Observe that, here again, the results obtained when $L = n^2$ are significantly better than when $L = 2n$. Even with the higher value of L, however, the computation time remains reasonably low.

Fig. 14 sums up all the previous results. It illustrates the effect of each class of constraints on the problem and allows some comparison of the mean–variance frontiers computed in each case.
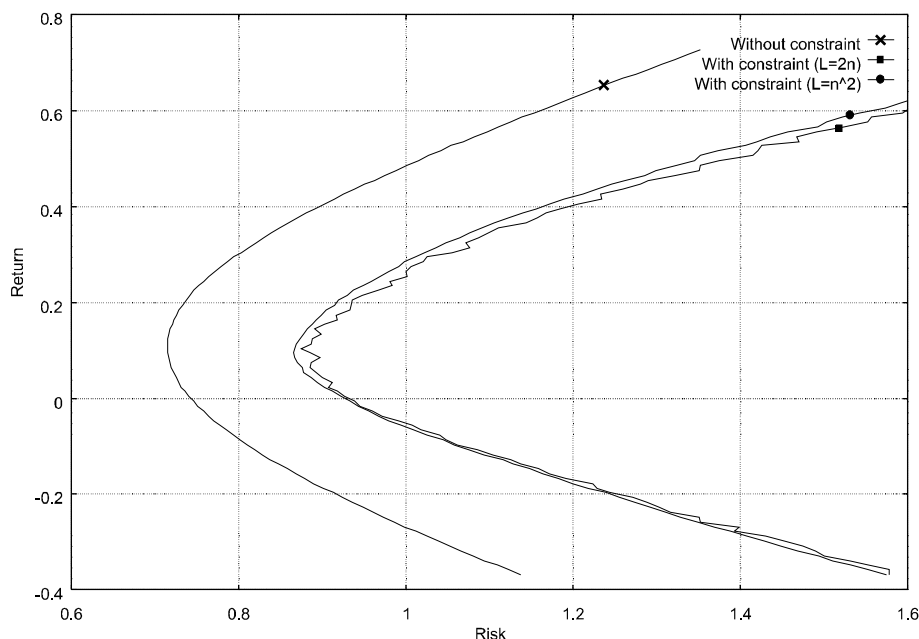

Fig. 11. Stop if no accepted move for $5L$ iterations; #equities n, 151; #portfolios, 110; $L = 2n$ or n^2; $N = 20$; $\rho_1 = 0.005$; $\rho_2 = 0.001$; $\rho_1 \rightarrow \rho_2 : \leqslant 10$ moves; time: $L = 2n :< 4$ seconds per portfolio, $L = n^2 : 4$ minutes 44 seconds per portfolio.
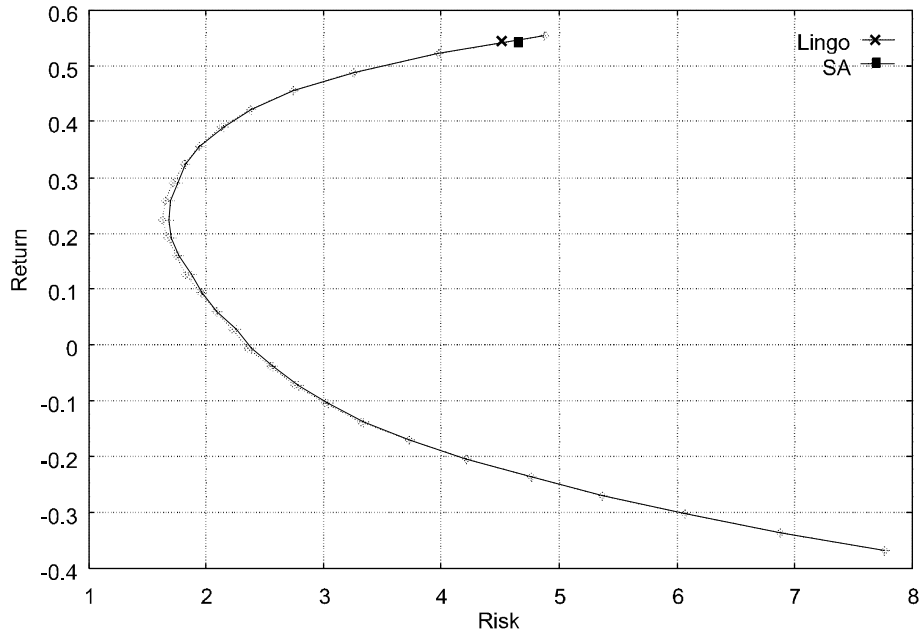
Fig. 12. Stop if no accepted move for $5L$ iterations; #equities n, 30; #portfolios, 30; $L = 2n$; $\underline{x}_i = 0 \; \forall i$; $\bar{x}_i = 1 \; \forall i$; $N = 20$; $\rho_1 = 0.005$; $\rho_2 = 0.001$; $\rho_1 \rightarrow \rho_2 : \leqslant 10$ moves; time : < 0.15 seconds per portfolio.
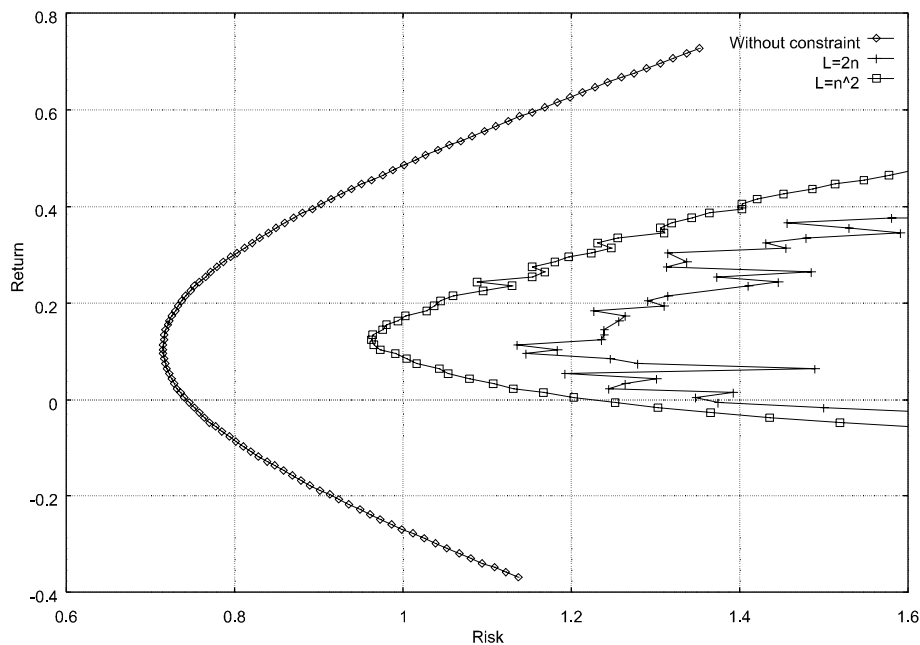

Fig. 13. Stop if no accepted move for $5L$ iterations; #equities n, 151; #portfolios, 110; $L = 2n$ or n^2; $\underline{x}_i = 0 \; \forall i$; $\bar{x}_i = 1 \; \forall i$; $x_i^{(0)} | N = 20$; $\underline{B}_i = 0.05 \; \forall i$; $\underline{S}_i = 0.05 \; \forall i$; $N = 20$; $\rho_1 = 0.005$; $\rho_2 = 0.001$; $\rho_1 \rightarrow \rho_2 : \leqslant 10$ moves; time: $L = 2n : < 2$ seconds per portfolio, $L = n^2 : 3$ minutes 33 seconds per portfolio.
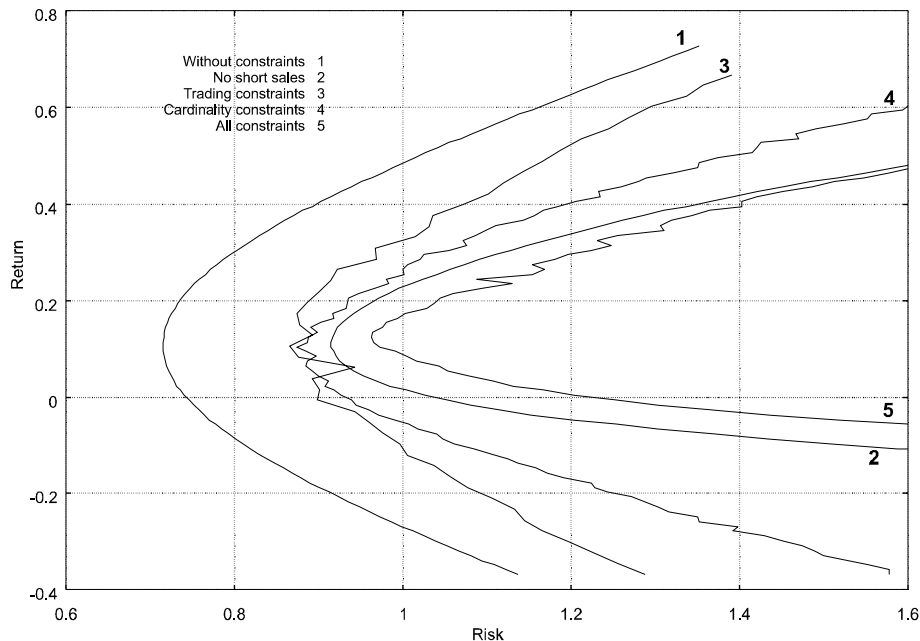
Fig. 14. Fig. 1 (without constraint) + Fig. 3 (ceiling) + Fig. 10 $L = n^3$ (trading) + Fig. 11 $L = n^2$ (cardinality) + Fig. 13 $L = n^2$ (all).

7. Conclusions

Portfolio selection gives rise to difficult optimization problems when realistic side-constraints are added to the fundamental Markowitz model. Exact optimization algorithms cannot deal efficiently with such complex models. It seems reasonable, therefore, to investigate the performance of heuristic approaches in this framework.

Simulated annealing is a powerful tool for the solution of many optimization problems. Its main advantages over other local search methods are its flexibility and its ability to approach global optimality. Most applications of the SA metaheuristic, however, are to combinatorial optimization problems. In particular, its applicability to portfolio selection problems is not fully understood, yet. The main objective of this paper was therefore to investigate the adequacy of simulated annealing for the solution of a difficult portfolio optimization model.

As SA is a metaheuristic, there are quite a lot of choices to make in order to turn it into an actual algorithm. We have developed an original way to generate neighbors of a current solution. We have also proposed specific approaches to deal with each specific class of constraint, either by explicitly restricting the portfolios to remain in the feasible region or by penalizing infeasible portfolios.

Let us now try to draw some conclusions from this research. On the positive side, we can say that the research was successful, in the sense that the resulting algorithm allowed us to approximate the mean–variance frontier for medium-size problems within acceptable computing times. The algorithm is able to handle more classes of constraints than most other approaches found in the literature. Although there is a clear trade-off between the quality of the solutions and the time required to compute them, the algorithm can be said to be quite versatile since it does not rely on any restrictive properties of the model. For instance, the algorithm does not assume any underlying factor model for the generation of the covariance matrix. Also, the objective function could conceivably be replaced by any other measure of risk

(semi-variance or functions of higher moments) without requiring any modification of the algorithm. This is to be contrasted with the algorithms of Perold [22] or Bienstock [2], which explicitly exploit the fact that the objective function is quadratic and that the covariance matrix is of low rank.

On the negative side, it must be noticed that the tailoring work required to account for different classes of constraints and to fine-tune the parameters of the algorithm was rather delicate. The trading constraints, in particular, are especially difficult to handle because of the discontinuities they introduce in the space of feasible portfolios. Introducing additional classes of constraints or new features in the model (e.g. transaction costs) would certainly prove quite difficult again.

Acknowledgements

References

[1] E. Aarts, J.K. Lenstra, Local Search in Combinatorial Optimization, John Wiley & Sons, New York, 1997.
[2] D. Bienstock, Computational study of a family of mixed-integer quadratic programming problems, Mathematical Programming 74 (1996) 121–140.
[3] F. Catanas, On a neighbourhood structure for portfolio selection problems, Working paper, Centre for Quantitative Finance, Imperial College, London, 1998.
[4] V. Černý, Thermodynamical approach to the travelling salesman problem: An efficient simulation algorithm, Journal of Optimization Theory and Applications 45 (1985) 41–51.
[5] T.J. Chang, N. Meade, J.E. Beasley, Y.M. Sharaiha, Heuristics for cardinality constrained portfolio optimisation, Computers and Operations Research 27 (2000) 1271–1302.
[6] Computational Science Education Project, Mathematical Optimization, Electronic book at http://csep1.phy.ornl.gov/mo/mo.html, 1995.
[7] G.M. Constantinides, A.G. Malliaris, Portfolio theory, in: R.A. Jarrow, V. Maksimovic, W.T. Ziemba (Eds.), Finance, Elsevier Science, Amsterdam, 1995, pp. 1–30.
[8] H. Dahl, A. Meeraus, S.A. Zenios, Some financial optimization models: I. Risk management, in: S.A. Zenios (Ed.), Financial Optimization, Cambridge University Press, Cambridge, 1996, pp. 3–36.
[9] A. Dekkers, E. Aarts, Global optimization and simulated annealing, Mathematical Programming 50 (1991) 367–393.
[10] R.S. Dembo, J.M. Mulvey, S.A. Zenios, Large-scale nonlinear network models and their application, Operations Research 37 (1989) 353–372.
[11] E.J. Elton, M.J. Gruber, Modern Portfolio Theory and Investment Analysis, fourth ed., John Wiley, New York, 1991.
[12] F. Hamza, J. Janssen, Extension réaliste pour la sélection de portefeuilles en présence des coûts de transaction, Working paper, Université Libre de Bruxelles, 1997.
[13] D.S. Johnson, C.R. Aragon, L.A. McGeoch, C. Scevon, Optimization by simulated annealing: An experimental evaluation. Part I, Graph partitioning, Operations Research 37 (1989) 865–892.
[14] S. Kirkpatrick, C.D. Gelatt, P.M. Vecchi, Optimization by simulated annealing, Science 220 (1983) 671–680.
[15] H. Konno, H. Yamazaki, Mean-absolute deviation portfolio optimization model and its applications to Tokyo stock market, Management Science 37 (1991) 519–531.
[16] H. Levy, M. Sarnat, Portfolio and Investment Selection: Theory and Practice, Prentice-Hall International, Englewood Cliffs, NJ, 1984.
[17] A. Loraschi, A. Tettamanzi, M. Tomassini, P. Verda, Distributed genetic algorithms with an application to portfolio selection problems, in: D.W. Pearson, N.C. Steele, R.F. Albrecht (Eds.), Proceedings of the International Conference on Artificial Neural Networks and Genetic Algorithms ICANNGA95, Springer-Verlag, Berlin, 1995, pp. 384–387.
[18] L. Lovász, M. Simonovits, Random walks in a convex body and an improved volume algorithm, Random Structures and Algorithms 4 (1993) 359–412.

[19] D.G. Luenbeger, Investment Science, Oxford University Press, Oxford, 1998.

[20] H.M. Markowitz, Portfolio selection, Journal of Finance 7 (1952) 77–91.

[21] I.H. Osman, G. Laporte, Metaheuristics: A bibliography, Annals of Operations Research 63 (1996) 513–623.

[22] A.F. Perold, Large-scale portfolio optimization, Management Science 30 (1984) 1143–1160.

[23] M. Pirlot, General local search heuristics in combinatorial optimization: A tutorial, Belgian Journal of Operations Research, Statistics and Computer Science 32 (1992) 7–68.

[24] H. Takehara, An interior point algorithm for large scale portfolio optimization, Annals of Operations Research 45 (1993) 373–386.

[25] P.J.M. van Laarhoven, E.H. Aarts, Simulated Annealing: Theory and Applications, Kluwer Academic Publishers, Dordrecht, 1988.

[26] P. Wolfe, The simplex method for quadratic programming, Econometrica 27 (1959) 382–398.

[27] Z.B. Zabinsky, R.L. Smith, J.F. McDonald, H.E. Romeijn, D.E. Kaufman, Improving hit-and-run for global optimization, Journal of Global Optimization 3 (1993) 171–192.