Ann Oper Res (2012) 193:129–158
DOI 10.1007/s10479-011-0862-y

Heuristic optimisation in financial modelling

Manfred Gilli · Enrico Schumann

Published online: 30 March 2011

Abstract There is a large number of optimisation problems in theoretical and applied finance that are difficult to solve as they exhibit multiple local optima or are not 'well-behaved' in other ways (e.g., discontinuities in the objective function). One way to deal with such problems is to adjust and to simplify them, for instance by dropping constraints, until they can be solved with standard numerical methods. We argue that an alternative approach is the application of optimisation heuristics like Simulated Annealing or Genetic Algorithms. These methods have been shown to be capable of handling non-convex optimisation problems with all kinds of constraints. To motivate the use of such techniques in finance, we present several actual problems where classical methods fail. Next, several well-known heuristic techniques that may be deployed in such cases are described. Since such presentations are quite general, we then describe in some detail how a particular problem, portfolio selection, can be tackled by a particular heuristic method, Threshold Accepting. Finally, the stochastics of the solutions obtained from heuristics are discussed. We show, again for the example from portfolio selection, how this random character of the solutions can be exploited to inform the distribution of computations.

Keywords Optimisation heuristics · Financial optimisation · Portfolio optimisation

This paper builds on an invited lecture by Manfred Gilli at the APMOD 2008 conference in Bratislava, Slovakia. The authors would like to thank the conference organisers, in particular Georg Pflug and Ronald Hochreiter, and participants of the session. Both authors also gratefully acknowledge financial support from the EU Commission through MRTN-CT-2006-034270 COMISEF.

M. Gilli (✉)
Department of Econometrics, University of Geneva and Swiss Finance Institute, Bd du Pont d'Arve 40, 1211 Geneva 4, Switzerland
e-mail: Manfred.Gilli@unige.ch

E. Schumann
VIP Value Investment Professionals AG, Zug, Switzerland
e-mail: es@vipag.com

1 Introduction

Financial economics is essentially concerned with two questions: how much to save, and how to save, that is, how to invest income not consumed (Constantinides and Malliaris 1995). Traditionally, economists have formulated both these questions as optimisation problems (Dixit 1990); the latter one has, however, received much greater attention in applied finance, and here the optimisation models have come to be deployed in practice.

To solve such models, many researchers and practitioners rely on what we call here 'classical' optimisation techniques. Classical methods are, for the purpose of this paper, defined as methods that require convexity or at least well-behaved objective functions as they are often based on gradients or related indicators for descent-direction. They are mathematically well-founded; numerically, there are powerful solvers available which can efficiently tackle even large-scale instances of given problems. Methods that belong to this approach are for instance linear and quadratic programming. The efficiency and elegance of these methods comes at a cost, though, since considerable constraints are put on the problem formulation, that is, the functional form of the optimisation criterion and the constraints. The analyst often has to shape the problem in a way that it can be solved by such methods. Thus, the answer that the final model provides is a precise one, but often only to an approximative question.

An alternative approach which we describe in this paper is to use heuristic optimisation techniques. Heuristics are a relatively new development in optimisation theory. Even though early examples date back to the 1960s or so, these methods have become practically relevant only in recent decades with the enormous growth in computing power. Heuristics aim at providing good and fast approximations to optimal solutions; the underlying theme of heuristics may thus be described as seeking approximative answers to exact questions.[1] In fact, heuristics have been shown to work well for problems that are completely infeasible for classical approaches (Michalewicz and Fogel 2004). They are conceptually often very simple; implementing them rarely requires high levels of mathematical sophistication or programming skills. Heuristics are flexible as adding, removing or changing constraints or exchanging objective functions can be easily accomplished. These advantages come at a cost as well, as the obtained solution is only a (in almost all cases) stochastic approximation, a random variable. However, such a solution may still be better than a poor deterministic one (which, even worse, may not even become recognised as such) or no solution at all when classical methods cannot be applied. In fact, for many practical purposes, the goal of optimisation is probably far more modest than to find the truly best solution. Rather, any good solution, where 'good' means an improvement of the status quo, is appreciated.

There are two points that we wish to stress at the outset. First, we do not suggest to consider heuristics as better optimisation techniques than classical methods; the question rather is when to use what kind of method. If classical techniques can be applied, heuristic methods will practically always be less efficient.[2] When, however, given problems do not fulfil the requirements of classical methods (and the number of such problems seems large), the suggestion made in this article is not to tailor the problem to the available optimisation technique, but to choose an alternative, heuristic, technique to optimise.

[1] In optimisation theory, the solution to a model is the vector of decision variables that minimises the given objective, possibly subject to constraints, i.e., there is no need to speak of 'optimal solutions'. In this paper we use the term in a less strict sense; a solution is rather the output of a software package, notwithstanding its quality.

[2] In some cases the borderline between these two approaches may be less clearcut than we describe it here; in particular, hybridisation is possible. See Maniezzo et al. (2009).

The second point concerns the empirical application of optimisation in finance, and it applies to both classical and heuristic techniques. In this article, we will mostly be concerned with computational and numerical issues for given, well-defined problems, and given data. When constructing optimisation models in practice, the question what to optimise, that is how to formulate the optimisation model, is just as important as how to optimise. Financial modelling always starts with an actual problem (e.g., how to invest?). To solve this problem, we write down a model, consisting for our purposes of an objective function and constraints. Then, we solve this model, typically on a computer. So we can roughly divide the modelling process into two steps: from reality to the model, and then from the model to its numerical solution. (For more discussion of this point, see Gilli and Schumann (2010c)). The question then is: how do we evaluate a solution? We can take the easy way, and only look at the numerical solution of our model. But the result of our computation may or may not be a good solution to our actual problem. For example, when fitting models to historical financial time series, we may find 'spikes' in the objective functions that are often entirely spurious. Hence, if the optimiser finds such solutions (and it should, if it works properly), the result will likely be an overfitting. (An area where this is extreme is algorithmic trading, see Dacorogna et al. (2001, Chap. 11).) Possible solutions, which may include more emphasis on data modelling or incorporating alternative, more robust objective functions, are in our view often under-researched, in particular when it comes to their actual empirical performance. The advantage of heuristics here is that when we formulate the model, we are quite unconstrained with regard to the tractability of the model; hence emphasis can be put on the empirical merits of specific models. Optimisation is only a tool; it is the application of this tool that matters.

The paper is organised as follows: Sect. 2 will detail several examples of problems that arise in theoretical and applied finance. Our paper is not a survey, hence we do not claim that our selection of problems is comprehensive; we rather want to illustrate and motivate the use of heuristic methods in finance. For more detailed studies, see for example Maringer (2005). Schlottmann and Seese (2004) or Gilli et al. (2008) give more references to specific applications. In Sect. 3 we will give a brief introduction to heuristic methods, i.e., methods that can be applied to solve these problems. The emphasis will be on principles, rather than on details. Here again, our choice of techniques aims to motivate the use of heuristics; we do not aim to be comprehensive. Section 4 then discusses an important aspect of heuristics, namely the stochastic nature of the obtained solutions. Heuristic methods as presented in Sect. 3 may be regarded as 'recipes' rather than specific algorithms.[3] To demonstrate the process of implementing such a recipe, in Sect. 5 we will show how to apply a particular heuristic, Threshold Accepting, to a portfolio selection problem. Section 6 concludes.

2 Optimisation problems in finance

In this section we present several financial problems that lead to optimisation problems that cannot be solved with classical methods. The list is by no means exhaustive; its purpose is only to show that optimisation problems for which classical methods fail occur even in widely researched areas. Hence, these applications are prime candidates for the use of heuristics.

[3]Some authors prefer the notion of meta-heuristics for the concepts underlying these techniques, whereas only the specific implementations are called heuristics.

2.1 Portfolio optimisation with alternative risk measures

In the framework of modern portfolio optimisation (Markowitz 1952, 1959), a portfolio of assets is completely characterised by a desired property, the 'reward', and something undesirable, the 'risk'. Markowitz identified these two properties with the expectation and the variance of returns, respectively, hence the expression mean–variance optimisation (MVO). There exists by now a large body of evidence that financial asset returns are not normally distributed (Cont 2001), thus describing a portfolio by only its first two moments is often regarded as insufficient. Alternatives to MVO have been proposed, in particular replacing variance as the risk measure.[4]

Assume an investor has wealth v_0 and wishes to invest for one period. A given portfolio x, as it comprises risky assets, maps into a distribution of wealth at the end of the period, or, equivalently, into a distribution of losses $\ell(x)$. (Using losses is a convention that is common in the insurance literature. Let v_T be the end-of-period wealth, then $\ell(x) \equiv v_0 - v_T$. Thus we do not work with returns, but actual currency amounts, and we switch the sign of gains and losses. Hence, a positive number is a loss, a negative number is a gain. We could equivalently write the model in terms of gains or returns.)

The optimisation problem can be stated as follows

$$\begin{aligned} \min_x \quad & f\big(\ell(x)\big) \\ & E(\ell) \le -v_0 r_d \\ & x_j^{\inf} \le x_j \le x_j^{\sup} \quad j \in \mathcal{J} \\ & K_{\inf} \le \#\{\mathcal{J}\} \le K_{\sup} \\ & \dots \end{aligned}$$

The objective function, $f(\cdot)$, is a risk measure or a combination of multiple objectives to be minimised. Essentially, we could write down any function that—given our data—maps a portfolio into its quality. In fact, we have found it helpful not to think in terms of a mathematical description, but rather something like `solution.quality = function(x,...)`. That is, we only need to be able to write down (to program) a mapping from a portfolio to its quality, given a data set. Building blocks for an objective function include the portfolio's drawdown, partial moments, or loss quantiles; but from the viewpoint of computation it could be whatever the analyst wishes to optimise.

The vector x stores the (integer) numbers of assets held (we could also work with weights, but we do not need to). r_d is the desired return, hence $E(\ell) \le -v_0 r_d$ is the minimum return constraint. $x_j^{\inf}$ and $x_j^{\sup}$ are vectors of minimum and maximum holding sizes, respectively, for those assets included in the portfolio (i.e., those in $\mathcal{J}$). If short-sales were allowed, this constraint would read $x_j^{\inf} \le |x_j| \le x_j^{\sup}$. $K_{\inf}$ and $K_{\sup}$ are cardinality constraints that set a minimum and maximum number of assets to be in $\mathcal{J}$. There may be restrictions on transaction costs (without restrictions on their functional form), turnover, or lot size constraints (i.e., restrictions on the multiples of assets that can be traded). We may also add constraints that under certain market conditions, the portfolio needs to behave in a certain way (usually give a required minimum return).

Similar to this framework are index tracking problems where investors try to replicate a pre-defined benchmark (see for example Gilli and Këllezi (2002b)). This benchmark need

[4] We do not pursue this possibility here, but the techniques discussed later in this paper can also be used for utility optimisation, see for example Maringer (2008).

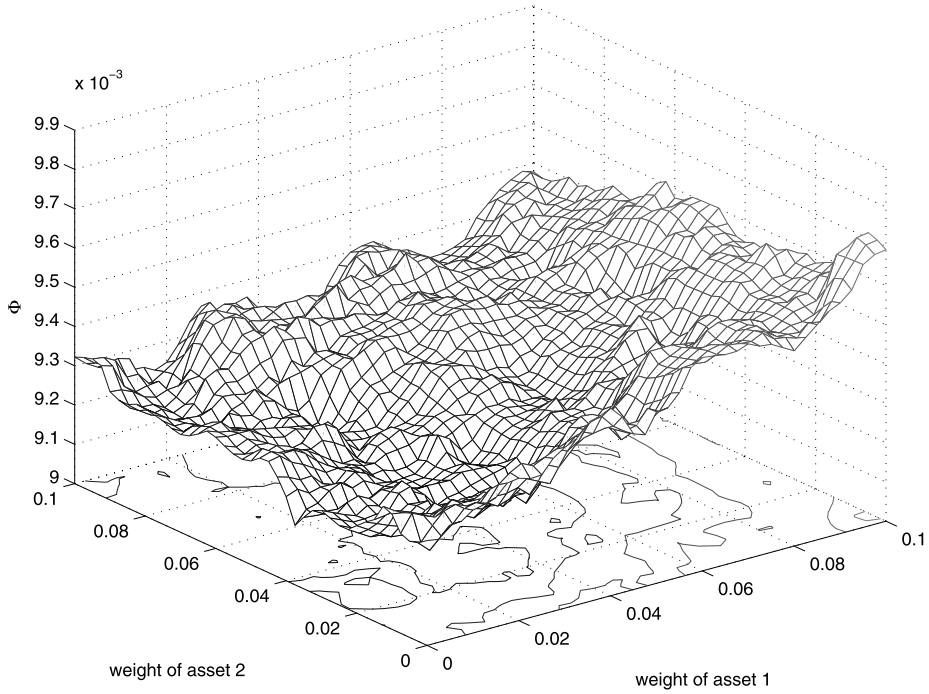


Fig. 1 Objective function values for VaR

not be a passive equity index. In the last few years, for instance, there have been attempts to replicate the returns of hedge funds, see Lo (2008).

Applying alternative risk measures generally necessitates to use the empirical distribution of returns. (As an extreme example, there seems little advantage in minimising kurtosis when stock returns are modelled by a Brownian Motion.) The resulting optimisation problem is rarely convex, in particular under reasonable constraints (like cardinality restrictions). To give an example, Fig. 1 shows the values of the objective function that particular solutions map into, for a particular problem where f is the portfolio's Value-at-Risk (VaR). The surface is not smooth, and we have many small valleys and basins. Any search that requires a globally convex solution, like gradient-based methods, will stop at the first local minimum encountered.

For some objective functions, the optimisation problem can be reformulated to be solved with classical methods, examples are Gaivoronski and Pflug (2005) or Rockafellar and Uryasev (2000), Chekhlov et al. (2005). Unfortunately, such solutions are very problem-specific and do not accommodate changes in the model formulation. How to use a heuristic in portfolio selection will be discussed more thoroughly in Sect. 5.

2.2 Model selection

Linear regression is a widely-used technique in finance. A common application are factor models in which the returns of single assets are described as functions of other variables.

Then

$$r = \begin{bmatrix} \phi_1 & \dots & \phi_k \end{bmatrix} \begin{bmatrix} \beta_1 \\ \vdots \\ \beta_k \end{bmatrix} + \epsilon$$

where r is a vector of returns for a given asset, ϕ are the vectors of factor realisations, β are the factor loadings and ϵ contains the residual. Such models are applied for instance to construct covariance matrices or in attempts to forecast future returns. The factors ϕ may be macroeconomic quantities or firm specific characteristics; alternatively, the analyst may use statistical factors, for instance extracted by principal components analysis. In practice, observable factors are often preferred, in particular since they are easier to interpret and to explain to clients. Given the vast amounts of financial data available, these factors may have to be picked from hundreds or thousands of available variables, in particular since we may also consider lagged variables (Maringer 2004). Hence, model selection becomes a critical issue, as we wish to only use a small number k of regressors from K possible ones, where $K \gg k$. (The analyst may use an information criterion, which penalises additional regressors, as the objective function; alternatively, in practice the problem may often be formulated as maximising in-sample fit under the restriction that k is not greater than a small fixed number.)

2.3 Robust/resistant regression

Empirical evidence over the last decades has shown that the Capital Asset Pricing Model (CAPM) explains asset returns in the cross-section rather badly (Fama and French 1993, 2004). However, when interpreting the CAPM as a one-factor model (Luenberger 1998, Chap. 8), the β-estimates become useful measures of a stock's general correlation with the market, which may for instance be used to construct covariance matrices (Chan et al. 1999).

The standard method to obtain parameter estimates in a linear regression is Ordinary Least Squares (OLS). OLS has appealing theoretical and practical (numerical) properties, but obtained estimates are often unstable in the presence of extreme observations which are rather common in financial time series (Chan and Lakonishok 1992; Knez and Ready 1997; Genton and Ronchetti 2008). Earlier contributions in the finance literature suggest some form of shrinkage of extreme β-estimates towards more reasonable levels, with different theoretical justifications (see for example Blume 1971; Vasicek 1973). Alternatively, the usage of robust or resistant estimation methods to obtain the regression parameters has been proposed (Chan and Lakonishok 1992; Martin and Simin 2003). Among the latter approaches, high breakdown point estimators are often regarded as desirable. The breakdown point of an estimator is the smallest percentage of contaminated (outlying) data that may cause the estimator to be affected by a bias. The Least Median of Squares (LMS) estimator, suggested by Rousseeuw (1984), ranks highly in this regard, as its breakdown point is (almost) 50%. (Note that OLS may equivalently be called Least Mean of Squares.)

There is of course a conceptual question as to what constitutes an outlier in financial time series. Markets may well produce extreme returns, and disregarding these by dropping or winsorising them may mean throwing away information. Errors in the data, though, for example stock splits that have not been accounted for, are clearly outliers. Such data errors seem to occur on a wide scale, even when using commercial data providers (Ince and Porter 2006). In particular then if large amounts of data are processed automatically, resistant regression techniques may be advisable. Unfortunately, LMS regression leads to non-convex optimisation models (Gilli and Winker 2008, Sect. 5.2). A particular search space is shown in Fig. 2.

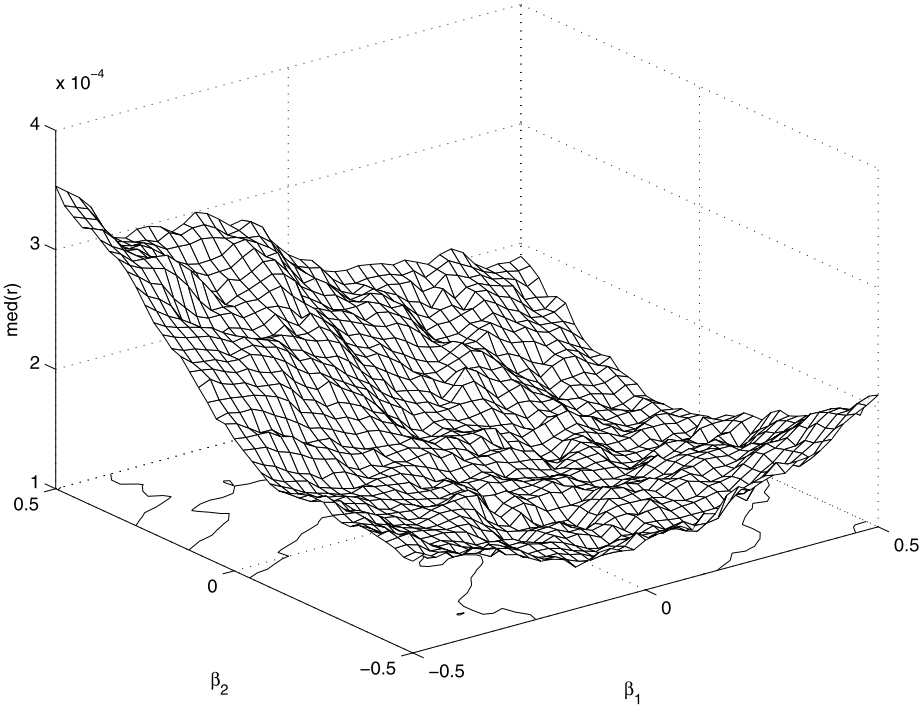


Fig. 2 LMS objective function

2.4 Agent-based models

Agent-based models (ABM) abandon the attempt to model markets and financial decisions with one representative agent (Kirman 1992). This results in models that quickly become analytically intractable, hence researchers rely on computer simulations to obtain results. ABM are capable of producing many of the stylised facts actually observed in financial markets like volatility clustering, jumps or fat tails. For overviews on ABM in finance, see for example LeBaron (2000, 2006).

Unfortunately, the conclusion of many studies stops at asserting that these models can in principle produce realistic market behaviour when parameters (like preferences of agents) are appropriately specified. This leads to the question what appropriate values should be, and how different models compare with one another when it comes to explaining market facts.

Gilli and Winker (2003) suggest to estimate the parameters of such models by indirect inference. This requires an auxiliary model that can easily be estimated, which in this case is simply a combination of several moments of the actual price data. A given set of parameters for the ABM is evaluated by measuring the distance between the average realised moments of the simulated series and the moments obtained from real data. This distance is then to be minimised by adjusting the parameters of the ABM.[5] Figure 3 shows the resulting search space for a particular ABM (the one described in Kirman 1993). The objective function does

[5]See Winker et al. (2007) for a more detailed analysis of objective functions for such problems.

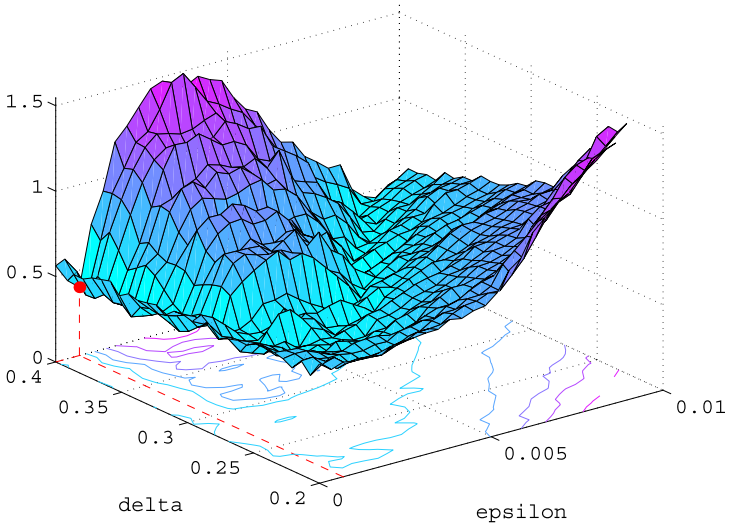


Fig. 3 Simulated objective function for Kirman's model for two parameters

not seem too irregular at all, but since the function was evaluated by a stochastic simulation of the model, it is always noisy and does not allow for the application of classical methods.

2.5 Calibration of yield structure models

The model of Nelson and Siegel (1987) and its extended version, introduced by Svensson (1994), are widely used to approximate the term structure of interest rates. Many central banks use the models to represent the spot and forward rates as functions of time to maturity; in several studies (e.g., Diebold and Li 2006) the models have also been used for forecasting interest rates.

Let $y_t(\tau)$ be the yield of a zero-coupon bond with maturity τ at time t, then the Nelson–Siegel model describes the zero rates as

$$y_t(\tau) = \beta_{1,t} + \beta_{2,t}\left[\frac{1-\exp(-\gamma_t)}{\gamma_t}\right] + \beta_{3,t}\left[\frac{1-\exp(-\gamma_t)}{\gamma_t} - \exp(-\gamma_t)\right] \tag{1}$$

where $\gamma_t = \frac{\tau}{\lambda_t}$. The Svensson version is given by

$$\begin{aligned} y_t(\tau) = {} & \beta_{1,t} + \beta_{2,t}\left[\frac{1-\exp(-\gamma_{1,t})}{\gamma_{1,t}}\right] + \beta_{3,t}\left[\frac{1-\exp(-\gamma_{1,t})}{\gamma_{1,t}} - \exp(-\gamma_{1,t})\right] \\ & + \beta_{4,t}\left[\frac{1-\exp(-\gamma_{2,t})}{\gamma_{2,t}} - \exp(-\gamma_{2,t})\right] \end{aligned} \tag{2}$$

where $\gamma_{1,t} = \frac{\tau}{\lambda_{1,t}}$ and $\gamma_{2,t} = \frac{\tau}{\lambda_{2,t}}$. The parameters of the models (β and λ) can be estimated by minimising the difference between the model rates y_t and observed rates y_t^{M} where the superscript stands for 'market'. An optimisation problem could be stated as

$$\min_{\beta,\lambda} \sum \left(y_t - y_t^{\mathrm{M}}\right)^2$$

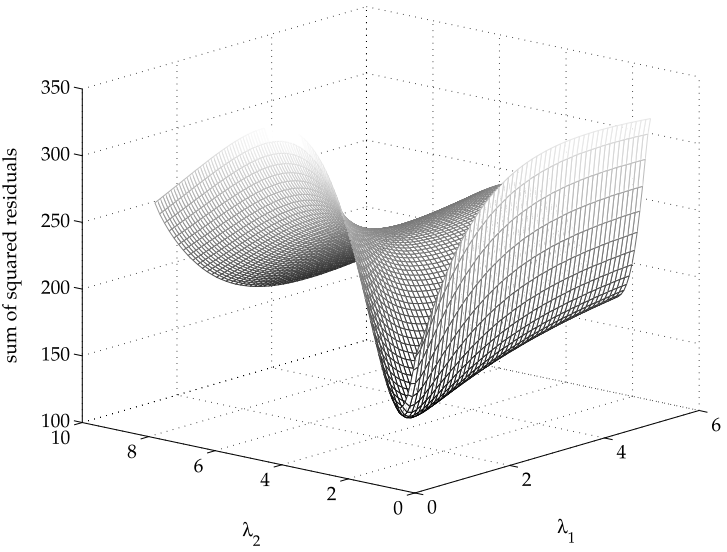


Fig. 4 Nelson–Siegel–Svensson model objective function

subject to constraints. We need to estimate four parameters for model (1), and six for model (2). Again, this optimisation problem is not convex; an example for a search space is given in Fig. 4.

2.6 Calibration of option pricing models

Prices of options and other derivatives are usually modelled as functions of the underlying securities' characteristics (Madan 2001). Parameters for such models are often inferred by solving inverse problems, that is, we try to find parameter values for which the model gives prices that are close to actual market prices. In case of the Black–Scholes–Merton model only one parameter, volatility, needs to be specified; this can be done efficiently with Newton's method (Manaster and Koehler 1982). More recent option pricing models (see for instance Bakshi et al. 1997; Bates 2003) aim to generate prices that are consistent with the empirically observed implied volatility surface (Cont and da Fonseca 2002). Calibrating these models generally requires to set more parameters, and leads to more difficult optimisation problems.

One particular pricing model is that of Heston (1993); it is popular because it offers closed-form solutions for option prices. Under the Heston model the stock price (S) and its variance (V) dynamics are described by

$$\begin{aligned} \mathrm{d}S_t &= \mu S_t \mathrm{d}t + \sqrt{V_t} S_t \mathrm{d}W_t^1 \\ \mathrm{d}V_t &= \kappa(\theta - V_t)\mathrm{d}t + \sigma\sqrt{V_t}\mathrm{d}W_t^2 \end{aligned}$$

where the two Brownian motion processes are correlated, that is $\mathrm{d}W_t^1 \mathrm{d}W_t^2 = \rho \mathrm{d}t$. As can be seen from the second equation, volatility is mean-reverting in the Heston model. Under the risk-neutral probability, the model requires the specification of 5 parameters (Mikhailov and Nögel 2003). Even though some of these parameters could be estimated from the time series

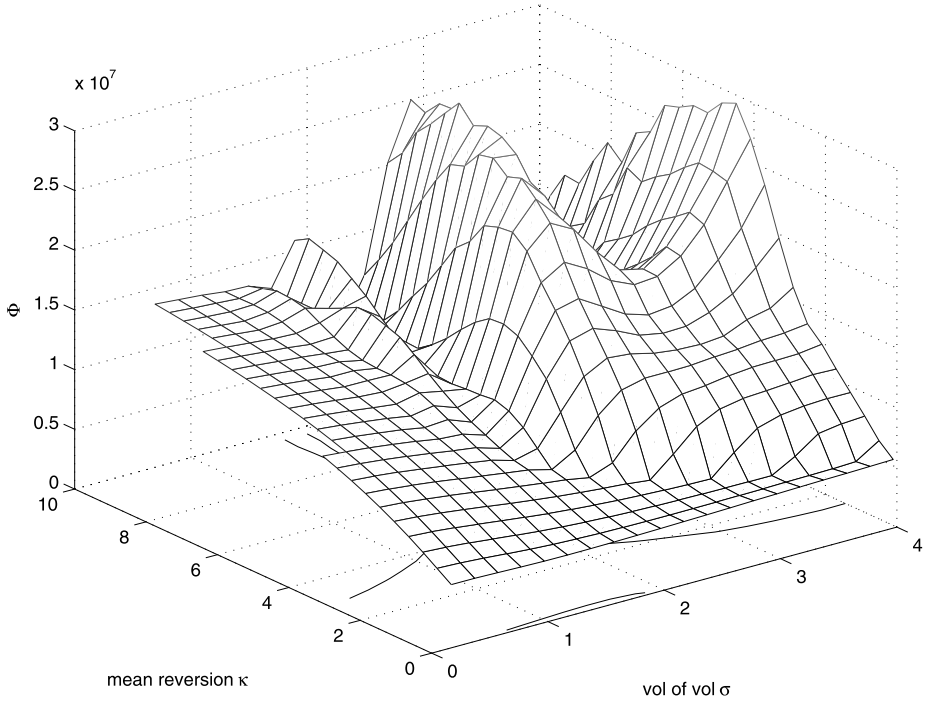


Fig. 5 Heston model objective function

of the underlying, the general approach to fit the model is to minimise the distance between the theoretical and observed prices. Hence a possible objective function is

$$\min \sum_{n=1}^{N} w_n (C_n^{\mathrm{H}} - C_n^{\mathrm{M}})^2$$

where N is the number of option quotes available, C^{H} and C^{M} are the theoretical and actual option prices, respectively, and w are weights (Hamida and Cont 2005). The optimisation model also includes parameter restrictions, for example certain parameters like correlations will need to lie in reasonable ranges, or we may want to enforce the parameters to be such that the volatility cannot become negative.

Figure 5 shows the resulting objective function values when we vary two parameters (volatility of volatility and mean-reversion speed) with the remaining parameters fixed. As can be seen, in certain parts of the parameter domain the resulting objective function is not well-behaved, hence standard methods may not find the global minimum.

3 Heuristic optimisation methods

In this section, we will outline the basic principles of heuristic methods, and summarise several well-known methods. These descriptions are not meant as definite references (all the presented techniques exist in countless variations), but are to give the basic rules by which these methods operate.

3.1 Definition & principles

The term 'heuristic' is used in different disciplines (like mathematics, psychology, or computer science) for different, though related, purposes. In this paper, we will always stay in the framework of optimisation models, thus our basic problem is to minimise a function, often subject to constraints. Heuristics, then, are a class of numerical methods that can solve such problems. Following similar definitions in Zanakis and Evans (1981), Barr et al. (1995) and Winker and Maringer (2007), we characterise the term optimisation heuristic through several criteria:

- The method should give a good stochastic approximation of the true optimum; 'goodness' can be measured in computing time and solution quality.
- The method should be robust to changes in the given problem's objective function and constraints. Furthermore, results should not vary too much with changes in the parameter settings of the heuristic.
- The technique should be easy to implement.
- Implementation and application of the technique should not require subjective elements.

Such a definition is not unambiguous. Even in the optimisation literature we find different characterisations of the term. In Operations Research, heuristics are often not regarded as stand-alone methods but as workarounds for problems in which 'real' techniques like linear programming do not work satisfactorily, see for instance Hillier (1983); or the term is used for ad hoc adjustments to methods that seem to work well but whose advantage cannot be proved mathematically. We will not follow this conception. Heuristics shall be defined here as general-purpose methods that can handle problems that are sometimes completely infeasible for classical approaches. A large and growing number of methods meets the given criteria. Still, it is interesting to note that even though there exists considerable evidence of the good performance of heuristics, they are still not widely applied in research and practice.

We shall divide heuristics into Local Search methods and constructive methods (Gilli and Winker 2009). More fine-grained classifications are possible, but for our purpose this division is sufficient. For Local Search methods, the algorithm moves from solution to solution, that is, in each iteration a complete existing solution is changed to obtain a new solution. The term 'local' should not always be taken too literally, as some methods (e.g., Genetic Algorithms) are discontinuous in their creation of new solutions. Hence a new solution may be significantly different from the old one; it will, however, usually share some characteristics with its predecessor. Constructive methods on the other hand build new solutions in a stepwise procedure, i.e., the algorithm starts with an 'empty' solution and adds components iteratively. A standard example for this approach comes from the Travelling Salesman Problem. Here solution methods exist in which an algorithm starts with one city and then adds the remaining cities one at a time until a complete tour is created. In this paper, we will only consider Local Search methods. To keep the presentation simple, we only differentiate between trajectory methods and population-based methods, both terms to be explained below. For a more complete classification system of heuristics, see Talbi (2002) or Winker and Gilli (2004).

The concept of (stochastic) Local Search is not new, but was often not regarded as a complete method. Rather, it was considered a component within other techniques, for example as a safeguard against saddlepoints in methods relying on measures of descent direction (Gill et al. 1986, p. 295). A classical Local Search starts with a (possibly random) feasible solution x^c and picks (usually again randomly) a new solution x^n close to the old one. This new solution is called the neighbour solution. If x^n is better than x^c, the new solution is

accepted, if not, it is rejected. For a given objective function, a Local Search is completely described by how it chooses a neighbour solution and by its stopping criterion. The latter may simply be a preset number of steps. Algorithm 1 describes the procedure.

Algorithm 1 Pseudocode for Local Search

```
Initialise n_Steps
Randomly generate current solution x^c ∈ X
for i = 1 : n_Steps do
    Generate x^n ∈ N(x^c)  and compute  Δ = f(x^n) − f(x^c)
    if Δ < 0 then x^c = x^n
end for
x^sol = x^c
```

A large number of variations of Local Search exist. We can for instance wrap Algorithm 1 into a loop of restarts; but in particular, it is the choice of the neighbourhood $\mathcal{N}$ that influences the algorithm's performance. The neighbourhood defines how we move from one solution to another. In the extreme, the neighbourhood may allow to move from any solution to any possible solution in just one step, but generally the term 'neighbourhood' will keep its literal meaning and solutions are changed only gradually. For an introduction to Local Search, see Hoos and Stützle (2004); more recent developments are Mladenović and Hansen (1997), Schrimpf et al. (2000) or Pisinger and Ropke (2010). See also Gendreau and Potvin (2010).

In a sufficiently well-behaved setting a Local Search will, for a suitable neighbourhood definition and enough iterations, succeed in finding the global minimum, even though it is certainly not the most efficient method. The compensation for this lack of efficiency is that Local Search only requires that the objective function can be evaluated for a given solution x; there is no need for the objective function to be continuous or differentiable, or to actually compute the gradient. Unfortunately, for problems with many local minima, a single restart of a Local Search will still stop at the first local optimum it encounters. Heuristic methods that build on Local Search employ different strategies to overcome such local minima. A few of these methods will be outlined next.

3.2 Trajectory methods

3.2.1 Simulated Annealing

Trajectory methods evolve a single solution over time. By changing this solution gradually, the algorithm follows a path (trajectory) through the search space. One of the oldest and best-known methods in this class is Simulated Annealing (SA), introduced in Kirkpatrick et al. (1983). SA was conceived for combinatorial problems, but can easily be used for continuous problems as well. Algorithm 2 gives the pseudocode of the procedure.

Like a Local Search, SA starts with a random solution x^c and creates a new solution x^n by adding a small perturbation to x^c. If the new solution is better ($\Delta < 0$), it is accepted. In case it is worse, though, SA applies a stochastic acceptance criterion, thus there is still a chance that the new solution is accepted, albeit only with a certain probability. This probability is a decreasing function of both the order of magnitude of the deterioration and the time the algorithm has already run. The latter feature is controlled by the temperature parameter T which is reduced over time; hence impairments in the objective function become less likely to be accepted and eventually SA turns into classical Local Search. The algorithm stops after a predefined number of iterations $R_{\max}$.

Algorithm 2 Pseudocode for Simulated Annealing

```
1: Generate initial solution x^c, initialise R_max and T
2: for r = 1 to R_max do
3:     while stopping criteria not met do
4:         Compute x^n ∈ N(x^c)     (neighbour to current solution)
5:         Compute Δ = f(x^n) − f(x^c) and generate u (uniform random variable)
6:         if (Δ < 0) or (e^{−Δ/T} > u) then x^c = x^n
7:     end while
8:     Reduce T
9: end for
```

3.2.2 *Threshold Accepting*

Threshold Accepting (TA) was introduced by Dueck and Scheuer (1990) and is very similar to SA. Algorithm 3 shows that the two methods differ mainly in their acceptance criterion (line 6 in Algorithm 2, line 7 in Algorithm 3). In fact, Moscato and Fontanari (1990) suggested the same deterministic updating rule in SA and called it 'threshold updating'. Both SA and TA are sometimes referred to as threshold methods.

Algorithm 3 Pseudocode for Threshold Accepting

```
1: Initialise n_Rounds and n_Steps
2: Compute threshold sequence τ_r
3: Randomly generate current solution x^c ∈ X
4: for r = 1 : n_Rounds do
5:     for i = 1 : n_Steps do
6:         Generate x^n ∈ N(x^c)  and compute  Δ = f(x^n) − f(x^c)
7:         if Δ < τ_r then x^c = x^n
8:     end for
9: end for
10: x^sol = x^c
```

Whereas in SA solutions that worsen the objective function are accepted stochastically, TA accepts deteriorations unless they are greater than some threshold τ_r. The n_{Rounds} thresholds decrease over time, hence like SA the algorithm turns into a Local Search.

For an in-depth description of TA, see Winker (2001). In Sect. 5 the application of TA to a portfolio optimisation problem will be discussed.

3.2.3 *Tabu Search*

Most heuristics differ from classical methods by introducing an element of chance (e.g., by randomly picking neighbour solutions). Tabu Search (TS), at least in its standard form, is an exception as it is deterministic for a given starting value.[6] TS is described in Glover (1986), Glover and Laguna (1997) and detailed in Algorithm 4. TS was designed for discrete search spaces; its strategy to overcome local minima is to keep a memory of recently visited solutions. These are forbidden (tabu) as long as they stay in the algorithm's memory. Thus, a TS can manage to walk away from a local minimum as it is temporarily not allowed to revisit this solution.

[6] We can easily alter the algorithm (line 4) to choose a neighbour solution randomly, or include randomisation at other stages. But see Glover (2007).

Algorithm 4 Pseudocode for Tabu Search

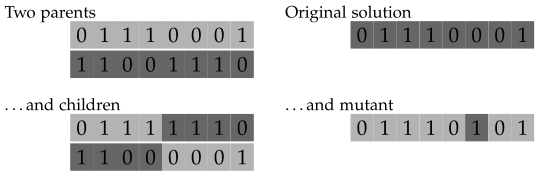
1: Generate current solution x^c and initialise tabu list $T = \emptyset$
2: **while** stopping criteria not met **do**
3: Compute $V = \{x | x \in \mathcal{N}(x^c)\} \backslash T$
4: Select $x^n = \min(V)$
5: $x^c = x^n$ and $T = T \cup x^n$
6: Update memory
7: **end while**

3.3 Population-based methods

For trajectory methods, the main strategy for escaping local minima was to temporarily allow uphill-moves. Population-based methods employ the same principle, but they do so by maintaining a whole collection of different solutions at a time, some of which are worse than others. Population-based methods are usually good at exploration, i.e., they often work well for large search spaces like in combinatorial problems.

3.3.1 Genetic Algorithms

The best-known technique in this category are Genetic Algorithms (GA). Genetic algorithms were described by John Holland in the 1970s (Holland 1992); pseudocode can be found in Algorithm 5. GA are inspired by evolutionary biology, hence the procedure appropriately starts with a whole population of solutions; the objective function becomes a fitness function; iterations become generations. In standard GA, solutions are coded as binary strings like 0 1 1 1 0 0 0 1. Such a string may be a binary representation of an integer or real number; but in many discrete problems there is a more natural interpretation: in a selection problem, a 1 may indicate a selected item from an ordered list, a 0 may stand for an item that does not enter the solution. New candidate solutions, called children or offspring, are created by crossover (i.e., mixing existing solutions) and mutation (i.e., randomly changing components of solutions), as illustrated here:



a) Crossover b) Mutation

To keep the population size $\#\{P\}$ constant, at the end of each generation, there is a selection among parents and children (line 10). Different variations exist, for example only the $\#\{P\}$ fittest solutions may stay in P, or the survival of a solution may be stochastic with the probability of survival proportional to a solution's fitness.

3.3.2 Differential Evolution

A more recent contribution to population-based methods is Differential Evolution (DE) (Storn and Price 1997). DE was developed for continuous functions; Algorithm 6 assumes

Algorithm 5 Pseudocode for Genetic Algorithms

Generate initial population P of solutions
while stopping criteria not met **do**
  Select $P' \subset P$ (mating pool), initialise $P'' = \emptyset$ (set of children)
  for $i = 1$ to n **do**
    Select individuals x^{a} and x^{b} at random from P'
    Apply crossover to x^{a} and x^{b} to produce x^{child}
    Randomly mutate produced child x^{child}
    $P'' = P'' \cup x^{\mathrm{child}}$
  end for
  $P = \mathrm{survive}(P', P'')$
end while

Algorithm 6 Pseudocode for Differential Evolution

Initialise parameters n_{P}, n_{G}, $\mathtt{F}$ and $\mathtt{CR}$
Initialise population $P^{(1)}_{j,i}$, $j = 1, \ldots, d$, $i = 1, \ldots, n_{\mathrm{P}}$
for $k = 1$ to n_{G} **do**
  $P^{(0)} = P^{(1)}$
  for $i = 1$ to n_{P} **do**
    Generate $r_1, r_2, r_3 \in \{1, \ldots, n_{\mathrm{P}}\}$, $r_1 \neq r_2 \neq r_3 \neq i$
    Compute $P^{(v)}_{\cdot,i} = P^{(0)}_{\cdot,r_1} + \mathtt{F} \times (P^{(0)}_{\cdot,r_2} - P^{(0)}_{\cdot,r_3})$
    for $j = 1$ to d **do**
      if $u < \mathtt{CR}$ **then** $P^{(u)}_{j,i} = P^{(v)}_{j,i}$ **else** $P^{(u)}_{j,i} = P^{(0)}_{j,i}$
    end for
    if $f(P^{(u)}_{\cdot,i}) < f(P^{(0)}_{\cdot,i})$ **then** $P^{(1)}_{\cdot,i} = P^{(u)}_{\cdot,i}$ **else** $P^{(1)}_{\cdot,i} = P^{(0)}_{\cdot,i}$
  end for
end for

Algorithm 7 Pseudocode for Particle Swarm

Initialise parameters n_{P}, n_{G} and c
Initialise particles $P^{(0)}_i$ and velocity $v^{(0)}_i$, $i = 1, \ldots, n_{\mathrm{P}}$
Evaluate objective function $F_i = f(P^{(0)}_i)$, $i = 1, \ldots, n_{\mathrm{P}}$
$Pbest = P^{(0)}$, $Fbest = F$, $Gbest = \min_i(F_i)$, $gbest = \mathrm{argmin}_i(F_i)$
for $k = 1$ to n_{G} **do**
  for $i = 1$ to n_{P} **do**
    $\Delta v_i = c\,u\,(Pbest_i - P^{(k-1)}_i) + c\,u\,(Pbest_{\mathrm{gbest}} - P^{(k-1)}_i)$
    $v^{(k)}_i = v^{(k-1)} + \Delta v_i$
    $P^{(k)}_i = P^{(k-1)}_i + v^{(k)}_i$
  end for
  Evaluate objective function $F_i = f(P^{(k)}_i)$, $i = 1, \ldots, n_{\mathrm{P}}$
  for $i = 1$ to n_{P} **do**
    if $F_i < Fbest_i$ **then** $Pbest_i = P^{(k)}_i$ and $Fbest_i = F_i$
    if $F_i < Gbest$ **then** $Gbest = F_i$ and $gbest = i$
  end for
end for

that n_P solutions are stored in real-valued vectors of length d. In each generation, the algorithm creates a candidate solution for each existing solution $P^{(k)}_{\cdot,i}$. This new solution is created by first adding the difference, weighted by a parameter $\mathtt{F}$, between two other solutions to a third solution. Then an elementwise crossover takes place with probability $\mathtt{CR}$

between this 'auxiliary' solution $P^{(v)}_{\cdot,i}$ and the existing solution $P^{(k)}_{\cdot,i}$. If this final candidate solution $P^{(u)}_{\cdot,i}$ is better than $P^{(k)}_{\cdot,i}$, it replaces it; if not, the old solution is kept.

3.3.3 Particle Swarm

While the metaphor for DE and GA was evolution, the narrative for Particle Swarm (PS) is based on flocks of birds that search for food (Eberhart and Kennedy 1995). Like DE, PS (see pseudocode in Algorithm 7) works for continuous functions, the population of n_P solutions are stored in real-valued vectors. In each iteration, a solution is updated by adding another vector called velocity v_i. This velocity changes over the course of the optimisation. More specifically, at the start of each iteration the directions towards the best solution found so far by the particular solution, $Pbest_i$, and the best overall solution, $Pbest_{gbest}$, are determined. The sum of these two directions (which are the differences between the respective vectors, see line 7) are perturbed by multiplication with a uniform random variable u and a constant c. The vector so obtained is added to the previous v_i; the resulting updated velocity is added to the respective solution.

This brief presentation of heuristics has not been complete; it did not even include the innumerable variations that exist of the described techniques. Different methods can also be combined into hybrid methods. It is often difficult to clearly define when a particular algorithm is considered a method of its own, and not just a variant. Such a distinction is not only driven by the actual characteristics of a technique, but also in a path-dependent way by how well researchers and operators accept a given idea. Threshold Accepting, for instance, changes only one element in Simulated Annealing, the acceptance criterion. Strictly speaking, it does not even change it, but uses a special case. Still, the method has been accepted. Moscato and Fontanari (1990) suggested the same mechanism, but did not consider the technique a new method. Another example are Memetic Algorithms (Moscato 1989). A Memetic Algorithm develops a population of solutions, but each solution also searches locally. The single solutions then cooperate (e.g., by crossover), but also compete (good solutions replace inferior ones). In essence, this is a hybrid of a population-based procedure and a Local Search technique; still, the algorithm is accepted as a single method.

Ultimately, the application should determine what method to use. The chosen method can then be tweaked until it performs as good as it gets, perhaps by changing its parameters (e.g., the mutation rate in GA), but it is also possible to enhance methods with new features, or to create hybrids. But on the other hand, it is useful to define specific techniques more strictly in terms of their characteristics and then 'stick to the algorithm', thus to establish a common language. As pointed out above, how well a method is accepted depends not only on its merits, but also on how knowledge about the method diffuses throughout communities of researchers and practitioners. In order to promote the proliferation of heuristic methods, it should help to have clear algorithms in mind, not just rough principles. Repeated successes of a method to solve problems of a given class (or even several classes) should then not only provide us with a picture of the strengths and weaknesses of the technique, but also increase the confidence that operators and researchers have into the method, and hence its acceptance. This is less of a problem in engineering, scientific computing, or operations research, where researchers seem readier to accept new methods; but in economics and finance there is a long way to go. In our view, the only way to contribute to the proliferation of heuristics in finance is by repeatedly demonstrating their capability to solve optimisation problems like those in Sect. 2.

4 Stochastics and convergence of the solution

4.1 Stochastics and computational resources

Many heuristic methods deliberately include stochastic elements, for example when creating a new candidate solution, or when deciding whether to accept a candidate solution. Restarting the same algorithm with a different seed value will in general not lead to the same solution. A related stochasticity may occur also for classical methods in the case of non-convex problems; here the obtained results may differ for different starting values.

If we characterise a solution by the obtained objective function value, then the result of an optimisation run can be regarded as a realisation of a random variable with a distribution $\mathcal{D}$ (Gilli and Winker 2008).[7] This distribution $\mathcal{D}$ is not symmetric, but bounded to the left (for minimisation) and degenerates for increasing computational resources to a single point, namely the global minimum. There exist convergence proofs for several heuristics, including TA (Althöfer and Koschnick 1991). The trouble with $\mathcal{D}$ is that we do not know its shape; but fortunately, we can easily estimate it. We can execute a reasonably large number of optimisation runs, each giving a solution with objective function f_i, and use the empirical distribution function of these values as an estimator for $\mathcal{D}$.

Most optimisation heuristics are conceptually simple, but computationally intensive. Computational cost can be measured by the total number of function evaluations or iterations the algorithm takes. Let the total amount of computational resources available be denoted by $\mathcal{I}$, then

$$\mathcal{I} = n_{\text{Restarts}} \times n_{\text{Iterations}}.$$

$\mathcal{D}$ will depend on the total number of iterations per restart ($n_{\text{Iterations}}$) and also on the specific parameter settings of the method. For instance, for a Local Search method $\mathcal{D}$ will depend on the chosen neighbourhood function. The restarts are draws from $\mathcal{D}$. The time an algorithm needs to produce a solution is proportional to $\mathcal{I}$ in a serial environment, but this does not hold for computations in parallel since we can, at least, distribute the restarts.

For a given heuristic and a given problem, two issues arise. First, increasing $\mathcal{I}$ should lead to better solutions, that is the average solution should decrease while the variability of solutions should also decrease and eventually go to zero as $\mathcal{I}$ goes to infinity. The speed of convergence for some increasing but finite sequence $\mathcal{I}_1, \mathcal{I}_2, \ldots, \mathcal{I}_p$ can be estimated.

Second, for a fixed amount of computational resources available, it is important to know how to allocate these resources among restarts and iterations. Intuitively, we may rather use fewer iterations and thus have a less-favourable distribution of solutions, but still by picking the best of several restarts produce better results than just doing one optimisation with many steps, see Winker (2001, pp. 129–134).

It is interesting to analyse how such considerations change in a parallel environment. To give a concrete example, we look at the portfolio optimisation problem described earlier; for more details, the reader is referred to Gilli and Schumann (2010a).

4.2 Distributing restarts in portfolio optimisation

The stochastics of the solution can be exploited to inform how to distribute computations for heuristic optimisation problems. The basic intuition is that we can trade off iterations

[7] Usually we will consider the distribution of objective function values when we investigate $\mathcal{D}$, even though we may sometimes also be interested in the distribution of the resulting choice variables into which the respective objective function values map.

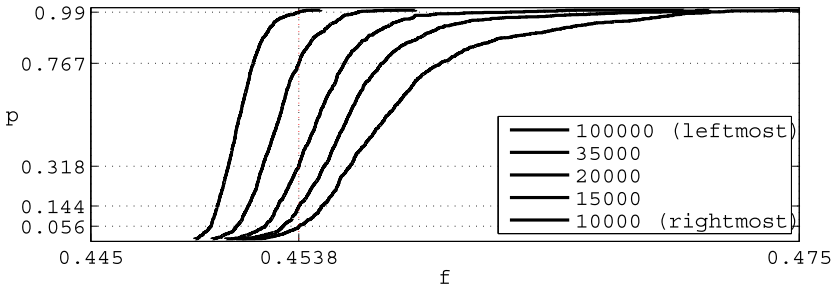


Fig. 6 Distribution of solutions for increasing number of iterations

per restart with the number of restarts. That is, we may allow for fewer steps in each single optimisation run, but simply generate more draws by restarting more often (and then pick the best of these solutions).

As TA is a trajectory method there is no natural way to parallelise a single restart. There are approaches (often derived from parallel applications of Simulated Annealing), but these implementations usually change the nature of TA from a trajectory into a population-based technique. A simpler course of action is to exploit the fact that the restarts are independent, and thus to allocate them to different nodes in a distributed computing environment. This has the further advantage of being very robust if a node breaks down, since the solution does not require all nodes to give a result.

Figure 6 shows the distribution of the objective function in a portfolio optimisation problem. The particular f chosen here was the Omega function introduced by Keating and Shadwick (2002); the optimisation included several constraints including sector weightings and cardinality restrictions.[8] Formally, f is given by

$$\frac{\int_{-\infty}^{r_d}(r_d - r)g(r)\mathrm{d}r}{\int_{r_d}^{\infty}(r - r_d)\,g(r)\mathrm{d}r}$$

where $g(\cdot)$ is the density of the portfolio's returns. With discrete scenarios and desired return $r_d = 0$, this may be computed as

$$\frac{\sum \ell_s \mathbf{1}_{\{\ell_s > 0\}}}{-\sum \ell_s \mathbf{1}_{\{\ell_s < 0\}}}$$

where $\mathbf{1}$ is an indicator function. The figure shows the distribution of outcomes for 1000 restarts when the number of iterations was fixed at levels between 100 000 and 10 000. We see how the solutions converge towards a suspected global minimum.

Given a desired quality of the solution, we can now explicitly quantify the trade-off between iterations and restarts. In our experiments, we set the desired quality equal to the 99th quantile of the distribution of the solutions obtained with the maximum number of iterations (100 000). This objective function value, denoted f^*, was to be achieved with a probability of at least 99%. In the problem here, f^* is equal to 0.4538, as can be seen from Fig. 6. For each distribution corresponding to an optimisation run with fewer iterations, we

[8] Strictly speaking, we minimised the ratio of the lower partial moment and upper partial moment of order 1 where the threshold was a return of 0. This is not exactly the same as Omega, as the latter requires the computation for varying thresholds.

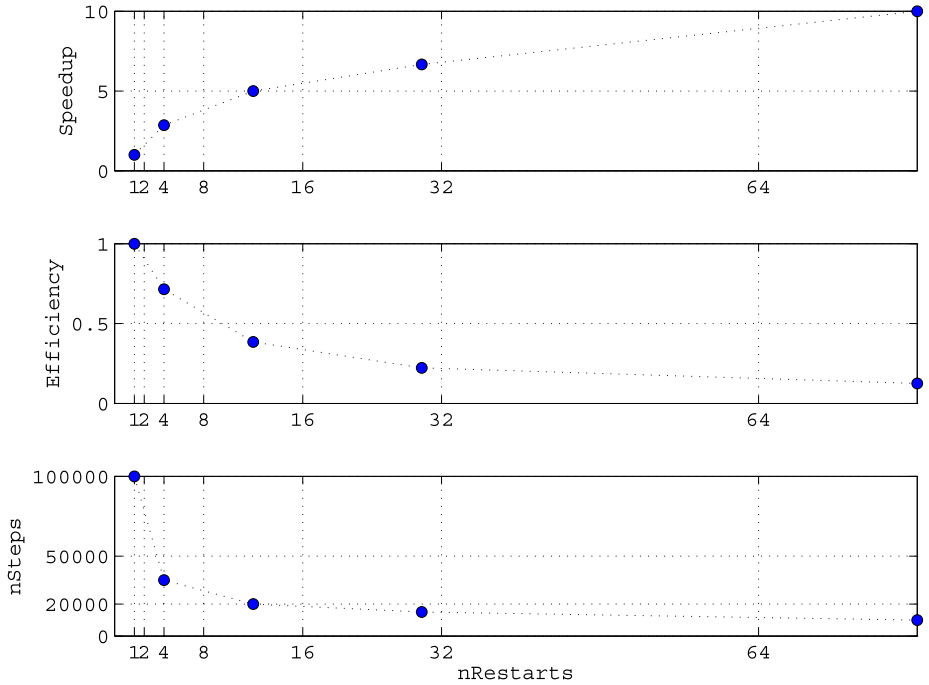


Fig. 7 Speedup (*upper panel*), efficiency (*middle panel*) and required iterations (*lower panel*) as a function of the number of nodes

can now compute the number of restarts n_{Restarts} necessary to obtain at least one result below f^* with the desired probability.

Let $C_{n_{\text{Restarts}}}$ be the respective number of steps. Then the potential speedup for n_{Restarts} restarts on different processors is

$$S_{n_{\text{Restarts}}} = \frac{C_0}{C_{n_{\text{Restarts}}}}$$

where C_0 is the computational resources employed for the most expensive run, that is 100 000 steps. The efficiency $E_{n_{\text{Restarts}}}$ is given by

$$E_{n_{\text{Restarts}}} = \frac{S_{n_{\text{Restarts}}}}{n_{\text{Restarts}}}.$$

The results are shown in Fig. 7. For example, with 16 nodes the number of iterations is decreased to below 20 000, which corresponds to a speedup of more than 5.

5 A case study—portfolio optimisation

In this section we move from the general description—the 'recipe'—to the implementation. In particular, we will discuss how Threshold Accepting (TA) can be used to select portfolios, a problem introduced in Sect. 2. For more details, the reader is referred to Gilli and Këllezi

(2002a), Gilli et al. (2006), Gilli and Schumann (2010b), Gilli et al. (2011). The optimisation algorithms, written in Matlab 2007a, can be downloaded from http://comisef.eu.

The optimisation procedure described here is based on scenarios (Dembo 1991; Hochreiter 2008), and can thus be split into two stages. The first stage is the creation of the scenarios, the second one is to find the optimal weights given the set of scenarios. As we want to concentrate on the optimisation here, the first stage will only be discussed briefly. But for an actual application both stages deserve equal attention (see the example in Gilli et al. 2010). In fact, the easiest way to obtain scenarios is to consider every historical return as one scenario, hence explicitly modelling the data is not necessary for the algorithm. There is, however, evidence that the method of scenario creation considerably influences the out-of-sample performance of selected portfolios. The critical decision is how much non-sample information we are willing to impose on the scenario-generation process. Many approaches rely on resampling; simple bootstrapping for instance keeps cross-sectional dependencies without making assumptions about their functional form, but it destroys dependencies over time.[9] If we are willing to assume a data-generating process (DGP) for either or both cross-sectional and serial dependence, we can estimate this DGP and resample from the residuals of the model. Alternatively, bootstrapping whole blocks is possible and does not require a DGP to be specified.

Here we construct simple regression models for the returns:

$$r_{it} = \alpha_i + \beta_i\, r_{Mt} + \epsilon_{it}, \quad i = 1, \ldots, n_A \tag{3}$$

where r_{it} is the return of asset i in period t, r_{Mt} is the return of a suitable index in t, and ϵ_{it} is the remaining error. After estimating α and β for each of the n_A assets, one can resample from the index-returns and from the residuals to obtain new return scenarios. Thus, for the application here, we only model the cross-section of returns. Objective functions based on drawdown, for instance, would require a different approach. More complex models, including higher order terms and different regressors, can be used.

Before we implement TA we need to decide on four issues: the objective function, a neighbourhood, an acceptance criterion for new solutions (i.e., a threshold sequence), and a stopping criterion.

5.1 The objective function

The objective function f is, at least conceptually, not problematic, but is given by the problem at hand. In our case, f could be a risk or performance measure like the ratio of lower semi-variance to the upper semi-variance, to be minimised.

Heuristics are computationally intensive. The main part of running time is usually spent on evaluating the objective function. In practice it therefore often pays (in terms of reduced computing time) to analyse and profile the objective function extensively. Sometimes, the objective function can also be updated; then it does not have to be evaluated completely for a new solution, but certain results from prior iterations can be reused. To give a concrete example, assume there are n_A assets and n_S price scenarios stored in an $n_S \times n_A$ matrix P. The matrix P can, for a given portfolio x, be transformed into losses by

$$\ell = v_0 \iota - Px,$$

[9] This weakness of bootstrapping is sometimes pointed out where it is not appropriate. If, for example, one-period optimisation is considered in which the optimisation criterion does not take into account the intertemporal dependencies, there seems little need to model them. In general, the scenario generation methods only needs to capture the aspects of the data which are relevant for the given objective.

where ι is a vector of ones. f is then a function of ℓ; clearly, given a scenario set P, any function f can easily be evaluated. Note that we will treat a solution simply as a vector x. If the universe of assets is large, and the number of assets in the portfolio comparatively small, it may be more practical to store only one 'short' vector x' which stores non-zero holdings, and a second vector that identifies the assets.

Assume an algorithm started with an initial random portfolio x^{c} and now has to evaluate x^{n}. This means that the product Px^{c} has already been computed. As will be discussed below, a new portfolio will be created by a small perturbation of the original portfolio, hence

$$x^{\mathrm{n}} = x^{\mathrm{c}} + x^{\Delta}$$

where x^{Δ} is a vector with few nonzero elements (usually only two). Then

$$Px^{\mathrm{n}} = P(x^{\mathrm{c}} + x^{\Delta}) = \underbrace{Px^{\mathrm{c}}}_{\text{known}} + Px^{\Delta}.$$

Since many elements of x^{Δ} are zero, the relevant part of P consists only of a few columns. Hence, creating a matrix P_* that only stores the columns where x^{Δ} is nonzero, and a vector x_*^{Δ} that consists only of the nonzero elements of x^{Δ}, the matrix computation Px^{Δ} can often be sped up considerably by replacing it by $P_* x_*^{\Delta}$. In essence, this makes the time required to evaluate the objective function independent of the number of assets.

5.2 The neighbourhood function

To move from one solution to the next, we need to define a neighbourhood from which new candidate solutions are chosen. In portfolio optimisation, there exists a very natural way to create neighbour solutions: pick one asset in the portfolio randomly (this may also be the cash position), 'sell' a small quantity of this asset, and 'invest' the amount obtained in another asset. If short positions are allowed, the chosen asset to be sold does not have to be in the portfolio. The small quantity may either be a random number, or a fixed fraction (say, 0.2%). Experiments suggest that, for practical purposes, both methods give similar results.

5.3 The threshold sequence

Having defined the neighbourhood, we can set the thresholds. Both elements of TA are strongly connected. Larger neighbourhoods, which imply larger changes from one candidate solution to the next, should be accompanied by larger initial threshold values, et vice versa. Winker and Fang (1997) suggest a data-driven method to obtain the thresholds; here we apply a variation of this approach as used in Gilli et al. (2006). The basic idea is to take a random walk through the data in which the steps are made according to the neighbourhood definition. At every iteration, the changes in the objective function value are recorded. The thresholds are then a number of decreasing quantiles of these changes. Algorithm 8 summarises the procedure.

Many variations are possible. For example, the algorithm here uses equally-spaced quantiles (e.g., for $n_{\mathrm{Rounds}} = 5$, the quantiles used are the 80th, 60th, 40th, 20th and 0th).[10] There

[10]Software packages like Matlab or R use the convention that the 0th quantile equals the minimum of the sample. Hence the last threshold is not necessarily 0.

Algorithm 8 Pseudocode for computation of threshold sequence

Randomly choose $x^{\mathrm{c}} \in \mathcal{X}$
for $i = 1 : n_{\mathrm{Deltas}}$ **do**
  Compute $x^{\mathrm{n}} \in \mathcal{N}(x^{\mathrm{c}})$ and $\Delta_i = |f(x^{\mathrm{c}}) - f(x^{\mathrm{n}})|$
  $x^{\mathrm{c}} = x^{\mathrm{n}}$
end for
Compute empirical distribution F of Δ_i, $i = 1, \ldots, n_{\mathrm{Deltas}}$
Compute threshold sequence $\tau_r = F^{-1}\left(\frac{n_{\mathrm{Rounds}} - r}{n_{\mathrm{Rounds}}}\right)$, $r = 1, \ldots, n_{\mathrm{Rounds}}$

is some evidence that the efficiency of the algorithm can be increased by setting a low starting quantile, say, the 50th. In general, however, TA is robust to different settings of these parameters.

5.4 Constraints

There are several generic approaches to include constraints into the optimisation. A first one used here is to create new solutions in such a way that they conform with the given constraints. The budget constraint for example is automatically enforced by the specification of the neighbourhood. Cardinality constraints can be implemented in this way as well. An alternative technique is to implement restrictions by penalty terms. If a constraint is violated, the objective function is increased by an amount that increases with the magnitude of the violation. The penalty term often also increases over time, so to allow the algorithm to move relatively freely initially.

In general, penalising the objective function to enforce constraints has several advantages. First, the computational architecture has not to be changed, since one only has to add a scalar to the objective function. Second, the approach works very well if the search space is not convex, or even disconnected. A final advantage is that penalties allow to incorporate soft constraints which can sometimes be more appropriate then hard ones.

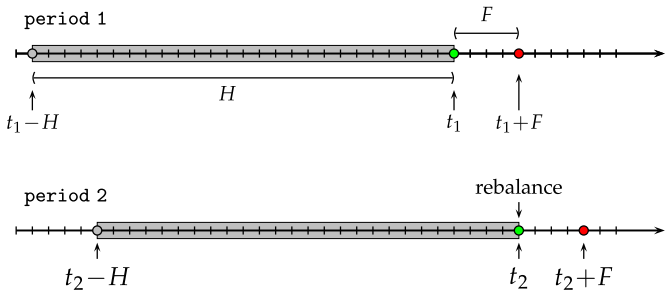
5.5 The stopping criterion

A stopping criterion is introduced by setting a fixed number of steps.

5.6 Results

In an empirical application we used a data set provided by DynaGest S.A., an investment firm domiciled in Geneva. The data set consists of daily return observations of stock prices of 600 European companies (the DJ Euro Stoxx universe), all EUR-denominated, starting in January 1999. The time series were filtered for sufficient market capitalisation (no small caps); transaction costs were applied at any rebalancing date. Given the data, we did a rolling-window backtest for various objective functions. Thus we optimised the model at point in time t_1, utilising data from $t_1 - H$ to $t_1 - 1$, where H was usually set to around 250 days (one year). Then, the resulting portfolio was held until $t_1 + F$, with F set to 3 months. At this point, a new optimisation took place, using data from $t_2 - H$ until $t_2 - 1$, holding the portfolio until $t_2 + F$, and so on. In other words, we contructed a portfolio using data from the last year, held the portfolio for three months, and then rebalanced. Altogether,

we split the data set in 35 subperiods. Such a moving-forward scheme is illustrated in the following picture.



We tested a large number of alternative portfolio selection models with objective functions based on quantiles, partial moments, and other non-standard performance and risk measures. The purpose of the study was to see whether alternative portfolio selection models can be valuable alternatives to mean–variance, and it seems they are. Detailed results are provided in Gilli and Schumann (2011). This in turn provides evidence for the usefulness of heuristics, since none of the models tested (other than equal weight and mean–variance which where used as benchmarks) could have been solved with classical optimisation methods.

But before we present some results, we would like to come back to the problem discussed in the Introduction: how do we evaluate a solution in this setting? We have written down an optimisation model, actually many different models, and computed their solutions. But that was not our goal: we rather wanted to see if the alternative models really yield better solutions to the actual problem (finding well-performing portfolios). We nevertheless start from the numerical solution.

As discussed in the last section, heuristics are stochastic methods, hence the result of a single run can be treated as the realisation of a random variable. Generally, there is a trade off between search time (computing time) and the quality of a solution (see Fig. 6). We search for longer, we find (on average) better solutions. But we need to state carefully what we mean by 'finding a better solution' since there are actually two relationships that we look at: if we search for longer, we should find better solutions to our model, that is, we have a trade-off between computing time and quality of the in-sample solution. Second, we would hope, better solutions to the model should result in better solutions to our actual problem (e.g., selecting a portfolio that yields a high risk-adjusted return); the simplest way to see if this is true is to look at the out-of-sample performance of a portfolio.

The first relationship (more iterations result in better in-sample solutions) seems reasonable, though to be practically useful we need be more concrete; we need to run experiments to estimate the speed with which solutions improve. But the second trade-off is not at all clear. There is no reason to assume that there exists a positive relationship between in-sample and out-of-sample quality. We can give an example well-known in portfolio optimisation. If we wanted to maximise return, we should not write it without safeguards into our objective function because there is overwhelming evidence that we cannot predict future returns, and there is a cost involved in failing to correctly predict: the chosen portfolio performs poorly.[11]

[11] This implies that possibly we do not want the best possible solution for a given model. From an optimisation viewpoint this is awkward: our model is clearly misspecified. Thus we should adjust the model such that we are interested in its optimum. Agreed; but how can we know? Only by testing.

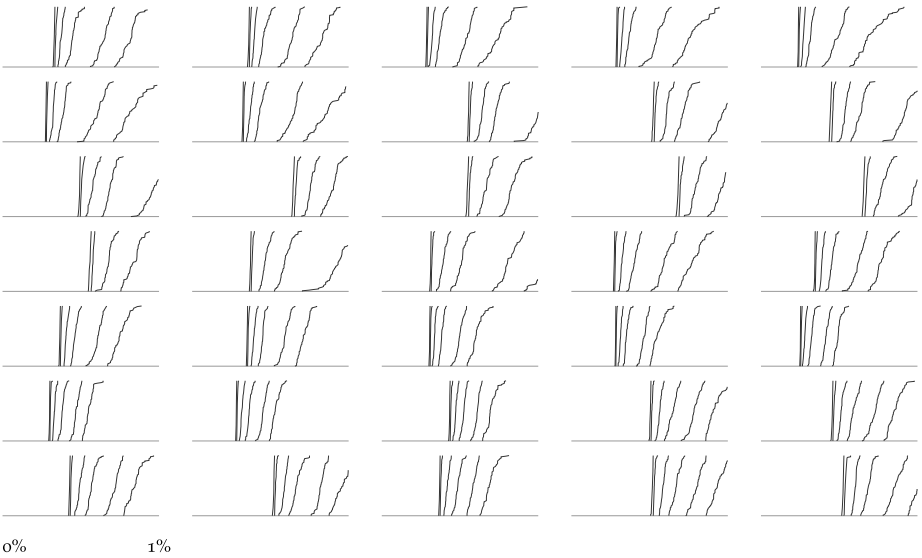


Fig. 8 Distributions of in-sample risk for 35 subperiods. *All diagrams* all scaled the same way; all x-axes run from zero to 1% daily risk. Each distribution comes from 100 restarts of TA. Iterations were set to 1 (random portfolios), 1000, 5000, 15 000, 50 000 and 100 000 steps, respectively. With more iterations, the distributions move to the left and become steeper

So we need to take three steps: (i) demonstrate that our optimisation heuristic is capable of solving our model, (ii) show that there is a meaningful relationship between the model's solution and the actual problem's solution (i.e., compare in-sample and out-of-sample quality of the solution), and (iii) finally, see whether alternative models really provide advantages over mean–variance models. Results for step (i) and (ii) were performed and reported in Gilli and Schumann (2010d); results for step (iii) can be found in Gilli and Schumann (2011). Here we briefly summarise their findings.

5.6.1 Solving the model

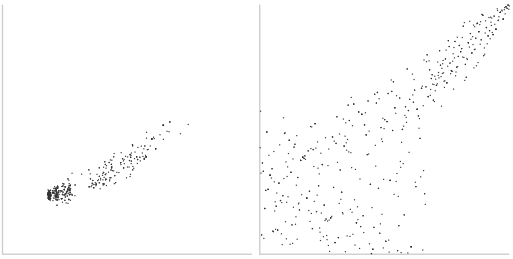
We look into a particular problem: minimise the square root of semi-variance, subject to (i) the number of assets in the portfolio must be between 20 and 50, and (ii) any weight of an included asset must lie between 1% and 5%. We tested random portfolios, and portfolios obtained after 1000, 5000, 15 000, 50 000 and 100 000 steps. Figure 8 shows this in-sample convergence for the 35 successive subperiods of our problem; compare with Fig. 6 from the previous section.

All figures show distribution functions (i.e., estimates of $\mathcal{D}$) of the realised objective function values (i.e., in-sample risk), each estimated from 100 restarts with a given number of iterations. All figures have the same scaling. The x-axis shows a range from zero risk to 1% daily risk. We see that when we increase the numbers of steps, the distributions move to the left, to finally settle around a specific point (the range among the solutions computed with 100 000 iterations is often less than 0.01%). The actual position of this point varies over the periods since market conditions change. Of course, we cannot know if we have found the global minimum, but TA seems to consistently find 'good' solutions. So TA seems capable of solving our model.

5.6.2 *In-sample to out-of-sample*

Every solution (i.e., every portfolio) maps into an in-sample risk (our objective function), but also into an out-of-sample, realised risk. We can now sort our solutions by in-sample objective function value, and then see if out-of-sample quality is a roughly monotonous function of this quality, and if the slope of this function is economically meaningful. We look again at the 35 subperiods. For each period, we pool all solutions, and plot the in-sample risk (x-axis) against the out-of-sample risk (y-axis). This is shown in Fig. 9.

On the left, we plot in-sample risk against out-of-sample risk in percentage points. Out-of-sample risk is positively related to in-sample risk, even though the relationship is noisy. But except for the first subperiod, the picture is encouraging: lower in-sample risk leads to lower out-of-sample risk. In the right panel of Fig. 9 we plot the same data, but the points are replaced by their ranks. We can again see the positive relationship between in-sample and out-of-sample, yet we also see that when we move to the bottom-left corners, i.e., to the area where we have lower risk, the relationship deteriorates. Let us look at a concrete example, the sixth period; we reproduce the respective diagram from Fig. 9.



The correlation between the in-sample and out-of-sample risk is 0.94 over all solutions. But if we only look at the best 100 solutions, the correlation goes down to -0.02. In any case, these best 100 solutions vary in-sample by less than one basis point from one another (compare the sixth diagram in Fig. 8). So even if we were to find a sizable correlation, there is no more economic justification for preferring a 'better' solution over a worse one.

Gilli and Schumann (2010d) analyse this effect in detail; essentially, we can make the in-sample stochastics of the solution so small that further improvements to the model solution do not lead to predictable improvements of the problem's solution any more. Thus, optimisation makes sense, but only up to a point. Beyond this point there is no more need to further improve a solution: any in-sample improvement only leads to unpredictable (or meaningless) changes in out-of-sample performance.

5.6.3 *Out-of-sample results for different strategies*

A single wealth trajectory of a given portfolio for a given data set conceals a lot of uncertainty. First, our data (in-sample and out-of-sample) represents just one realisation of some unknown return-generating process. Further, any scenario generation method that contains stochastic elements (e.g., any resampling method) introduces uncertainty, since repeatedly running the procedure will lead to different scenario sets, and thus to different optimal portfolios. And finally, as we have seen, there is the optimisation itself: heuristics (like here TA) are mostly stochastic algorithms, hence the resulting optimal portfolios of repeated runs, even for the same scenario set, will likely differ from one another.

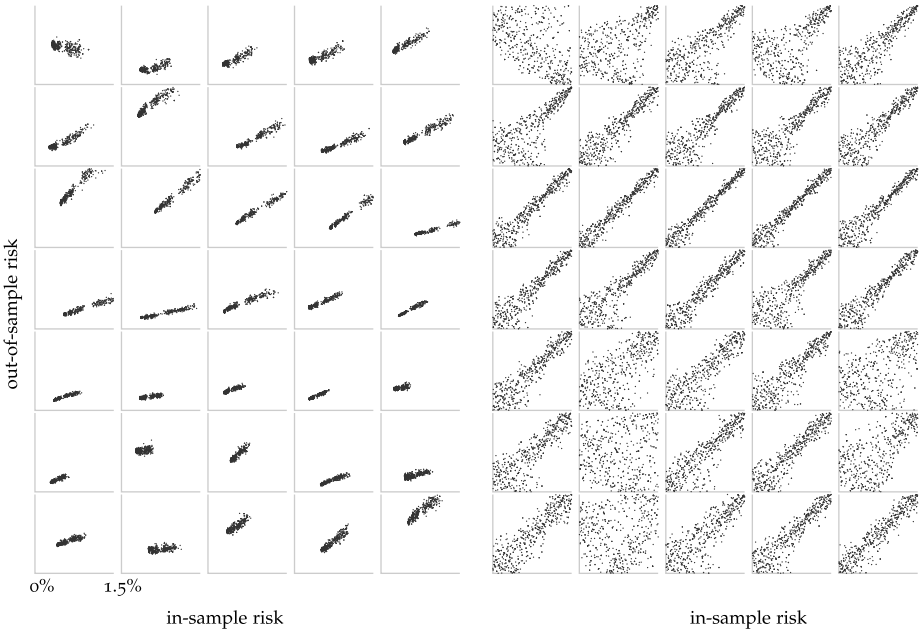


Fig. 9 *All figures* show in-sample risk (x-axis) compared with out-of-sample risk (y-axis) for 35 subperiods. The *left panels* show the actual magnitude of risk (in percentage points, all scales run from 0% to 1.5%), on the *right* we plot ranks against ranks. Lower ranks stand for lower risk, i.e., the bottom-left corner in *each plot* is the region where in-sample and out-of-sample risk is low

In finance, essentially all models are sensitive to small changes in the data (see for instance Acker and Duck 2007; Gilli and Schumann 2010c), hence we cannot ignore these uncertainties; we need robustness checks. Gilli and Schumann (2011) use a simple resampling procedure to obtain an idea of the variability of results. For each subperiod i, we

(1) take historical data from $t_i - H$ to $t_i - 1$ and delete a small number of observations (we randomly select and delete 10% of the observations),
(2) create scenarios with the linear regression model (3) and compute an optimal portfolio,
(3) compute the portfolio's out-of-sample performance from t_i to $t_i + F$.

After going through all subperiods, we have obtained one out-of-sample path for our strategy. Now we repeat this whole procedure 100 times, and thus obtain 100 out-of-sample paths. Note that the out-of-sample data are never changed. While this procedure confounds the different sources of uncertainty that we described above, it still gives a rough idea of the problem's sensitivity to changes in the data. We obtain not a single point estimate, but a collection of out-of-sample paths by which we evaluate a given strategy. Figure 10 gives annualised out-of-sample returns and Sharpe ratios for the minimum-variance portfolio (as a benchmark), the Omega ratio as defined in Sect. 4, and the Upside potential ratio (Sortino et al. 1999).

6 Conclusion

Many optimisation problems in financial modelling and estimation are difficult to solve as they exhibit multiple local optima or have discontinuities in their objective functions;

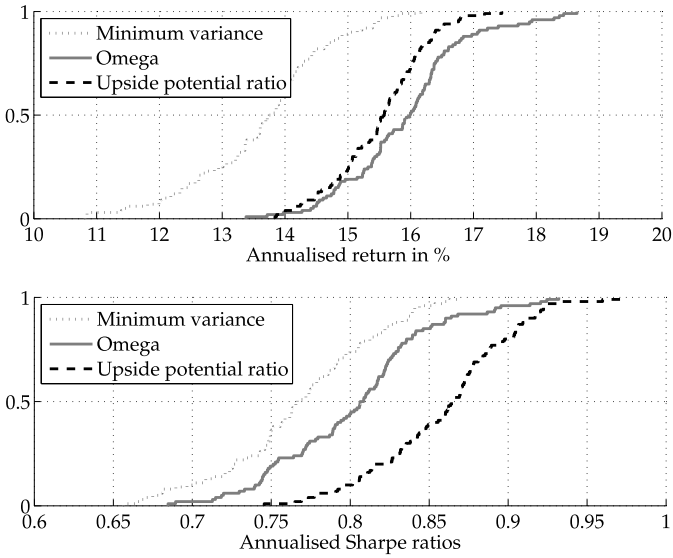


Fig. 10 Annualised returns and Sharpe ratios for Minimum-variance, Omega, and Upside potential ratio

several examples have been presented in this paper. In such cases an often observed practice is to 'convexify' the problem by making additional assumptions, simplifying the model (for instance by dropping constraints), or imposing prior knowledge (e.g., limiting the parameter domain). This makes it possible to employ classical optimisation methods, but the results obtained are not necessarily good solutions to the original problem. An alternative approach, which we have advocated in this paper, is the use of optimisation heuristics like Simulated Annealing or Genetic Algorithms. These methods are powerful enough to handle objective functions and side constraints essentially without restrictions on their functional form.

In applied financial modelling, we are faced with three questions: 'what do we want to optimise?', 'what can we estimate?', and 'what can we compute?'. The first question describes our actual problem; the second the constraints imposed by our (unfortunately very limited) capabilities to forecast financial quantities. With heuristics, the third question—'what can we compute?'—becomes much less of a constraint, and thus leaves us to deal with the actual problem and the empirical difficulties. As we said in the Introduction, optimisation is just a tool, and like with any tool, it is its application that matters.

References

Acker, D., & Duck, N. W. (2007). Reference-day risk and the use of monthly returns data. *Journal of Accounting, Auditing and Finance*, *22*(4), 527–557.

Althöfer, I., & Koschnick, K.-U. (1991). On the convergence of "Threshold Accepting". *Applied Mathematics & Optimization*, *24*(1), 183–195.

Bakshi, G., Cao, C., Chen, Z. (1997). Empirical performance of alternative option pricing models. *Journal of Finance*, *52*(5), 2003–2049.

Barr, R. S., Golden, B. L., Kelly, J. P., Resende, M. G. C., & Stewart, W. R. (1995). Designing and reporting on computational experiments with heuristic methods. *Journal of Heuristics*, *1*(1), 9–32.

Bates, D. S. (2003). Empirical option pricing: a retrospection. *Journal of Econometrics*, *116*(1–2), 387–404.

Blume, M. E. (1971). On the assessment of risk. *Journal of Finance*, *26*(1), 1–10.

Chan, L. K. C., & Lakonishok, J. (1992). Robust measurement of beta risk. *Journal of Financial and Quantitative Analysis*, *27*(2), 265–282.
Chan, L. K. C., Karceski, J., & Lakonishok, J. (1999). On portfolio optimization: forecasting covariances and choosing the risk model. *The Review of Financial Studies*, *12*(5), 937–974.
Chekhlov, A., Uryasev, S., & Zabarankin, M. (2005). Drawdown measure in portfolio optimization. *International Journal of Theoretical and Applied Finance*, *8*(1), 13–58.
Constantinides, G. M., & Malliaris, A. G. (1995). Portfolio theory. In R. A. Jarrow, V. Maksimovic, & W. T. Ziemba (Eds.), *Handbooks in operations research and management science: Vol. 9*: *Finance* (pp. 1–30). Amsterdam: North-Holland.
Cont, R. (2001). Empirical properties of asset returns: stylized facts and statistical issues. *Quantitative Finance*, *1*, 223–236.
Cont, R., & da Fonseca, J. (2002). Dynamics of implied volatility surfaces. *Quantitative Finance*, *2*, 45–60.
Dacorogna, M. M., Gençay, R., Müller, U. A., Olsen, R. B., & Pictet, O. V. (2001). *An introduction to high-frequency finance*. San Diego: Academic Press.
Dembo, R. S. (1991). Scenario optimization. *Annals of Operation Research*, *30*(1), 63–80.
Diebold, F. X., & Li, C. (2006). Forecasting the term structure of government bond yields. *Journal of Econometrics*, *130*(2), 337–364.
Dixit, A. K. (1990). *Optimization in economic theory* (2nd ed.). London: Oxford University Press.
Dueck, G., & Scheuer, T. (1990). Threshold Accepting. A general purpose optimization algorithm superior to Simulated Annealing. *Journal of Computational Physics*, *90*(1), 161–175.
Eberhart, R. C., & Kennedy, J. (1995). A new optimizer using Particle Swarm theory. In *Proceedings of the sixth international symposium on micromachine and human science*, Nagoya, Japan (pp. 39–43).
Fama, E. F., & French, K. R. (1993). Common risk factors in the returns on stocks and bonds. *Journal of Financial Economics*, *33*, 3–56.
Fama, E. F., & French, K. R. (2004). The capital asset pricing model: theory and evidence. *The Journal of Economic Perspectives*, *18*(3), 25–46.
Gaivoronski, A. A., & Pflug, G. (2005). Value-at-risk in portfolio optimization: properties and computational approach. *The Journal of Risk*, *7*(2), 1–31.
Gendreau, M., & Potvin, J.-Y. (Eds.) (2010). *Handbook of metaheuristics* (2nd ed.). Berlin: Springer.
Genton, M. G., & Ronchetti, E. (2008). Robust prediction of beta. In: E. J. Kontoghiorghes, B. Rustem, & P. Winker (Eds.), *Computational methods in financial engineering—essays in honour of Manfred Gilli*. Berlin: Springer.
Gill, P. E., Murray, W., & Wright, M. H. (1986). *Practical optimization*. Amsterdam: Elsevier.
Gilli, M., & Këllezi, E. (2002a). A global optimization heuristic for portfolio choice with VaR and expected shortfall. In: E. J. Kontoghiorghes, B. Rustem, & S. Siokos (Eds.), *Applied optimization series: Computational methods in decision-making, economics and finance* (pp. 167–183). Dordrecht: Kluwer Academic.
Gilli, M., & Këllezi, E. (2002b). The Threshold Accepting heuristic for index tracking. In P. Pardalos & V. K. Tsitsiringos (Eds.), *Applied optimization series: Financial engineering, e-commerce and supply chain* (pp. 1–18). Boston: Kluwer Academic.
Gilli, M., & Schumann, E. (2010a). Distributed optimisation of a portfolio's omega. *Parallel Computing*, *36*(7), 381–389.
Gilli, M., & Schumann, E. (2010b). Portfolio optimization with "Threshold Accepting": a practical guide. In S. E. Satchell (Ed.), *Optimizing optimization: the next generation of optimization applications and theory*. Amsterdam: Elsevier.
Gilli, M., & Schumann, E. (2010c). Optimization in financial engineering—an essay on 'good' solutions and misplaced exactitude. *Journal of Financial Transformation*, *28*, 117–122.
Gilli, M., & Schumann, E. (2010d). Optimal enough? *Journal of Heuristics*. doi:10.1007/s10732-010-9138-y.
Gilli, M., & Schumann, E. (2011). Risk-reward optimisation for long-run investors: an empirical analysis. *European Actuarial Journal*. Available from http://www.actuaries.org/Munich2009/Programme_EN.cfm.
Gilli, M., & Winker, P. (2003). A global optimization heuristic for estimating agent based models. *Computational Statistics & Data Analysis*, *42*(3), 299–312.
Gilli, M., & Winker, P. (2008). A review of heuristic optimization methods in econometrics. *Swiss finance institute research paper No. 08-12*.
Gilli, M., & Winker, P. (2009). Heuristic optimization methods in econometrics. In D. A. Belsley & E. Kontoghiorghes (Eds.), *Handbook of computational econometrics*. New York: Wiley.
Gilli, M., Këllezi, E., & Hysi, H. (2006). A data-driven optimization heuristic for downside risk minimization. *The Journal of Risk*, *8*(3), 1–18.
Gilli, M., Maringer, D., & Winker, P. (2008). Applications of heuristics in finance. In D. Seese, C. Weinhardt, & F. Schlottmann (Eds.), *Handbook on information technology in finance*. Berlin: Springer.

Gilli, M., Schumann, E., di Tollo, G., Cabej G. (2010). Constructing 130/30-portfolios with the omega ratio. *Journal of Asset Management*. doi:10.1057/jam.2010.25.
Gilli, M., Maringer, D., & Schumann, E. (2011). *Numerical methods and optimization in finance*. Amsterdam: Elsevier.
Glover, F. (1986). Future paths for integer programming and links to artificial intelligence. *Computers & Operations Research*, *13*(5), 533–549.
Glover, F. (2007). Tabu Search—uncharted domains. *Annals of Operation Research*, *149*(1), 89–98.
Glover, F., & Laguna, M. (1997). *Tabu Search*. Dordrecht: Kluwer Academic.
Hamida, S. B., & Cont, R. (2005). Recovering volatility from option prices by evolutionary optimization. *Journal of Computational Finance*, *8*(4), 1–2.
Heston, S. L. (1993). A closed-form solution for options with stochastic volatility with applications to bonds and currency options. *The Review of Financial Studies*, *6*(2), 327–343.
Hillier, F. S. (1983). Heuristics: a Gambler's roll. *Interfaces*, *13*(3), 9–12.
Hochreiter, R. (2008). Evolutionary stochastic portfolio optimization. In A. Brabazon & M. O'Neill (Eds.), *Natural computing in computational finance*. Berlin: Springer.
Holland, J. H. (1992). *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control and artificial intelligence*. Cambridge: MIT Press.
Hoos, H. H., & Stützle, T. (2004). *Stochastic Local Search: foundations and applications*. San Mateo: Morgan Kaufmann.
Ince, O. S., & Porter, R. B. (2006). Individual equity return data from Thomson datastream: handle with care! *Journal of Financial Research*, *29*(4), 463–479.
Keating, C., & Shadwick, B. (2002). An introduction to omega. *AIMA Newsletter* (April 2002).
Kirkpatrick, S., Gelatt, C. D., & Vecchi, M. P. (1983). Optimization by Simulated Annealing. *Science*, *220*(4598), 671–680.
Kirman, A. P. (1992). Whom or what does the representative individual represent? *The Journal of Economic Perspectives*, *6*(2), 117–136.
Kirman, A. P. (1993). Ants, rationality, and recruitment. *The Quarterly Journal of Economics*, *108*(1), 137–156.
Knez, P. J., & Ready, M. J. (1997). On the robustness of size and book-to-market in cross-sectional regressions. *Journal of Finance*, *52*(4), 1355–1382.
LeBaron, B. (2000). Agent-based computational finance: suggested readings and early research. *Journal of Economic Dynamics & Control*, *24*, 679–702.
LeBaron, B. (2006). Agent-based computational finance. In L. H. Tesfatsion & K. H. L. Judd (Eds.), *Handbook of computational economics* (Vol. II, pp. 1187–1233). Amsterdam: Elsevier.
Lo, A. W. (2008). *Hedge funds—an analytic perspective*. Princeton: Princeton University Press.
Luenberger, D. G. (1998). *Investment science*. Oxford: Oxford University Press.
Madan, D. B. (2001). On the modelling of option prices. *Quantitative Finance*, *1*, 481.
Manaster, S., & Koehler, G. (1982). The calculation of implied variances from the Black–Scholes model: a note. *Journal of Finance*, *37*(1), 227–230.
Maniezzo, V., Stützle, T., & Voß, S. (Eds.) (2009). *Matheuristics: hybridizing metaheuristics and mathematical programming*. Berlin: Springer.
Maringer, D. (2004). Finding the relevant risk factors for asset pricing. *Computational Statistics & Data Analysis*, *47*, 339–352.
Maringer, D. (2005). *Portfolio management with heuristic optimization*. Berlin: Springer.
Maringer, D. (2008). Risk preferences and loss aversion in portfolio optimization. In E. J. Kontoghiorghes, B. Rustem, & P. Winker (Eds.), *Computational methods in financial engineering—essays in honour of Manfred Gilli*. Berlin: Springer.
Markowitz, H. M. (1952). Portfolio selection. *Journal of Finance*, *7*(1), 77–91.
Markowitz, H. M. (1959). *Portfolio selection*. New York: Wiley.
Martin, R. D., & Simin, T. T. (2003). Outlier-resistant estimates of beta. *Financial Analysts Journal*, *59*(5), 56–69.
Michalewicz, Z., & Fogel, D. B. (2004). *How to solve it: modern heuristics*. Berlin: Springer.
Mikhailov, S., & Nögel, U. (2003). *Heston's stochastic volatility model implementation, calibration and some extensions*. Wilmott, pp. 74–79.
Mladenović, N., & Hansen, P. (1997). Variable neighborhood search. *Computers & Operations Research*, *24*(11), 1097–1100.
Moscato, P. (1989). *On evolution, search, optimization, Genetic Algorithms and martial arts—towards memetic algorithms*. Technical Report 790, CalTech California Institute of Technology.
Moscato, P., & Fontanari, J. F. (1990). Stochastic versus deterministic update in Simulated Annealing. *Physics Letters A*, *146*(4), 204–208.

Nelson, C. R., & Siegel, A. F. (1987). Parsimonious modeling of yield curves. *Journal of Business*, *60*(4), 473–489.

Pisinger, D., & Ropke, S. (2010). Large neighborhood search. In M. Gendreau & J.-Y. Potvin (Eds.), *Handbook of metaheuristics* (2nd ed.). Berlin: Springer.

Rockafellar, R. T., & Uryasev, S. (2000). Optimization of conditional value-at-risk. *The Journal of Risk*, *2*(3), 21–41.

Rousseeuw, P. J. (1984). Least median of squares regression. *Journal of the American Statistical Association*, *79*(388), 871–880.

Schlottmann, F., & Seese, D. (2004). Modern heuristics for finance problems: a survey of selected methods and applications. In S. T. Rachev (Ed.), *Handbook of computational and numerical methods in finance*. Basel: Birkhäuser.

Schrimpf, G., Schneider, J., Stamm-Wilbrandt, H., & Dueck, G. (2000). Record breaking optimization results using the ruin and recreate principle. *Journal of Computational Physics*, *159*(2), 139–171.

Sortino, F., van der Meer, R., & Plantinga, A. (1999). The Dutch triangle. *Journal of Portfolio Management*, *26*(1), 50–58.

Storn, R. M., & Price, K. V. (1997). Differential Evolution—a simple and efficient heuristic for global optimization over continuous spaces. *Journal of Global Optimization*, *11*(4), 341–359.

Svensson, L. E. O. (1994). Estimating and interpreting forward interest rates: Sweden 1992–1994. *IMF Working Paper 94/114*.

Talbi, E.-G. (2002). A taxonomy of hybrid metaheuristics. *Journal of Heuristics*, *8*(5), 541–564.

Vasicek, O. A. (1973). A note on the cross-sectional information in Bayesian estimation of security betas. *Journal of Finance*, *28*(5), 1233–1239.

Winker, P. (2001). *Optimization heuristics in econometrics: applications of Threshold Accepting*. New York: Wiley.

Winker, P., & Fang, K.-T. (1997). Application of Threshold-Accepting to the evaluation of the discrepancy of a set of points. *SIAM Journal on Numerical Analysis*, *34*(5), 2028–2042.

Winker, P., & Gilli, M. (2004). Applications of optimization heuristics to estimation and modelling problems. *Computational Statistics and Data Analysis*, *47*(2), 211–223.

Winker, P., & Maringer, D. (2007). The Threshold Accepting optimisation algorithm in economics and statistics. In E. J. Kontoghiorghes & C. Gatu (Eds.), *Advances in computational management science: Vol. 9. Optimisation, econometric and financial analysis* (pp. 107–125). Berlin: Springer.

Winker, P., Gilli, M., & Jeleskovic, V. (2007). An objective function for simulation based inference on exchange rate data. *Journal of Economic Interaction and Coordination*, *2*, 125–145.

Zanakis, S. H., & Evans, J. R. (1981). Heuristic "optimization": why, when, and how to use it. *Interfaces*, *11*(5), 84–91.