COMISEF WORKING PAPERS SERIES

WPS-007 09/02/2009

Heuristic Optimisation in Financial Modelling

M. Gilli
E. Schumann

Heuristic Optimisation in Financial Modelling[⋆]

Manfred Gilli[a,*] and Enrico Schumann[b]

[a]*Department of Econometrics, University of Geneva and Swiss Finance Institute*

[b]*Department of Econometrics, University of Geneva*

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 (eg, 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. This paper argues that an alternative approach is the application of optimisation heuristics like Simulated Annealing or Genetic Algorithms. These methods have been shown to be capable to handle non-convex optimisation problems with all kinds of constraints. To motivate the use of such techniques in finance, the paper presents 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, the paper describes 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. It is shown, again for the example from portfolio selection, how this random character of the solutions can be exploited to inform the distribution of computations.

Key words: Optimisation heuristics, Financial Optimisation, Portfolio Optimisation *JEL codes*: C61, C63, G11
First version: 2008-09-30. *Last changes*: 2009-02-07.

[⋆] Both authors gratefully acknowledge financial support from the EU Commission through MRTN-CT-2006-034270 COMISEF.

[*] Corresponding author: Department of Econometrics, University of Geneva, Bd du Pont d'Arve 40, 1211 Geneva 4, Switzerland. Tel.: + 41 22 379 8222; fax: + 41 22 379 8299.

Email addresses: Manfred.Gilli@unige.ch (Manfred Gilli), Enrico.Schumann@unige.ch (Enrico Schumann).

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 usually require convexity and relatively well-behaved objective functions as they are often based on gradients or similar 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 that will be described 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. 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 accomplished very easily. These advantages come at a cost as well, as the obtained solution is only a 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. 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.

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 only 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, seems just as important as how to optimise. For example, when fitting models to historical financial time series, one may find 'spikes' in the objective functions which 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, ch. 11).) Possible solutions, which may include more emphasis on data modelling or incorporating alternative, more robust objective functions, are in our view very often 'under-researched', in particular when it comes to their actual empirical performance. The advantage of heuristics here is that when formulating the model, the analyst is quite unconstrained with regard to tractability of the model, hence emphasis can be put on the empirical merits of specific methods. Optimisation is only a tool; it is the application of this tool that matters.

This paper is not a survey, but a selection of problems from finance to illustrate and motivate the use of heuristic methods in this field. For more detailed studies, see for example to Maringer (2005); Schlottmann and Seese (2004) or Gilli *et al.* (2008) give more references to specific applications. The remaining paper is organised as follows. Section 2 will detail several examples of problems that arise in theoretical and applied finance; Section 3 will give a brief introduction to heuristic methods which can be applied to solve these problems. The emphasis will be on principles, rather than on details. Heuristic methods as presented here may be regarded as 'recipes' rather than specific algorithms.[1] To demonstrate the process of implementing such a recipe, in Section 4 it will be shown how to apply a particular heuristic method, Threshold Accepting, to a portfolio selection problem. Section 5 then discusses an important aspect of heuristics, namely the stochastic nature of the obtained solutions. Section 6 concludes.

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

2 Optimisation problems in finance

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.[2]

Assume an investor has wealth v_0 and wishes to invest for one period. A given portfolio, as it comprises risky assets, maps into a distribution of wealth at the end of the period, or, equivalently, into a distribution of losses ℓ. The optimisation problem can be stated as follows

$$\begin{aligned}
&\min_{x \in \mathbb{R}^{n_{\mathrm{A}}}} \ \Phi(\ell) \\[1ex]
&E(\ell) \leq -v_0\, r_d \\
&x_j^{\inf} \leq x_j \leq x_j^{\sup} \qquad j \in \mathcal{J} \\
&K_{\inf} \leq \#\{\mathcal{J}\} \leq K_{\sup} \\
&\ldots
\end{aligned}$$

The objective function, $\Phi(\ell)$, is a risk measure or a combination of multiple objectives to be minimised. Candidates include the portfolio's drawdown, partial moments, or whatever the analyst wishes to optimise. The vector x stores the (integer) numbers of assets held; r_d is the desired return. $x_j^{\inf}$ and $x_j^{\sup}$ are vectors of minimum and maximum holding sizes, respectively, for those assets included in the portfolio (ie, those in $\mathcal{J}$). (If short-sales are allowed, this constraint should rather read $x_j^{\inf} \leq |x_j| \leq x_j^{\sup}$.) $K_{\inf}$ and $K_{\sup}$ are cardinality constraints which 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 (ie, restrictions on the multiples of assets that can be traded). One may also add constraints that under certain market conditions, the portfolio needs to behave in a certain way (usually give a required minimum return).

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

Similar to this framework are *index tracking* problems where investors try to replicate a pre-defined benchmark (see for example Gilli and Këllezi (2002*b*)). This benchmark need 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 using 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 very often not convex, in particular under reasonable constraints (like cardinality restrictions). To give an example, Figure 1 shows the search space, that is the values of the objective function that particular solutions map to, for a particular problem where Φ is the portfolio's Value-at-Risk (`VaR`). As can be seen, the surface is not convex and not smooth. Any search that requires a globally convex solution, like gradient-based methods, will stop at the first local minimum encountered.

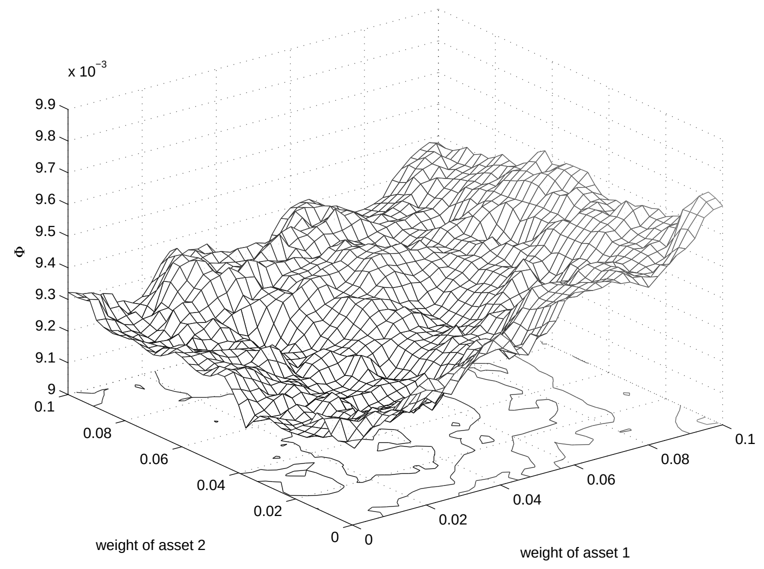


Fig. 1. Objective function for `VaR`.

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) (see also `www.aorda.com`). Unfortunately, such solutions are usually 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 Section 4.

2.2 Model selection

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

$$r = \begin{bmatrix} f_1 & \cdots & f_k \end{bmatrix} \begin{bmatrix} \beta_1 \\ \vdots \\ \beta_k \end{bmatrix} + \epsilon \qquad (1)$$

where r is a vector of returns for a given asset, f are the vectors of factor realisations, β are the factor loadings and ϵ contains the residual. Such models are widely applied in practice, for instance to construct covariance matrices or in attempts for forecast future returns. The factors f 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 one may also consider lagged variables (Maringer, 2004). Hence, model selection becomes a critical issue, as one wishes 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 smaller 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, ch. 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 very 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). Some 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, section 5.2). A particular search space is shown in Figure 2.

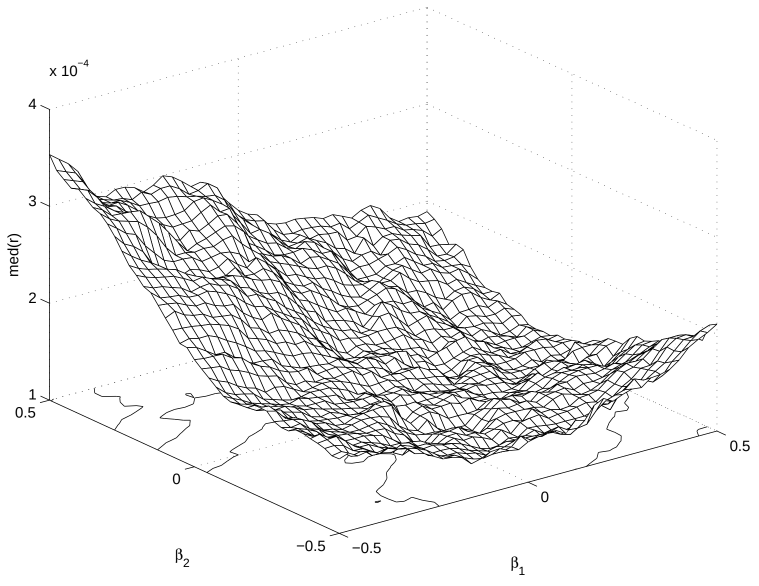


Fig. 2. LMS objective function.

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 which 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 principal produce realistic market behaviour when parameters (like preferences of agents) are specified appropriately. This leads to the question what appropriate values should be like, and how different models compare with one other 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.[3] Figure 3 shows the resulting search space for a particular ABM (see Kirman (1993)). The objective function does 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 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 one tries to obtain 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 which can be done efficiently by utilising 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

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

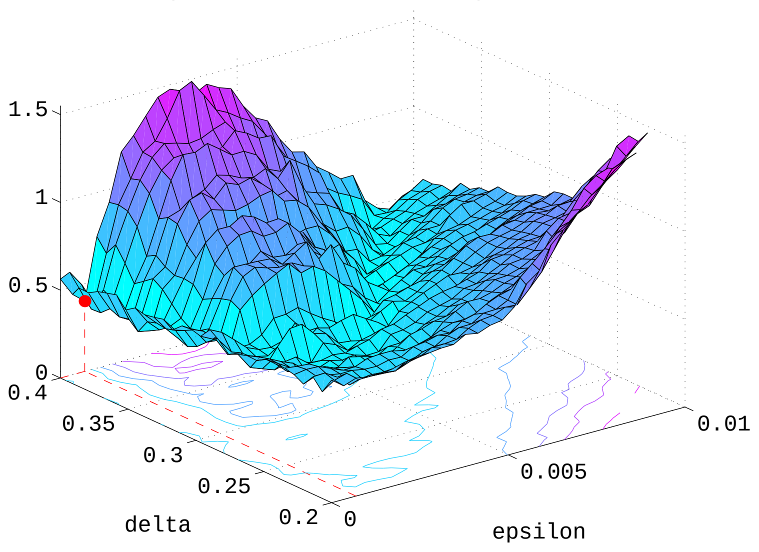


Fig. 3. Simulated objective function for Kirman's model for two parameters.

surface (Cont and da Fonseca, 2002). Calibrating these models generally requires to set more parameters, and often leads to more difficult optimisation problems.

One particular pricing model is the Heston model (Heston, 1993) which is often used as it gives closed-form solutions for option prices. Under the Heston model the stock price (S) and its variance (V) dynamics are described by

$$\begin{aligned} dS_t &= \mu S_t dt + \sqrt{V_t} S_t dW_t^1 \\ dV_t &= \kappa(\theta - V_t)dt + \sigma\sqrt{V_t} dW_t^2 \end{aligned}$$

where the two Brownian motion processes are correlated, that is $dW_t^1 dW_t^2 = \rho dt$. As can be seen from the second equation, volatility is mean-reverting in the Heston model. In total, the model requires the specification of 6 parameters (Mikhailov and Nögel, 2003). Even though some of these parameters could be estimated from the time series of the underlying, the general approach to fit the model is to minimise the squared difference between the theoretical and observed prices. Hence the 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). (Sometimes the optimisation model also includes parameter restrictions, for example to enforce the parameters to be such that the volatility cannot become negative.)

Figure 4 shows the resulting objective function values for two parameters (volatility of volatility and mean-reversion speed) with the remaining ones fixed. As can be seen, in certain parts of the parameter domain the resulting objective function is not convex, hence standard methods may not find the global minimum.

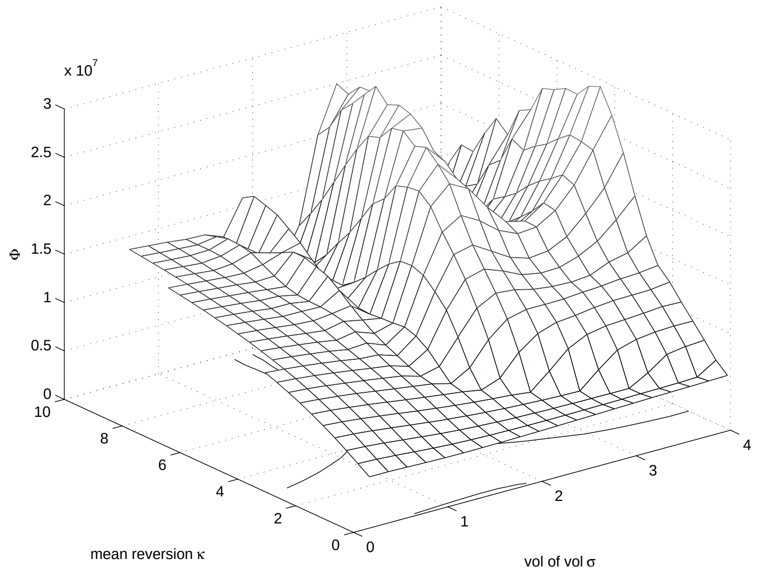


Fig. 4. Heston model objective function.

3 Heuristic optimisation methods

3.1 Definition & principles

The term 'heuristic' is used in many fields of science for different, though related, purposes. To characterise the term as used for optimisation techniques, Winker and Maringer (2007), following Barr *et al.* (1995), suggest several criteria:

- The method method should give a 'good' stochastic approximation of the true optimum.
- It 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.

A large and growing number of methods meets these demands. Still, it is interesting to note that even though there exists considerable evidence for the good performance of such methods, they are still not widely applied in research and practice.[4]

Heuristics maybe be divided into local search methods and constructive methods (Gilli and Winker, 2008). 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 sometimes not be taken too literally, as some methods (eg, 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, that is 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 where 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 simplify the presentation, we use a rather coarse classification scheme that only differentiates between *trajectory* methods and *population-based* methods, both terms to be explained below. For a more complete classification system of heuristics, see Winker and Gilli (2004).

The concept of 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.*, 2004, 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 often 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 its stopping criterion. The latter may simply be a preset number of steps. Algorithm 1 describes the procedure.

Algorithm 1 Pseudocode for Local Search.

```
1: Initialise $n_{\mathrm{Steps}}$
2: Randomly generate current solution $x^{\mathrm{c}} \in \mathcal{X}$
3: for $i = 1 : n_{\mathrm{Steps}}$ do
4:     Generate $x^{\mathrm{n}} \in \mathcal{N}(x^{\mathrm{c}})$ and compute $\Delta = f(x^{\mathrm{n}}) - f(x^{\mathrm{c}})$
5:     if $\Delta < 0$ then $x^{\mathrm{c}} = x^{\mathrm{n}}$
6: end for
7: $x^{\mathrm{sol}} = x^{\mathrm{c}}$
```

[4] Brandimarte (2006), a well-known textbook on financial optimisation, for example devotes about 8 pages, of a total of more than 600 pages, to describe heuristic methods.

In a convex 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. Unfortunately, for problems with many local minima, a local search will 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 this 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 some 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). Algorithm 2 gives the pseudocode of the procedure.

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 $x^n \in \mathcal{N}(x^c)$    (neighbour to current solution)
5:         Compute $\triangle = f(x^n) - f(x^c)$ and generate $u$ (uniform random variable)
6:         if $(\triangle < 0)$ or $(e^{-\triangle/T} > u)$ then $x^c = x^n$
7:     end while
8:     Reduce $T$
9: end for
```

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 ($\triangle < 0$), 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 $R_{\max}$.

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, both SA and TA are sometimes referred to as *threshold methods*.

Algorithm 3 Pseudocode for Threshold Accepting.

```
 1: Initialise $n_{\text{Rounds}}$ and $n_{\text{Steps}}$
 2: Compute threshold sequence $\tau_r$
 3: Randomly generate current solution $x^{\text{c}} \in \mathcal{X}$
 4: for $r = 1 : n_{\text{Rounds}}$ do
 5:     for $i = 1 : n_{\text{Steps}}$ do
 6:         Generate $x^{\text{n}} \in \mathcal{N}(x^{\text{c}})$ and compute $\Delta = f(x^{\text{n}}) - f(x^{\text{c}})$
 7:         if $\Delta < \tau_r$ then $x^{\text{c}} = x^{\text{n}}$
 8:     end for
 9: end for
10: $x^{\text{sol}} = x^{\text{c}}$
```

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 Section 4 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 (eg, 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.[5] TS is described in 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.

[5] One may of course easily alter the algorithm (line 4) to choose a neighbour solution randomly.

Algorithm 4 Pseudocode for Tabu Search.

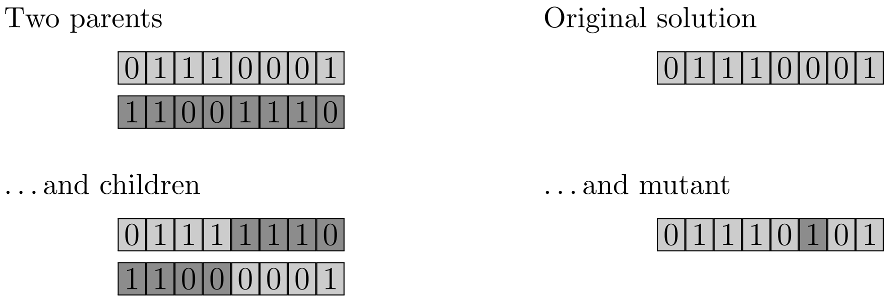
```
1: Generate current solution $x^{\mathrm{c}}$ and initialise tabu list $T = \emptyset$
2: while stopping criteria not met do
3:     Compute $V = \{x | x \in \mathcal{N}(x^{\mathrm{c}})\} \backslash T$
4:     Select $x^{\mathrm{n}} = \min(V)$
5:     $x^{\mathrm{c}} = x^{\mathrm{n}}$  and  $T = T \cup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

3.3 Population-based methods

3.3.1 Genetic Algorithms

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. This property of population-based methods also makes them good at exploration, that is they often work well for large search spaces (eg, in combinatorial problems).

Probably the best-known technique in this category are Genetic Algorithms (GA), first described by Holland in the 1970s (Holland, 1992); pseudocode can be found in Algorithm 5. GA are inspired by evolutionary biology, hence the procedure starts appropriately with a whole population of solutions; the objective function becomes a fitness function. In standard GA, solutions are coded as binary strings, that is like 0|1|1|1|0|0|0|1. Such a string may be a binary representation of an integer or real number; in many discrete problems there is an even more ‘natural’ interpretation: for instance 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 (ie, mixing existing solutions) and mutation (ie, randomly changing components of solutions), as illustrated here:



a) Crossover b) Mutation

To keep the population size $\#\{P\}$ constant, at the end of each iteration (here called 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.

Algorithm 5 Pseudocode for Genetic Algorithms.

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

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 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 `F`, between two other solutions to a third solution. Then an elementwise crossover takes place with probability `CR` between this 'auxiliary' solution $P^{(v)}_{\cdot,i}$ and the existing solution $P^{(k)}_{\cdot,i}$. If this final candidate solution $f(P^{(u)}_{\cdot,i})$ is better than $P^{(k)}_{\cdot,i}$, it replaces it; if not, the old solution is kept.

Algorithm 6 Pseudocode for Differential Evolution.

```
 1: Initialise parameters $n_{\mathrm{P}}$, $n_{\mathrm{G}}$, F and CR
 2: Initialise population $P^{(1)}_{j,i}$, $j = 1, \ldots, d$, $i = 1, \ldots, n_{\mathrm{P}}$
 3: for $k = 1$ to $n_{\mathrm{G}}$ do
 4:    $P^{(0)} = P^{(1)}$
 5:    for $i = 1$ to $n_{\mathrm{P}}$ do
 6:       Generate $r_1, r_2, r_3 \in \{1, \ldots, n_{\mathrm{P}}\}$, $r_1 \neq r_2 \neq r_3 \neq i$
 7:       Compute $P^{(v)}_{\cdot,i} = P^{(0)}_{\cdot,r_1} + \mathtt{F} \times (P^{(0)}_{\cdot,r_2} - P^{(0)}_{\cdot,r_3})$
 8:       for $j = 1$ to $d$ do
 9:          if $u < \mathtt{CR}$ then $P^{(u)}_{j,i} = P^{(v)}_{j,i}$ else $P^{(u)}_{j,i} = P^{(0)}_{j,i}$
10:       end for
11:       if $f(P^{(u)}_{\cdot,i}) < f(P^{(0)}_{\cdot,i})$ then $P^{(1)}_{\cdot,i} = P^{(u)}_{\cdot,i}$ else $P^{(1)}_{\cdot,i} = P^{(0)}_{\cdot,i}$
12:    end for
13: end for
```

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 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.

Algorithm 7 Pseudocode for Particle Swarm.

```
 1: Initialise parameters $n_\mathrm{P}$, $n_\mathrm{G}$ and $c$
 2: Initialise particles $P_i^{(0)}$ and velocity $v_i^{(0)}$, $i = 1, \ldots, n_\mathrm{P}$
 3: Evaluate objective function $F_i = f(P_i^{(0)})$, $i = 1, \ldots, n_\mathrm{P}$
 4: $Pbest = P^{(0)}$, $Fbest = F$, $Gbest = \min_i(F_i)$, $gbest = \mathrm{argmin}_i(F_i)$
 5: for $k = 1$ to $n_\mathrm{G}$ do
 6:    for $i = 1$ to $n_\mathrm{P}$ do
 7:       $\triangle v_i = c\,u\,(Pbest_i - P_i^{(k-1)}) + c\,u\,(Pbest_\mathrm{gbest} - P_i^{(k-1)})$
 8:       $v_i^{(k)} = v^{(k-1)} + \triangle v_i$
 9:       $P_i^{(k)} = P_i^{(k-1)} + v_i^{(k)}$
10:    end for
11:    Evaluate objective function $F_i = f(P_i^{(k)})$, $i = 1, \ldots, n_\mathrm{P}$
12:    for $i = 1$ to $n_\mathrm{P}$ do
13:       if $F_i < Fbest_i$ then $Pbest_i = P_i^{(k)}$ and $Fbest_i = F_i$
14:       if $F_i < Gbest$ then $Gbest = F_i$ and $gbest = i$
15:    end for
16: end for
```

4 A case study—portfolio optimisation

In this section we move from the 'recipe' (the meta-heuristic) to the implementation to solve an actual problem. In particular, it will be discussed how Threshold Accepting (TA) can be used to select optimal portfolios, a problem introduced in Section 2. For more details, the reader is referred to Gilli and Këllezi (2002*a*); Gilli *et al.* (2006). The optimisation algorithms, written in Matlab 2007a, can be downloaded from `www.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 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. It needs to be stressed, though, that for empirical applications both stages probably demand equal attention. 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 one is 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.[6] If one is willing to assume a data-generating process (DGP) for either or both cross-sectional and serial dependence, one 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, for example

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

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. More complex models, including higher order terms and different regressors, can be used.

Being a local search method, implementing TA requires the analyst to specify four issues: The objective function, a neighbourhood, an acceptance criterion for new solutions (ie, a threshold sequence), and a stopping criterion.

4.1 The objective function

The objective function Φ is, at least conceptually, not problematic, but is given by the problem at hand. In our case, Φ may be the average drawdown, or the the ratio of lower semi-variance to the upper semi-variance, both to be minimised. For a given set of scenarios and a given portfolio, this function can easily be computed.

[6] This weakness of bootstrapping is sometimes pointed out where it is not appropriate. If, for example, one-period optimisation is considered where 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.

Most heuristics methods 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 more 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

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

where ι is a vector of ones. Φ is then a function of ℓ.

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

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

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 $P_* x^{\Delta}_*$.

4.2 The neighbourhood function

To move from one to solution to the next, one needs 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 relatively similar results.

4.3 The threshold sequence

Having defined the neighbourhood, one 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 have a random walk through the data where 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.

Algorithm 8 Pseudocode for computation of threshold sequence.

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

Many variations are possible. For example, the algorithm here uses equally-spaced quantiles (eg, for $n_{\text{Rounds}} = 5$, the quantiles used are the 80th, 60th, 40th, 20th and 0th[7]). There is some evidence that the efficiency of the algorithm can be increased by setting the starting quantile lower, say, the 50th. In general, however, TA seems to be very robust to different settings of these parameters.

4.4 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.

[7] 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.

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.

4.5 The stopping criterion

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

4.6 A few results

In an empirical application we used data given to us by DynaGest S.A., an investment firm domiciled in Geneva. The data set consists of daily return observations of stock prices of more than 500 European companies, all EUR-denominated. 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.

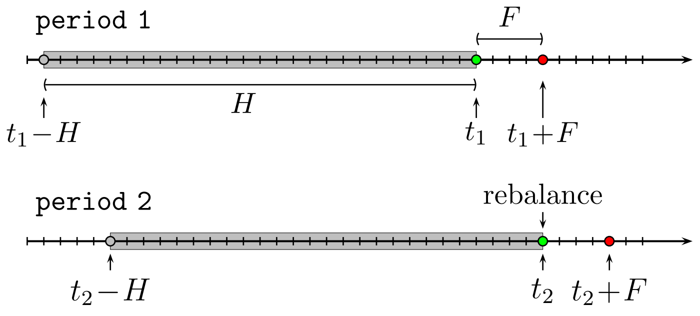


Figure 5 shows the out-of-sample results of such a 'walk-forward' for long-only portfolios minimising Value-at-Risk, Expected Shortfall and the `Omega`-function (the dotted lines indicate the dates when the portfolio was rebalanced).

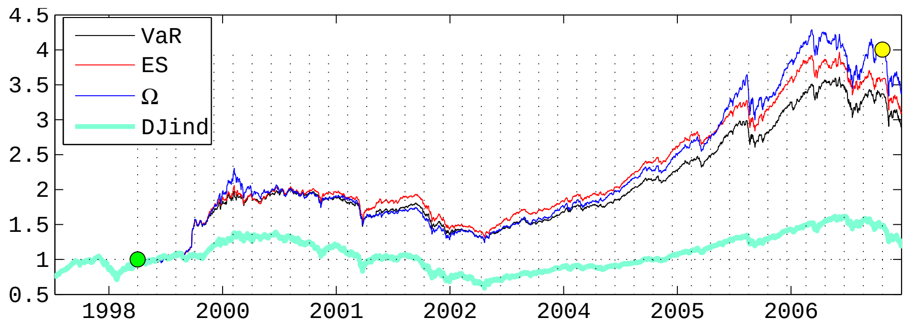


Fig. 5. Out-of-sample results for several selection criteria.

5 Stochastics and convergence of the solution

5.1 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/keep new candidate solutions. Restarting the same algorithm with a different seed value will in general not lead to the same solution. The results of the optimisation can thus be regarded as realisations of a random variable with some (unknown) distribution $\mathcal{D}$ (Gilli and Winker, 2008).[8] 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). Note that 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.

As mentioned before, heuristics are usually 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

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

$\mathcal{D}$ will only depend on the total number of iterations per restart (n_{Steps}); the restarts are draws from $\mathcal{D}$. The time an algorithm needs to produce a solution is equivalent (proportional) to $\mathcal{I}$ in a serial environment, but this does not hold for computations in parallel.

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

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 steps. Intuitively, one may rather use less iterations and thus have a 'worse' distribution of solutions, but still by using 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 again at the portfolio optimisation problem described in the last section; for more details, the reader is referred to Gilli and Schumann (2008).

5.2 Parallelisation

The stochastics of the solution can be exploited to inform how to distribute computations for heuristic optimisation problems. The basic intuition is that one may often trade off steps per restart with the number of restarts. That is, one may choose to allow for less steps in each single optimisation run, but simply generate more draws by restarting more often.

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 Φ chosen here was the `Omega`-function introduced by Keating and Shadwick (2002); the optimisation included several constraints including sector weightings and cardinality.[9] Formally, Φ is given

[9] 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.

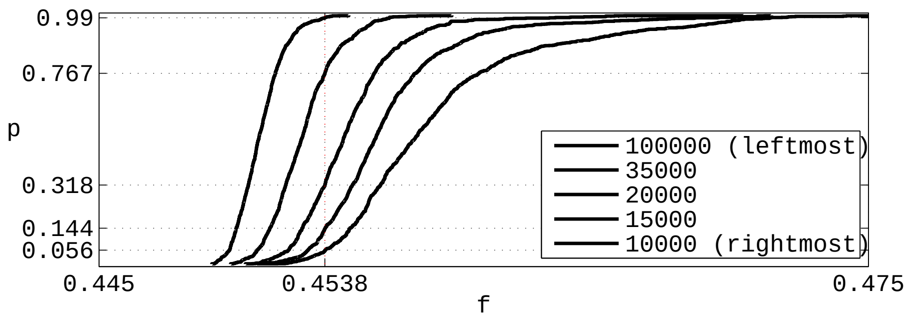


Fig. 6. Distribution of solutions for increasing number of iterations.

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 a large number of draws when the number of steps was fixed at levels between 100 000 and 10 000. It can be seen how the solutions converge towards a suspected global minimum.

Given a desired quality of the solution, one may now explicitly quantify the trade-off between steps 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 Figure 6. For each distribution corresponding to an optimisation run with fewer iterations, one can compute the number of restarts n_{Restarts} necessary to obtain at least one result below f^* with the desired by probability.

Let $C_{n_{\text{Restarts}}}$ be the respective number of steps. Then the potential speedup for n_{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 Figure 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.

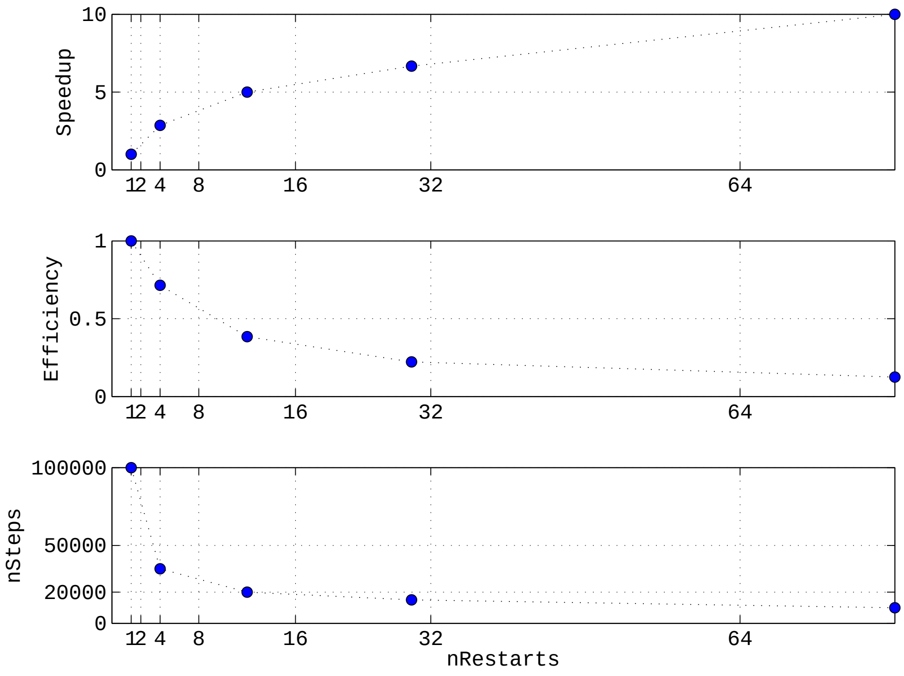


Fig. 7. Speedup (upper), efficiency (middle) and required iterations (lower) as a function of the number of nodes.

6 Conclusion

Many optimisation problems in financial modelling and estimation are difficult to solve as they are non-convex and often exhibit multiple local optima; several examples have been presented in Section 2. 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 (eg, limiting the parameter domain). Even though this allows to employ classical optimisation methods, the results obtained are not necessarily good solutions to the original problem. An alternative possibility is the use of optimisation heuristics like Simulated Annealing or Genetic Algorithms. These methods are powerful enough to handle non-convex optimisation problems; they can flexibly include side constraints without restrictions on the functional function of these constraints.

In their original description, heuristics are often not very precise in how to specify particular components or set parameters (hence the term 'meta-heuristics'). Their implementation often requires some decisions, for instance on how new candidate solutions are created, or how often certain operations (like mutation in Genetic Algorithms) are applied. Section 4 gave a brief case study of such an implementation process for a portfolio selection model. This showed that such specifications often follow quite naturally from the given problem description (though they still often require some experimentation).

An aspect of heuristic techniques that differs from classical methods is the stochastic nature of the obtained solutions. As was already pointed out in the Introduction, this is usually not a concern for practical applications. In fact, analysts who work with empirical data are normally used to uncertainty around parameter estimates (though the source of this uncertainty here is a different one). What is more, this stochasticity can even be exploited to inform how to distribute optimisation problems to parallel machines.

In summary, heuristic optimisation techniques allow to approach many problems in financial modelling in their original form, without need to simplify them. Thus they seem very useful instruments for research as well as applications in practice.

References

Althöfer, Ingo and Klaus-Uwe Koschnick (1991). On the Convergence of Threshold Accepting. *Applied Mathematics and Optimization* **24**, 183–195.

Bakshi, Gurdip, Charles Cao and Zhiwu Chen (1997). Empirical Performance of Altemative Option Pricing Models. *The Journal of Finance* **52**(5), 2003–2049.

Barr, Richard S., Bruce L. Golden, James P. Kelly, Mauricio G. C. Resende and William R. Stewart (1995). Designing and Reporting on Computational Experiments with Heuristic Methods. *Journal of Heuristics* **1**(1), 9–32.

Bates, David S. (2003). Empirical Option Pricing: A Retrospection. *Journal of Econometrics* **116**, 387–404.

Blume, Marshall E. (1971). On the Assessment of Risk. *The Journal of Finance* **26**(1), 1–10.

Brandimarte, Paolo (2006). *Numerical Methods in Finance and Economics – A Matlab-Based Introduction*. Wiley.

Chan, Louis K. C. and Josef Lakonishok (1992). Robust Measurement of Beta Risk. *The Journal of Financial and Quantitative Analysis* **27**(2), 265–282.

Chan, Louis K. C., Jason Karceski and Josef Lakonishok (1999). On Portfolio Optimization: Forecasting Covariances and Choosing the Risk Model. *The Review of Financial Studies* **12**(5), 937–974.

Chekhlov, Alexei, Stanislav Uryasev and Michael Zabarankin (2005). Drawdown measure in portfolio optimization. *International Journal of Theoretical and Applied Finance* **8**(1), 13–58.

Constantinides, G.M. and A.G. Malliaris (1995). Portfolio theory. In: *Finance (Handbooks in Operations Research and Management Science)* (R.A. Jarrow, V. Maksimovic and W.T. Ziemba, Ed.). Vol. 9. pp. 1–30. North-Holland.

Cont, Rama (2001). Empirical properties of asset returns: Stylized facts and statistical issues. *Quantitative Finance* **1**, 223–236.

Cont, Rama and José da Fonseca (2002). Dynamics of implied volatility surfaces. *Quantitative Finance* **2**, 45–60.

Dacorogna, Michel M., Ramazan Gençay, Ulrich A. Müller, Richard B. Olsen and Olivier V. Pictet (2001). *An introduction to high-frequency finance*. Academic Press.

Dembo, Ron S. (1991). Scenario Optimization. *Annals of Operations Research* **30**, 63–80.

Dixit, Avinash K. (1990). *Optimization in Economic Theory*. 2nd ed.. Oxford University Press.

Dueck, Gunter and Tobias Scheuer (1990). Threshold Accepting. A General Purpose Optimization Algorithm Superior to Simulated Annealing. *Journal of Computational Physics* **90**(1), 161–175.

Eberhart, Russell C. and James Kennedy (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, Eugene F. and Kenneth R. French (1993). Common risk factors in the returns on stocks and bonds. *Journal of Financial Economics* **33**, 3–56.

Fama, Eugene F. and Kenneth R. French (2004). The Capital Asset Pricing Model: Theory and Evidence. *The Journal of Economic Perspectives* **18**(3), 25–46.

Gaivoronski, A. and G. Pflug (2005). Value-at-risk in portfolio optimization: properties and computational approach. *Journal of Risk* **7**(2), 1–31.

Genton, Marc G. and Elvezio Ronchetti (2008). Robust Prediction of Beta. In: *Computational Methods in Financial Engineering—Essays in Honour of Manfred Gilli* (Erricos J. Kontoghiorghes, Berc Rustem and Peter Winker, Ed.). Springer.

Gill, Philip E., Walter Murray and Margaret H. Wright (2004). *Practical Optimization*. Elsevier. Amsterdam.

Gilli, Manfred and Enrico Schumann (2008). Distributed Optimisation of a Portfolio's Omega. *Swiss Finance Institute Research Paper No. 08-17*.

Gilli, Manfred and Evis Këllezi (2002*a*). A Global Optimization Heuristic for Portfolio Choice with VaR and Expected Shortfall. In: *Computational Methods in Decision-making, Economics and Finance* (E. J. Kontoghiorghes, B. Rustem and S. Siokos, Ed.). Applied Optimization Series. Kluwer Academic Publishers. pp. 167–183.

Gilli, Manfred and Evis Këllezi (2002*b*). The Threshold Accepting Heuristic for Index Tracking. In: *Financial Engineering, E-Commerce and Supply Chain* (P. Pardalos and V. K. Tsitsiringos, Ed.). Applied Optimization Series. Kluwer Academic Publishers, Boston. pp. 1–18.

Gilli, Manfred and Peter Winker (2003). A global optimization heuristic for estimating agent based models. *Computational Statistics & Data Analysis* **42**, 299–312.

Gilli, Manfred and Peter Winker (2008). A Review of Heuristic Optimization Methods in Econometrics. *Swiss Finance Institute Research Paper No. 08-12*.

Gilli, Manfred, Dietmar Maringer and Peter Winker (2008). Applications of Heuristics in Finance. In: *Handbook on Information Technology in Finance* (Detlef Seese, Christof Weinhardt and Frank Schlottmann, Ed.). Springer.

Gilli, Manfred, Evis Këllezi and Hilda Hysi (2006). A data-driven optimization heuristic for downside risk minimization. *The Journal of Risk* **8**(3), 1–18.

Glover, Fred W. and Manuel Laguna (1997). *Tabu Search*. Kluwer.

Hamida, Sana Ben and Rama Cont (2005). Recovering Volatility from Option Prices by Evolutionary Optimization. *Journal of Computational Finance* **8**(4), 1–2.

Heston, Steven 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.

Hochreiter, Ronald (2008). Evolutionary Stochastic Portfolio Optimization. In: *Natural Computing in Computational Finance* (Anthony Brabazon and Michael ONeill, Ed.). Springer.

Holland, John H. (1992). *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control and Artificial Intelligence*. MIT Press.

Ince, Ozgur S. and R. Burt Porter (2006). Individual Equity Return Data from Thomson Datastream: Handle with Care!. *Journal of Financial Research* **29**(4), 463–479.

Keating, Con and Bill Shadwick (2002). An Introduction to Omega. *AIMA Newsletter*.

Kirkpatrick, S., C. D. Gelatt and M. P. Vecchi (1983). Optimization by simulated annealing. *Science* **220**(4598), 671–680.

Kirman, Alan P. (1992). Whom or what does the representative individual represent?. *The Journal of Economic Perspectives* **6**(2), 117–136.

Kirman, Alan P. (1993). Ants, rationaliy, and recruitment. *The Quarterly Journal of Economics* **108**(1), 137–156.

Knez, Peter J. and Mark J. Ready (1997). On the Robustness of Size and Book-to-Market in Cross-Sectional Regressions. *The Journal of Finance* **52**(4), 1355–1382.

LeBaron, Blake (2000). Agent-based computational finance: Suggested readings and early research. *Journal of Economic Dynamics and Control* **24**, 679–702.

LeBaron, Blake (2006). Agent-based computational finance. In: *Handbook of Computational Economics Volume II* (Leigh Tesfatsion and Kenneth L. Judd, Ed.). pp. 1187–1233. Elsevier.

Lo, Andrew W. (2008). *Hedge Funds—An Analytic Perspective*. Princeton University Press.

Luenberger, David G. (1998). *Investment Science*. Oxford University Press. Oxford.

Madan, Dilip B. (2001). On the modelling of option prices. *Quantitative Finance* **1**, 481.

Manaster, Steven and Gary Koehler (1982). The Calculation of Implied Variances from the Black-Scholes Model: A Note. *The Journal of Finance* **37**(1), 227–230.

Maringer, Dietmar (2004). Finding the relevant risk factors for asset pricing. *Computational Statistics & Data Analysis* **47**, 339–352.

Maringer, Dietmar (2005). *Portfolio Management with Heuristic Optimization*. Springer.

Maringer, Dietmar (2008). Risk Preferences and Loss Aversion in Portfolio Optimization. In: *Computational Methods in Financial Engineering—Essays in Honour of Manfred Gilli* (Erricos J. Kontoghiorghes, Berc Rustem and Peter Winker, Ed.). Springer.

Markowitz, Harry M. (1952). Portfolio selection. *The Journal of Finance* **7**(1), 77–91.

Markowitz, Harry M. (1959). *Portfolio Selection*. Wiley. New York.

Martin, R. Douglas and Timothy T. Simin (2003). Outlier-Resistant Estimates of Beta. *Financial Analysts Journal* **59**(5), 56–69.

Michalewicz, Zbigniew and David B. Fogel (2004). *How to solve it: modern heuristics*. Springer.

Mikhailov, Sergei and Ulrich Nögel (2003). Hestons stochastic volatility model implementation, calibration and some extensions. *Wilmott* pp. 74–79.

Rockafellar, R. Tyrrell and Stanislav Uryasev (2000). Optimization of Conditional Value-at-Risk. *Journal of Risk* **2**(3), 21–41.

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

Schlottmann, Frank and Detlef Seese (2004). Modern Heuristics for Finance Problems: A Survey of Selected Methods and Applications. In: *Handbook of Computational and Numerical Methods in Finance* (Svetlozar T. Rachev, Eds.). Birkhäuser.

Storn, Rainer and Kenneth Price (1997). Differential evolution—a simple and efficient heuristic for global optimization over continuous spaces. *Journal of Global Optimization* **11**, 341–359.

Vasicek, Oldrich A. (1973). A Note on the Cross-Sectional Information in Bayesian Estimation of Security Betas. *The Journal of Finance* **28**(5), 1233–1239.

Winker, Peter (2001). *Optimization Heuristics in Econometrics: Applications of Threshold Accepting*. Wiley.

Winker, Peter and Dietmar Maringer (2007). The threshold accepting optimisation algorithm in economics and statistics. In: *Optimisation, Econometric and Financial Analysis* (Erricos John Kontoghiorghes and Cristian Gatu, Ed.). Vol. 9 of *Advances in Computational Management Science*. Springer. Berlin. pp. 107–125.

Winker, Peter and Kai-Tai Fang (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, Peter and Manfred Gilli (2004). Applications of optimization heuristics to estimation and modelling problems. *Computational Statistics and Data Analysis* **47**, 211–223.

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