Heuristics for Portfolio Selection

25 November 2013

Manfred Gilli
Université de Genève and Swiss Finance Institute
manfred.gilli@unige.ch

Enrico Schumann
AQ Investment AG
es@enricoschumann.net

'Thus computing is, or at least should be, intimately bound up with both the source of the problem and the use that is going to be made of the answers – it is not a step to be taken in isolation from reality.'

Richard W. Hamming, *An Essay on Numerical Methods*

Summary

Portfolio selection is about combining assets such that investors' financial goals and needs are best satisfied. When operators and academics translate this actual problem into optimisation models, they face two restrictions: the models need to be empirically meaningful, and the models need to be soluble. This chapter will focus on the second restriction. Many optimisation models are difficult to solve because they have multiple local optima or are 'badly-behaved' in other ways. But on modern computers such models can still be handled, through so-called heuristics. To motivate the use of heuristic techniques in finance, we present examples from portfolio selection in which standard optimisation methods fail. We then outline the principles by which heuristics work. To make that discussion more concrete, we describe a simple but effective optimisation technique called Threshold Accepting and how it can be used for constructing portfolios. We also summarise the results of an empirical study on hedge-fund replication.

1 Introduction

Before Markowitz, there was nothing.

Such a statement may exaggerate, but not much. Before Markowitz (1952), investments were good or bad solely according to how much profit they promised. Though investors and academic writers had an intuition that investing in several assets was better than putting 'all eggs in the one basket', the overall goal was to select assets with high returns. Assets were not added to portfolios for risk reasons alone.

Portfolio theory changed that. Markowitz showed that adding ever more assets to a portfolio can typically not completely remove return-variation, but that a careful choice of assets can reduce risk. Thus, managing risk became an explicit part of investment management.

Markowitz's model is quite simple: assets are held for a fixed period of time; which assets are chosen depends on two properties of the resulting portfolio, reward and risk. Markowitz equated reward with expected return and risk with return variance. In this chapter, we will look at such Markowitz-type models. There exist more complex models for sure (eg, spanning several time periods), but the one-period setting still provides enough empirical and computational difficulties that it warrants discussion. Thus, we shall stay in the one-period setting; we will, however, deviate from Markowitz's model in our definition of risk and reward.

In fact, already back in the 1950s Markowitz thought that there might be better specifications for risk. Downside semi-variance seemed more appealing than variance (Markowitz, 1959) because it does not penalise upside return-variation. Eventually, Markowitz rejected the idea because at that time it was too difficult to compute optimal portfolios. But with heuristics – the methods that we explain in this chapter – we can solve models without restrictions on the functional form of the selection criterion or the constraints; thus, downside-risk specifications can easily be handled.

However, we will not only discuss computation. After all, this chapter's topic is the selection of financial portfolios, and so we need to explain how this application is special, and how a prescription to optimise portfolios differs from a general discussion of numerical optimisation. We shall argue that the challenges to the practitioner of portfolio construction – we call him 'the analyst' – can be grouped into three topics:

(i) The modelling. The analyst needs to decide what financial goals and necessities exist and should be put into the model. And, of course, he cannot speak of abstract quantities such as 'risk' – everything has to be made precise.

(ii) The forecasting. Portfolio selection is about choosing assets today in the hope they do well tomorrow. Thus, the analyst can only use quantities in his models that he can forecast sufficiently well.

(iii) The computation. Given the model and forecasts, the analyst needs to solve the model. A 'good' model cannot be good if we cannot compute its solution. (The proof is simple: if we never could solve it, we could never test it. So how could we tell it was good in the first place?)

In fact, computational restrictions are much less of an obstacle to working with portfolio models than is sometimes thought. The computational power that the analyst has at his disposal today – literally in his hands, given the developments in tablets and other handheld computers – is such that many models that appear difficult in theory turn out to be very tractable practically.

But with power comes responsibility. In the past, a financial economist could well concentrate on theory, only to leave the lowly computational work to the specialist who, in turn, understood nothing of finance and economics. Today these roles have merged.[1] To be sure, such a separation was always

[1] Such a merging of roles did not only happen in computational finance; it also took place in publishing and data analysis in general.

bad, but at least it could serve as an excuse for ignoring certain aspects of the portfolio selection process. Today, the excuse has disappeared. It is the analyst's job to understand and manage the process of portfolio selection at all stages – modelling, forecasting, computation.

Two principles will guide us throughout this chapter (borrowed from Gilli et al., 2011a):

(i) the application matters, and

(ii) go experiment.

Principle (i) means that what matters is to select well-performing portfolios. Any computational technique that helps us to reach that goal is fine; likewise, any apparent computational difficulty needs to be judged by how much it impedes us reaching our goal.

Principle (ii) is particularly relevant to computation. During the process of portfolio selection, many situations arise in which we are faced with choices: many small decisions that have to be taken while writing down a model, preparing data or setting up an algorithm. We can rarely give general advice on such decisions, but that is not a problem: just go experiment and find out for yourself.

These two principles may also be read as kind of a warning: we will focus on the practical application of portfolio optimisation. We care little whether what we say fits into theoretical paradigms. Neither do we care about the kind of 'practical use' to which many results in academic finance and operations research are put – to end up in financial institutions' marketing departments, or those of software vendors. We describe optimisation methods that allow the analyst to specify models as he likes, but only to test those models and to discard those that do not work – which is, unfortunately, most of them. We will offer our judgement on what we deem important, and what is not. In our opinion; you are welcome to disagree.

The chapter is structured as follows. In the next section, we shall discuss problems and models (throughout this chapter, you will find us strictly distinguishing between those terms). Then, in Section 3, we will describe the principles of heuristics. As we said, these methods allow the analyst to handle models without restrictions on the functional form of the objective function and the constraints. Of course, a price must be paid for that, and so heuristics come with their own difficulties. In particular, almost all heuristics are stochastic algorithms, and hence solutions need to be evaluated with some care. We shall address these difficulties and suggest ways to handle them practically. Two example sections follow: in Section 4, we detail one heuristic, Threshold Accepting, and how it can be applied to portfolio selection. Then, in Section 5, we present a summary of an empirical study that used Threshold Accepting. Section 6 concludes.

2 Of problems, models and methods

We start with a brief discussion of a one-period optimisation model. These descriptions should be familiar to most readers. We will step back then and reflect on this model; we shall zoom in again and discuss how to solve the model only in the next section.

2.1 A one-period investment model

We want to invest a budget v_0 into assets from a universe of $n_\mathtt{A}$ assets. These assets are then held for a period of length T. Let p_0 be a vector of current (known) prices, then the budget constraint can be written as

$$p_0'x \leq v_0\,, \tag{1}$$

in which x is a vector of numbers of contracts we hold. At time T, prices will have evolved to p_T and the portfolio value will be

$$v_T = p_T'x\,. \tag{2}$$

To rank different choices for x, we need an objective function ϕ that maps v_T into a real number. In fact, the function may also look at the path $\{v\}_0^T$ that the portfolio value takes between portfolio inception and T. We remain in the one-period framework, since we do not trade between time 0 and T. Thus, the goal becomes

$$\underset{x}{\text{minimise}}\ \phi\left(\{v\}_0^T\right) \tag{3}$$

subject, at least, to the budget constraint. (In order to maximise, we minimise $-\phi$.) Possible specifications for ϕ could be moments (also partial or conditional moments), quantiles, or in essence any function that can be evaluated for $\{v\}_0^T$. See Gilli et al. (2011a, Chapter 13) for examples and a detailed discussion.

In many cases, we add further constraints. We may have legal or regulatory restrictions on how to invest. For instance, many institutional investors cannot legally hold more than a specific amount of assets from one counter party. Conceptually-simple restrictions can be difficult for standard optimisation methods; see for instance Scozzari et al. (2013).

Empirically, restrictions can provide safeguards against overfitting. There is much evidence that limiting position sizes improves performance. The classic reference is Frost and Savarino (1988); see also Jagannathan and Ma (2003).

Constraints can also help to make estimated parameters interpretable. Variances cannot be negative and probabilities must lie between zero and one. But with unconstrained numerical procedures, there is no guarantee that such restrictions hold, and we may need to impose them to get meaningful results.

2.2 Reality to model, and back

What we described in the last section is a model, not the actual problem. The problem is finding assets that give, loosely speaking, much reward with little risk. The model we described assumed a simple investment process (buy-and-hold) and fixed the notions of reward and risk through our choice of ϕ.

2.2.1 Sources of error

Modelling is the process of putting the actual problem into a form that can be understood by a computer. We have to make vague notions precise, and we often need to simplify and approximate. This, in turn, will introduce errors. The word error must not be understood in the sense that something did not work as expected. Approximation errors originate from the very practice of modelling.

Following a classic discussion in von Neumann and Goldstine (1947) – expanded in Morgenstern (1963) –, we group these errors into two categories: empirical errors (a.k.a. model errors) and numerical errors. The analyst's job is not only to acknowledge such errors, but to actually evaluate them. In this respect, we are fortunate in finance since we can often measure the magnitude of errors in meaningful units, namely euros (or dollars, francs or whatever your favourite currency). Some errors are simply bigger than others and, thus, matter more. True, such interpretation is often difficult and imprecise, but a carefully exploring, quantifying and discussing the effects of model choices etc. is always preferred to dismissing such a discussion as 'out-of-scope'.

Let us start with model errors. A portfolio in Markowitz' model is, in essence, a return distribution, which looks good or bad according to the objective function. How we define this objective function has considerable impact on what portfolio we choose. The word risk for instance is often used almost synonymously with return variance. But there are other ways to define risk. A typical objection against variance is that it penalises upside as well as downside. And indeed, already more than half a century ago, Markowitz thought about using downside semi-variance instead, which corresponds much better to the financial practitioner's notion of risk (Markowitz, 1959). To quantify how relevant these differences in model specification are, we need to empirically compare[2] different models with respect to how they help to solve our actual problem.

There are other factors than the objective function that affect the quality of a model. Transactions costs for instance can be relevant for specific asset classes, and hence a model that includes them may be better than a model that does not. That does not mean that the analyst should try to put every possible detail into the model. Every Unix user knows that `less` is more powerful than `more`, and the same often holds in modelling. In some cases, simple back-of-the-envelope calculations make clear that certain aspects cannot matter. But more often, whether a certain aspect should be modelled or not, is not clear from the start. Principle (ii) tells us what to do in such cases: we run experiments. We should stress that if we decide to ignore certain aspects in the model, it will be because we consider them unimportant, perhaps even harmful, in an economic or empirical sense, or because we cannot reliably implement them (think of mean return forecasts). But it will not be because we could not handle them computationally.

Once a model is established, it needs to be connected to reality. We need to input forecasts and expectations. We can, for instance, only minimise variance if we have a variance–covariance matrix. Again, such inputs may be good or bad, and we have another source of error. The difficulties in forecasting the required variables are well-established, see Brandt (2009) for an overview. And it is not only the forecasting problem: results are often extremely sensitive to seemingly minor setup variations, for instance, the chosen time horizon (LeBaron and Weigend, 1998; Acker and Duck, 2007; Gilli and Schumann, 2010). That makes it difficult to reject bad models.

The focus of this chapter is not the empirics of portfolio selection models, but their numerical solution. Nevertheless, we chose to review these problems to put this part of the portfolio selection process into perspective. From now on, we will assume that the model and its input have been fixed, so our task remains to solve the model, for which we use a computer. There are two sources of error. Round-off error, because we cannot represent every real number within finite memory; and truncation error, because all computations that 'go to the limit' (be it zero or infinity) must stop before the limit is reached.

Round-off error should rarely be a concern in financial optimisation (see also Trefethen, 2008). It can cause trouble, for sure, but its impact, when compared with model error, is many orders of magnitude smaller.

Truncation error is more relevant for our discussion. In principle, we could solve any optimisation model through random-sampling. If we sampled more and more candidate solutions, we should – in principle – come arbitrarily close to the model's solution. But clearly, in most cases that would be an extremely inefficient way to handle a model.

With heuristics, we face a variant of this truncation error. We have not actually detailed so far what heuristics actually are, but it should suffice to say here that they are iterative methods. The truncation error comes in here because heuristics only provide a stochastic approximation to the optimal solution. In other words, if we run a heuristic once, the result can be considered the realisation of random variable. The distribution of this random variable is a function of the computational effort we make. More precisely, more effort (eg, more iterations), better solutions.

Of course, only obtaining a stochastic approximation of a solution is not really satisfactory from the standpoint of optimisation theory. After all, a model's solution is the optimum; theoretically, there are no better or worse solutions, only *the* solution and everything else.

But such approximate solutions are still useful, simply because better *models* can be used, models that would be too difficult to solve with a classical method. And it turns out that in portfolio selection most models are difficult to solve. As

[2] In an empirically sound way, which essentially means careful data analysis and replication. See, for example, Cohen (1994).

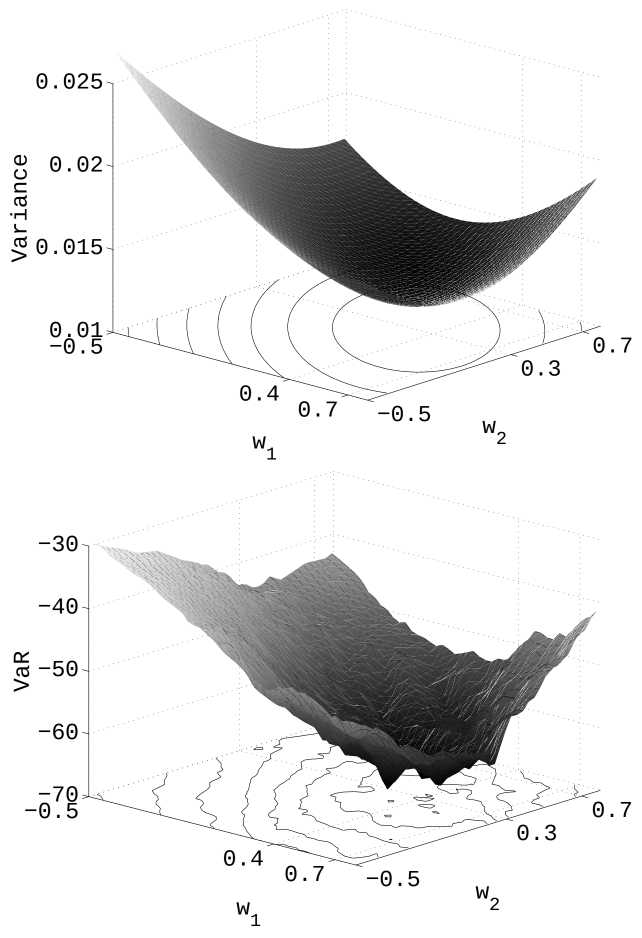


Figure 1: Objective function values for a portfolio selection model with three assets. *x*- and *y*-axis show weights for two assets; the third weight is fixed through the budget constraint. Left panel: objective function is variance. Right panel: objective function is Value-at-Risk.

an example, Figure 1 shows, in its left panel, the variance of a portfolio consisting of three assets. (Actually the squareroot of the variance. The third asset's weight is fixed through the budget constraint.) This is Markowitz's objective function. In the right panel we use the same dataset, but this time the objective function is Value-at-Risk, a quantile of the return distribution. The function for Value-at-Risk is not smooth, and a classic method that uses the gradient may become stuck in a local minimum (if the gradient provides useful guidance at all). Heuristics were specially designed to overcome such local minima, as we will discuss in the next section.

But let us summarise the discussion first. Selecting financial portfolios is much more than running an optimisation algorithm. The modelling process goes from actual problem to a model to the model's solution. (And, finally, we may want to implement the model-solution.)

To see whether we have solved the actual problem – in our case, selecting 'good' portfolios – or at least improved the current status, we follow the chain, but in reverse order. We first check whether our numerical techniques for solving the model work sufficiently well. Then we check if there is a relation between the quality of an in-sample solution and the quality of the out-of-sample solution; see Gilli and Schumann (2011a). This is a very important test of the quality of a model, since it is unreasonable to assume that 'only the minimum' will work well; a small perturbation of the optimal solution should not qualitatively change the results. For example, if we generated a large number of random solutions (see Burns, 2010, for a discussion) and sorted these random solutions by in-sample quality, then we would like to see a positive relation between in-sample and out-of-sample quality. In fact, such tests can also help to evaluate the required precision for an optimisation algorithm. If we cannot empirically establish this relation, there is little point in optimising. Finally, we can compare different models with one another (Gilli and Schumann, 2011b).

3 Heuristics

3.1 What are heuristics?

Different people mean different things when they speak of 'heuristics'. Very loosely, a heuristic is a decision rule (or *modus operandi*) that (i) typically helps to solve a problem or to a make good decision, and that (ii) is simple. This is roughly the definition of Pearl, 1984, and it coincides with a definition that many computer scientists and programmers employ: heuristics as simple rules that provide good answers to problems in typical cases.

In mathematics, a heuristic is a line of reasoning that cannot be formally proved but leads to correct conclusions nonetheless (Pólya, 1957). This idea deserves repetition, because it is relevant for practical optimisation, too: we may not be able to prove that something works, but we can have empirical evidence that it does. Or in other words: not being able to prove that something works does not imply that it does not work.

In psychology a heuristic is a rule-of-thumb, a simple prescription, for decision making. While D. Kahneman and A. Tversky's heuristics-and-biases programme gave the term a rather bad reputation (Tversky and Kahneman, 1974), a more favourable re-interpretation of their results has gained influence more recently; see for example Gigerenzer (2004, 2008).

In fact, there is something fascinating about simplicity when it comes to predicting and operating under uncertainty. Studies in fields such as econometrics, psychology, marketing research, machine learning or forecasting in general document that while simple methods lose out against more sophisticated ones in stylised settings, they yield excellent results in realistic situations (Makridakis et al., 1979, Makridakis and Hibon, 2000, Goldstein and Gigerenzer, 2009, Armstrong, 1978, Dawes, 1979, Dawes, 1994, Lovie and Lovie, 1986). More specifically, simple methods work well in the presence of noise and uncertainty – which is exactly what we have in finance.

But within their respective disciplines, those studies represent no more than niches. The basic idea – providing evidence that simple is sufficiently good – can also be found in the literature on portfolio optimisation in the 1970s; see for instance Elton and Gruber (1973) or Elton et al. (1978). The common thread throughout these papers was the justification of simplified financial models. The problem back then was to reduce computational cost, and the authors tried to empirically justify simpler techniques. Today, complicated models are feasible, but they are still not necessarily better. (Again, Principle (i): the application matters.)

We will define heuristics in a narrower, more-precise sense: as a class of numerical methods for solving optimisation models. The typical model is

$$\underset{x}{\text{minimise}}\, \phi(x, \text{data})$$

in which ϕ is a scalar-valued function and x is a vector of decision variables. (As we already said, by putting a minus in front of ϕ we can make it a maximisation model.) In many cases, we will add constraints to the model.

We find it helpful to not think in terms of a mathematical description, but rather to replace ϕ by something like

```
    solutionQuality = function(x, data).
```

That is, we need to be able to program a mapping from a solution to its quality, given the data. There is no need for a closed-form mathematical description of the function.[3] Indeed, in many applied disciplines there are no closed-form objective functions. The function ϕ could include an experimental setup, with x the chosen treatment and $\phi(x)$ the desirability of its outcome. Or evaluating ϕ might require a complicated stochastic simulation, such as an agent-based model.

A number of requirements describe an optimisation heuristic further (Zanakis and Evans, 1981, Barr et al., 1995, and Winker and Maringer, 2007, list similar criteria):

- The method should give a 'good' stochastic approximation of the true optimum, with 'goodness' measured in computing time or solution quality.
- The method should be robust when we change the model – for instance, when we modify the objective function or add a constraint – and also when we increase the problem size. Results should not vary too much for different parameter settings for 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, but it is a start. Actually, we think that users can only gain intuition about heuristics through studying examples – which we will do in the next section. But for now, we shall go on dwelling on principles.

In a broad sense, we can differentiate between two classes of heuristics, constructive methods and iterative-search methods. In this chapter, we shall concentrate on the latter type, so let us give a quick example for constructive methods and then not mention them any further. For a constructive method, an algorithm starts with an empty solution and adds components step-by-step; the procedure terminates when it has completed one solution. An example: a reasonable low-variance equity portfolio of cardinality N can be constructed by (i) obtaining forecasts for the marginal variances of all eligible assets, (ii) sort the assets by forecast variance and (iii) keep the N assets with the lowest forecast variance in the portfolio (equally-weighted); see Schumann (2013).

For iterative search methods the algorithm moves from solution to solution, that is, a complete existing solution is modified to obtain a new solution. Such a new solution may be quite different from previous ones, as some methods, such as Genetic Algorithms, create new solutions in a rather discontinuous ways. But still, a new solution will share characteristics with its predecessor (if that was not the case, we would be doing random-sampling).

3.2 Principles

The following pseudocode should make the idea of an iterative method more precise.

```
1: generate initial solution $x^c$
2: while stopping condition not met do
3:     create new solution $x^n = N(x^c)$
4:     if $A(\phi, x^n, x^c, \ldots)$ then $x^c = x^n$
5: end while
6: return $x^c$
```

In words: we start with a solution x^c, typically randomly chosen. Then, in each iteration, the function N ('neighbour') makes a copy of x^c and modifies this copy; thus, we get a new candidate solution x^n. The function A ('accept') decides whether x^n replaces x^c, typically by comparing the objective function values of the solutions. The process is repeated until a stopping condition is satisfied; finally, x^c is returned.

To implement such a method, we need to specify

- how we represent a solution x,
- how we evaluate a solution (the function ϕ),
- how we change a solution (the function N),
- how to decide whether to accept a solution (the function A),
- when to stop.

These building blocks would still apply to a classical method. For example, for a gradient-based method x would be a numeric vector; N would evaluate the gradient at x^c and then move minus the gradient with a specified stepsize; A would evaluate x^c and x^n, and replace x^c only if x^n is better; if not, the search is stopped.

Heuristics use other, often simpler, mechanisms. In fact, two characteristics will show up in almost all methods. (i) Heuristics will not insist on the best possible moves. A heuristic

[3] Mathematically a function is nothing but a mapping, so there is no contradiction here. But when people see $\phi(x)$ they intuitively often think of something like $\phi(x) = \sqrt{x} + x^2$. We would prefer they thought of a programme, not a formula.

may accept a new solution x^n even if it is worse than the current solution. (ii) Heuristics typically have random elements. For instance, a heuristic may change x^c randomly (instead of locally-optimally as in a gradient search). These characteristics make heuristics inefficient for well-behaved models. But for difficult models (for instance, such with many local optima as in Figure 1), they enable heuristics to move away from local optima.[4]

Let us give a concrete example, namely the problem we already used earlier: we want to select N assets, equally-weighted, out of a large number of assets, such that the resulting portfolio has a small variance. We assume that we have a forecast for the variance–covariance matrix available. Then a simple method for getting a very good solution to this model is a local search. For a local search,

- the solution x is a list of the included assets;
- the objective function ϕ is a function that computes the variance forecast for a portfolio x;
- the function N picks one neighbour by randomly removing one asset from the portfolio and adding another one;
- the function A compares $\phi(x^c)$ and $\phi(x^n)$, and if x^n is not worse, accepts it;
- the stopping rule is to quit after a fixed number of iterations.

Note that local search is still greedy in a sense, since it will not accept a new solution that is worse than the previous one. Thus, if the search arrives at a solution that is better than all its neighbours, it can never move away from it – even if this solution is only a local optimum. Heuristic methods that build on local search thus employ additional strategies for escaping such local optima.

And indeed, with a small – but important – variation we arrive at Simulated Annealing (Kirkpatrick et al., 1983). We use a different acceptance rule A: If the new solution is better, accept it. If it is worse, do still accept it, but only with a specific probability. This probability in turn depends on the new solution's quality: the worse it is, the less likely it is the solution is accepted. Also, the probability of acceptence is typically lower in later iterations (that is, the algorithm becomes pickier). In many implementations, the probability at later stages is essentially zero; thus, Simulated Annealing turns into a local search.

3.3 Constraints

Nothing in the pseudocode that we showed above ensures that a constraint on a solution x is observed. But it is often constraints that make models realistic and difficult. Several strategies exist for including restrictions into heuristics.

3.3.1 Throw away

If our model has only few constraints that are not often hit, the simplest approach is to 'throw away' infeasible new solutions. That is, if a neighbour solution violates a constraint, we just select another neighbour. Note that this means that we include the contraints in the acceptance function A.

3.3.2 Include constraint in N

We can directly use the constraint to create new, feasible solutions. In portfolio selection models we usually have a budget constraint; that is, we require that all asset weights sum to one. This constraint can be enforced when we compute new solutions by increasing some weights and decreasing others such that the sum of all weight changes is zero.

3.3.3 Transform x

An older but still used idea is to transform variables. This approach sometimes works for constraints that require that the elements of x lie in certain ranges; see the discussion in Powell (1972). For instance, $\sin(x)$ will map any real x to the range $[-1,1]$; $\alpha\,(\sin(x))^2$ will give a mapping to $[0,\alpha]$. But such transformations come with their own problems; see Gill et al. (1986, Section 7.4); in particular, it may become difficult to change a problem later on or to handle multiple constraints.

3.3.4 Repair x

We can introduce mechanisms to correct solutions that violate constraints. For example, if a solution x holds the portfolio weights, then dividing every element in x by the sum of the elements of x ensures that all weights sum to unity.

3.3.5 Penalise x

Finally, we can penalise infeasible solutions. Whenever a constraint is violated, we add a penalty term to the objective function and so downgrade the quality of the solution. In essence, we change the problem to an unconstrained one for which we can use the heuristic. The penalty is often made an increasing function of the magnitude of violation. Thus, the algorithm may move through infeasible areas of the search space, but will have guidance to return to feasible areas. The penalty approach is the most generic strategy to include constraints; it is convenient since the computational architecture needs hardly be changed. Penalties create soft constraints since the algorithm could in principle always override a penalty; practically, we can set the penalty so high that we have hard constraints.

[4] In principle, because of such mechanisms a heuristic could drift farther and farther off a good solution. But practically, that is very unlikely because every heuristic has a bias towards good solutions. In Threshold Accepting, the method that we describe in Section 4, that bias comes into effect because a better solution is always accepted, a worse one only if it is not too bad. Since we repeat this creating of new candidate solutions thousands of times, we can be very certain that the scenario of drifting-off a good solution does practically not occur.

3.4 Random solutions

The most common objection against using heuristics is the fact that, since heuristics explicitly rely on random mechanisms, their solutions are also random. This randomness, it is argued, makes it difficult to evaluate the quality of solutions computed by such algorithms. (The discussion in this section builds on Gilli et al., 2011a.)

3.4.1 Randomness

A naïve approach to solving an optimisation model could be this: randomly generate a large number of candidate solutions, evaluate all solutions and pick the best one. This best solution is our overall solution.

If we repeated the whole procedure a second time, our overall solution would probably be a different one. Thus, the solution x we obtain through our sampling strategy is stochastic. The difference between our solution and the actual optimum would be a kind of truncation error, since if we sampled more and more, we should in theory come arbitrarily close to the optimum. Importantly, the variability of the solution stems from our numerical technique; it has nothing to do with the error terms that we may have in models to account for uncertainty. Stochastic solutions may even occur with non-stochastic methods: think of search spaces like those we showed in Figure 1. Even if we used a deterministic method like a gradient search, the many local minima would make sure that repeated runs from different starting points result in different solutions.

We can treat the result of a stochastic algorithm as a random variable with some distribution D. What exactly the 'result' of a restart is depends on our setting. We will want to look at the objective function value (ie, the solution quality), but we may also look at the decision variables given by a solution, that is, the portfolio weights. In any case, we collect all the quantities of interest in a vector ϱ. The result ϱ_j of a restart j is a random draw from D.

The trouble is that we do not know what D looks like. But fortunately, there is a simple way to find out for a given model. We run a reasonably large number of restarts, each time store ϱ_j, and finally compute the empirical distribution function of the $\varrho_j, j = 1, \ldots,$ number-of-restarts as an estimate for D. For a given model or model class, the shape of the distribution D will depend on the chosen method. Some techniques will be more appropriate than others and give less variable and on average better results. And D will often depend on the particular settings of the method, in particular the number of iterations – the search time – that we allow for.

Unlike classical optimization techniques, heuristics can walk away from local minima; they will not necessarily get trapped. So if we let the algorithm search for longer, we can hope to find better solutions. For minimization problems, when we increase the number of iterations, the mass of D will move to the left and the distribution will become less variable. Ideally, when we let the computing time grow ever longer, D should degenerate into a single point, the global minimum. There exist proofs of this convergence to the global minimum for many heuristic methods (see Gelfand and Mitter, 1985, for Simulated Annealing; Rudolph, 1994, for Genetic Algorithms; Gutjahr, 2000, Stützle and Dorigo, 2002, for Ant Colony Optimisation; Bergh and Engelbrecht, 2006, for Particle Swarm Optimisation).

Unfortunately, these proofs are not much help for practical applications. They often rely on asymptotic arguments; and many such proofs are nonconstructive (eg, Althöfer and Koschnick, 1991, for Threshold Accepting): they demonstrate that parameter settings exist that lead (asymptotically) to the global optimum. Yet, practically, there is no way of telling whether the chosen parameter setting is correct in this sense; we are never guaranteed that D really degenerates to the global optimum as the number of iterations grows.

Fortunately, we do not need these proofs to make meaningful statements about the performance of specific methods. For a given model class, we can run experiments. Such experiments also help investigate the sensitivity of the solutions with respect to different parameter settings for the heuristic. Experimental results are of course no proof of the general appropriateness of a method, but they are evidence of how a method performs for a given class of models; often this is all that is needed for practical applications.

4 An example: Threshold Accepting

In this section, we will discuss one heuristic, Threshold Accepting, in more detail. The algorithm is a simplified variant of Simulated Annealing and was first proposed by Dueck and Scheuer (1990) and Moscato and Fontanari (1990). As far as we know, it was also the first optimisation heuristic used for portfolio selection (Dueck and Winker, 1992). For an overview of the application of other heuristics, such as Ant Systems or Simulated Annealing, to portfolio selection models, we recommend Maringer (2005).

4.1 The algorithm

Threshold Accepting (TA) builds on local search. A local search starts with a random feasible solution x^c (in our case, a random portfolio), which we call the 'current solution' since it represents the best we have so far. Then, a new solution x^n close to x^c is randomly chosen. We will discuss 'close to x^c' shortly. We call x^n a neighbour to the current solution, or simply a neighbour. If x^n is better than x^c, then x^n replaces x^c (ie, the neighbour becomes the current solution); if not, another neighbour is selected. Algorithm 1 gives this procedure in pseudocode. The stopping criterion here is a number of iterations n_{Steps}, fixed in advance by the analyst.

All that the method requires is that the objective function can be evaluated for a given portfolio x; there is no need for the objective function to be continuous or differentiable. Of course, for problems with many local minima, a local search will stop at the first local optimum it finds.

Algorithm 1 Local Search.

```
1: set $n_{\text{Steps}}$
2: randomly generate current solution $x^c$
3: for $i = 1 : n_{\text{Steps}}$ do
4:     generate $x^n = N(x^c)$ and compute $\Delta = \phi(x^n) - \phi(x^c)$
5:     if $\Delta < 0$ then $x^c = x^n$
6: end for
7: return $x^c$
```

TA adds a simple escape strategy for local minima: it will not only accept new solutions that are better, but also allow uphill moves, as long as the deterioration of ϕ does not exceed a fixed threshold (thus its name). Over time, that threshold decreases to zero; so eventually TA turns into a local search. The whole procedure is summarised in Algorithm 2.

Algorithm 2 Threshold Accepting.

```
1: set $n_{\text{Steps}}$ and $n_{\text{Rounds}}$
2: compute threshold sequence $\tau$
3: randomly generate current solution $x^c$
4: for $k = 1 : n_{\text{Rounds}}$ do
5:     for $i = 1 : n_{\text{Steps}}$ do
6:         generate $x^n = N(x^c)$ and compute $\Delta = \phi(x^n) - \phi(x^c)$
7:         if $\Delta < \tau_k$ then $x^c = x^n$
8:     end for
9: end for
10: return $x^c$
```

Compared with local search, the changes are actually small. We add an outer loop that controls the thresholds τ. In Statement 7, the acceptance criterion is changed from '$\Delta < 0$' to '$\Delta < \tau_k$', ie, from 'if improvement' to 'if not worse than τ_k'. For an actual implementation, we need a way to represent a solution, an objective function ϕ, the neighbourhood function N, the thresholds τ and the stopping criterion (see Section 3.2 above). The first and the last choice are quickly made: a solution is a numeric vector of positions (or perhaps weights); the stopping criterion is simply a fixed number of iterations.[5]

4.2 Implementation

4.2.1 The objective function

In Markowitz's mean–variance model, we need forecasts of the means, variances and correlations of the assets. With these inputs, we can easily compute forecasts of mean and variance for any specific portfolio.

Unfortunately, this approach rarely generalises to other specifications of risk and reward, or only in ways that are computationally very costly (Jondeau et al., 2007, ch. 9). For instance, even if we had forecasts of the lower partial moments of all the assets in the portfolio, we could not aggregate these to the lower partial moment of the whole portfolio (Grootveld and Hallerbach, 1999).

But we do actually not require such an aggregation. Instead, we will do scenario optimisation (Dembo, 1991). The simplest case is to regard every historical return observation as one scenario. However, using actual forecasts (eg, creating 'artifical' scenarios through resampling) can help to improve the out-of-sample performance of portfolios (Gilli and Schumann, 2009; Gilli et al., 2011b).

Suppose we have n_A assets and n_S return scenarios, all collected in a matrix R of size $n_S \times n_A$. We can equivalently work with price scenarios, computed as

$$P = (1 + R) \times \text{diag}(p_0)$$

in which 1 is a matrix of ones of size $n_S \times n_A$, and 'diag' is an operator that transforms a vector into a diagonal matrix. Note that the columns of P (or R) are not time series. Every row of P holds the prices for one possible future scenario that might occur, given initial prices p_0. In fact, for many objective functions (such as partial moments), it is not relevant whether the scenarios are sorted in time, since such criteria only capture the cross-section of returns. The portfolio values v in these scenarios are given by the product Px.

But there are selection criteria that need time series, for instance drawdown. Resampling is still possible to create path scenarios: we may, for instance, use models that capture serial dependencies and then resample from their residuals, or use a block bootstrap method. Assume we have only a single scenario of paths of the assets, and arrange the prices in a matrix P^{ts} of size $(T + 1) \times n_A$, where each column holds the prices of one asset. The portfolio values over time are then given by $v = P^{\text{ts}}x$. Note that Px gives a sample of portfolio values over the cross-section of scenarios, while $P^{\text{ts}}x$ gives one path of the portfolio value from time 0 to time T. For both scenarios and paths, given a vector v it is easy to evaluate an objective function.

Working with scenarios in this way is not restrictive: if we preferred a parametric model, we could always calibrate it to the scenarios (eg, compute a variance–covariance matrix from the scenarios). The approach offers another advantage, namely that we can make the time required to evaluate the objective function independent of the number of assets. Assume we work with the matrix P of price scenarios (the same holds for P^{ts}). This matrix is often fairly large, with thousands of scenarios for hundreds or thousands of assets. TA started with an initial portfolio x^c and now has to evaluate x^n. Hence 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

$$x^n = x^c + x^\Delta$$

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

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

Many elements of x^Δ are zero, and hence only a few columns of P are necessary for the matrix multiplication. So we create

[5]The number of iterations depends on the problem. Here, again, Principle (ii) tells us how to proceed: small-scale experiments will quickly provide us with a reasonable idea of how many iterations are needed. See Gilli et al. (2011a); in particular Chapters 11 and 12.

a matrix P_* that only holds the columns where x^Δ is nonzero and a vector x_*^Δ that contains only the nonzero elements of x^Δ; then we replace Px^Δ by $P_* x_*^\Delta$.

4.2.2 The neighbourhood function

To move from one solution to the next, we need to define a neighbourhood function N that creates new candidate solutions. For portfolio selection problems, we have a natural way to create neighbour solutions: pick one asset in the portfolio randomly, '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 small fixed fraction such as 0.1%. Experiments suggest that, for practical purposes, both methods give similar results; a fixed fraction may even be preferred.

4.2.3 The threshold sequence

The threshold sequence is an ordered vector of positive numbers that decrease to zero or at least become very small. The neighbourhood definition and the thresholds are tightly connected. Larger neighbourhoods, with larger changes from one candidate portfolio to the next, need generally be accompanied by larger initial threshold values, et vice versa.

Winker and Fang (1997)[6] suggest a data-driven method to obtain the thresholds: generate a large number of random solutions; select one neighbour of each solution according to the given neighbourhood definition; compute the difference between the objective function values for each pair. The thresholds are then a number of decreasing quantiles of these differences; see Algorithm 3.

Algorithm 3 Computing the threshold sequence – Variant 1.

```
1: set $n_{\text{Rounds}}$ (# of thresholds), $n_{\text{Deltas}}$ (# of random solutions)
2: for $i = 1$ to $n_{\text{Deltas}}$ do
3:     randomly generate current solution $x^{\text{c}}$
4:     generate $x^{\text{n}} = N(x^{\text{c}})$
5:     compute $\Delta_i = |\phi(x^{\text{n}}) - \phi(x^{\text{c}})|$
6: end for
7: sort $\Delta_1 \leqslant \Delta_2 \leqslant \cdots \leqslant \Delta_{n_{\text{Deltas}}}$
8: set $\tau = \Delta_{n_{\text{Rounds}}}, \ldots, \Delta_1$
```

The number of thresholds n_{Rounds} with this approach is typically large, hence few steps n_{Steps} per threshold suffice in the inner loop of Algorithm 2; often it is only one step per threshold.

Gilli et al. (2006) suggest to take a random walk through the data instead, recording the changes in the objective function value at every iteration. The thresholds are then a number of decreasing quantiles of these changes. See Algorithm 4.

[6]Similar techniques are used to obtain settings for Simulated Annealing; see for instance Johnson et al. (1989).

Algorithm 4 Computing the threshold sequence – Variant 2.

```
1: set $n_{\text{Rounds}}$ (# of thresholds), $n_{\text{Deltas}}$ (# of random steps)
2: randomly generate current solution $x^{\text{c}}$
3: for $i = 1 : n_{\text{Deltas}}$ do
4:     generate $x^{\text{c}} = N(x^{\text{c}})$ and compute $\Delta_i = |\phi(x^{\text{n}}) - \phi(x^{\text{c}})|$
5:     $x^{\text{c}} = x^{\text{n}}$
6: end for
7: compute empirical distribution (CDF) of $\Delta_i, i = 1, \ldots, n_{\text{Deltas}}$
8: compute threshold sequence $\tau_k = \text{CDF}^{-1}\left(\frac{n_{\text{Rounds}} - k}{n_{\text{Rounds}}}\right), k = 1, \ldots, n_{\text{Rounds}}$
```

4.2.4 Constraints

Several generic approaches for handling constraints were discussed in Section 3.3 above; we will typically use a mixture of those. The budget constraint for example is automatically enforced by the specification of the neighbourhood. Gilli et al. (2011a) provide complete implementations for various other constraints.

5 Portfolio selection with TA

To give an example[7] about the modelling approach, guided by the two principles mentioned in the introduction, ie, (i) the application matters, and (ii) go experiment, we construct a portfolio with the objective to 'at least' replicate a given hedge fund index. In the following we concentrate only on the modelling process.

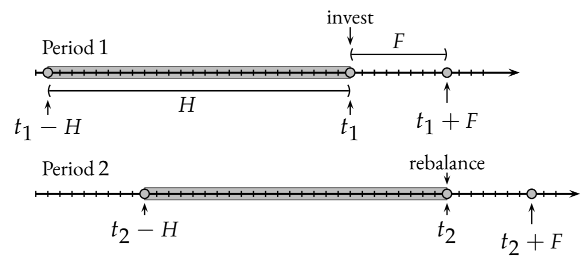
Data, backtesting scheme and reporting of results

The index to replicate is the Credit Suisse/Tremont Hedge Fund Index (CST) available at `www.hedgeindex.com`. According to the information onsite this index is asset-weighted, includes more than 5000 funds with a minimum of US$ 50 million under management. The observations are monthly and cover the period from January 1999 to October 2009.

The instruments used for the replication comprise equity, commodity and bond indices. In the set of equity indices we have about 54 series including broad market, blue chips, sector as well as size and style indices. There are 12 commodity indices and 12 bond indices from government, corporate and emerging markets. The set of data has been collected from Bloomberg.

To analyze the performance of the suggested portfolios we backtest the strategies over ten years with rebalancing where we account for 10 basis points of transaction costs. The rolling window has a historical length of H and a holding period of F. In this application H is one year and F is three months. This leads to trajectories of portfolios values where the portfolios have been rebalanced forty times. The scheme below summarises the technique.

[7]The example builds on Gilli et al. (2010).



To gain insight into the stochastics of the simulated portfolios we jackknife from the historical observations so as to compute a set of results[8] for which we then consider empirical distributions for different features of the portfolio.

For each objective function we report the characteristics of the backtested portfolio in 3 figures and one table, ie,: i) plot of median path of the portfolio value[9], ii) plot of the kernel estimation of the density of the empirical distribution of the mean yearly return, iii) plot of the empirical distribution of the correlation of the portfolio with the market and iv) a table showing the tracking error *TE* defined as the standard deviation of the excess return r_E, the Sharpe ratio S, the annualised return and volatility $\bar{r}$, *vol* and the correlation ρ_{r_P,r_M} with the market.

5.1 'Genesis' of a model

Instead of starting with a predefined idea about what the 'best' model should be we chose to begin with the simplest ideas and subsequently add (or remove) elements in the objective function ϕ according to whether or not they have a desirable impact on the results obtained by backtesting.

The general optimisation problem we have to solve for the different objective functions ϕ is then

$$\min_x \ \phi(x) \tag{4}$$

$$\sum_{j\in\mathcal{J}} x_j \, p_{0j} = v_0 \tag{4'}$$

$$x_j^{\text{inf}} \le x_j \le x_j^{\text{sup}} \qquad j \in \mathcal{J} \tag{4''}$$

$$K_{\text{inf}} \le \#\{\mathcal{J}\} \le K_{\text{sup}} \tag{4'''}$$

$$\vdots$$

where x is a vector with x_j being the quantity of asset j in the portfolio. The optimisation is subject to a set of constraints with in particular the budget constraint (4′) with v_0 the investable wealth and p_{0j} the price of asset j at the beginning of the investment period. Constraint (4″) specifies minimum and maximum holding size for the set of asset ($\mathcal{J}$) in the portfolio. Next we have the cardinality constraint (4‴) which allows for tractability of the resulting portfolios. Further constraints might be included such as total transaction cost, sector constraints and other liquidity issues. For all our portfolios the minimum holding and maximum holding for an asset is 1% respectively 20% and maximum cardinality is limited to $K_{\text{sup}} = 10$. Solutions will be computed with the Threshold Accepting (TA) heuristic described previously in Section 4.

Table 1: Results for median path of simulated portfolios for objective function (5) (Tracking error *TE*, Sharpe ratio *S*, annualised return and volatility $\bar{r}$, *vol*, correlation with market ρ_{r_P,r_M}).

	TE	*S*	$\bar{r}$	*vol*	ρ_{r_P,r_M}
$\lambda = 0.75$	2%	0.62	6%	10%	0.55
$\lambda = 1$	2%	0.45	4%	9%	0.62
CST	–	1.04	7%	7%	0.54

Step 1: Optimisation of tracking error and excess return

A first and straightforward idea is to construct a tracking portfolio, ie, a portfolio that minimises the distance between historical portfolio returns and the index returns. We denote r_P the historical return vector of the tracking portfolio, r_I the index returns and $r_E = r_P - r_I$ the excess return of the portfolio. In order not to penalise upside deviations for a portfolio we also consider the mean excess return $\overline{r_E} = \frac{1}{n}\sum(r_P - r_I)$ leading to the following objective function

$$\lambda \, \|r_E\|^2 - (1-\lambda)\,\overline{r_E} \tag{5}$$

where $\lambda \in [0, 1]$ defines a linear combination between tracking error and excess return.[10]

We computed results for the objective (5) for varying values of the parameter λ controlling the weighting between tracking error and reward. Figure 2 shows the median paths for 100 simulations for $\lambda = 1$ (dark line) which corresponds to minimizing only tracking error. We can trade tracking error against final wealth by decreasing λ. A good compromise is obtained for $\lambda = 0.75$ yielding portfolios close to the index whereas higher weights for the reward (lower values for λ) result in higher final wealth but also higher volatility.

Figure 3 illustrates another feature of the simulated portfolios. It shows the plot of the kernel estimation of the density of the empirical distribution of the mean yearly return of the tracking portfolio (left panel). The vertical line indicates the mean yearly return of the index. The right panel in Figure 3 shows the empirical distribution of the correlation of the optimised portfolios with the market. The dotted line corresponds to the correlation of the index with the market. For the tracking portfolios we observe higher values. Table 1 summarises the performance for the portfolios obtained with objective function (5).

The portfolio for $\lambda = 0.75$ has an average return increased by 50% compared with $\lambda = 1$ in exchange of insignificant loss in tracking performance and small increase of volatility. Furthermore the correlation of the portfolio with the market decreases.

[8]In this case we computed 100 trajectories for each specification of the objective function.

[9]The median path is defined with respect to the final wealth of the portfolios generated with the jackknifing.

[10]Such an approach has been explored in Gilli and Këllezi (2002) using artificial data.

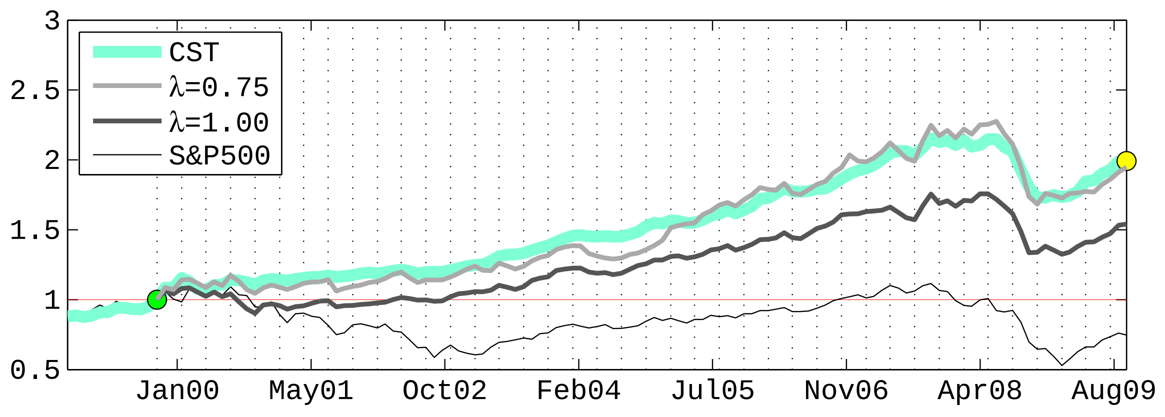


Figure 2: Median paths for portfolios minimizing objective function (5) for $\lambda = 0.75$ and $\lambda = 1$. For reference the performance of the CS Tremont (CST) and the S&P 500 is also shown. Dotted vertical lines indicate rebalancing dates.

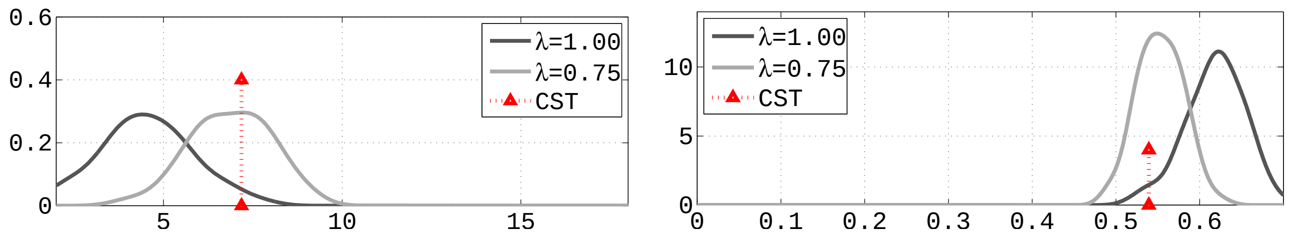


Figure 3: Densities of mean yearly return $\bar{r}$ (left panel) and correlation ρ_{r_P,r_M} (right panel).

Table 2: Results for the median path of the simulated portfolios for objective function (6).

	TE	*S*	$\bar{r}$	*vol*	ρ_{r_P,r_M}
$\lambda = 0.70$	3%	0.63	7%	10%	0.09
$\lambda = 1$	2%	0.60	4%	7%	0.35
CST	–	1.04	7%	7%	0.54

Step 2: Optimisation of tracking error, excess return and ρ_{r_P,r_M}

The goal is to construct a portfolio following the index as close as possible but being little sensitive to adverse market movements. This suggests to include the correlation between tracking portfolio and market into the objective function

$$\lambda \, \|r_E\|^2 - (1-\lambda)\,\overline{r_E} + \rho_{r_P,r_M} \tag{6}$$

where ρ_{r_P,r_M} denotes this correlation computed from the historical data. Minimizing the objective function minimises this correlation. The results for the portfolios minimizing the objective function (6) are given in Figure 4. For $\lambda = 1$, where no excess return enters the optimisation, we observe a particularly smooth median path almost not affected by the drop in the S&P 500 at the end of 2008.

Results in the right panel of Figure 5 are remarkable when compared with the ones in Figure 3 as the distributions indicate that, while maintaining the same level of returns, correlation is now well controlled.

Step 3: Optimisation of tracking error, excess return, ρ_{r_P,r_M} and ρ_{r_P,r_I}

Model (5) focusing on the tracking error, already leads to a portfolio highly correlated with the index. Nevertheless one can think to control this correlation more specifically by introducing it into the objective function. Denoting ρ_{r_P,r_I} this correlation between portfolio and the index, it can be maximised with the new objective function

$$\lambda \, \|r_E\|^2 - (1-\lambda)\,\overline{r_E} + \rho_{r_P,r_M} - \rho_{r_P,r_I} \,. \tag{7}$$

Notice that the effect of λ is not comparable between the different objective functions due to the impact of the additional terms. The results indicate no significant change in overall performance, we rather observe a shift to improved Sharpe ratios for all values of λ. Again lower values for λ, ie, higher weighting for excess return leads to portfolios with higher final wealth but of course at the cost of increased volatility.

As visible in Figure 7, from the point of view of returns $\lambda = 0.6$ produces attractive portfolios which moreover show quite low correlation with the market and high Sharpe ratios.

Step 4: Optimisation of tracking error, excess return, ρ_{r_P,r_M}, ρ_{r_P,r_I} and $\mathcal{D}_{\max}$

A further desirable feature of a portfolio would be to have low drawdown. For a series of portfolio values v_t, $t = 0,1,2\ldots T$

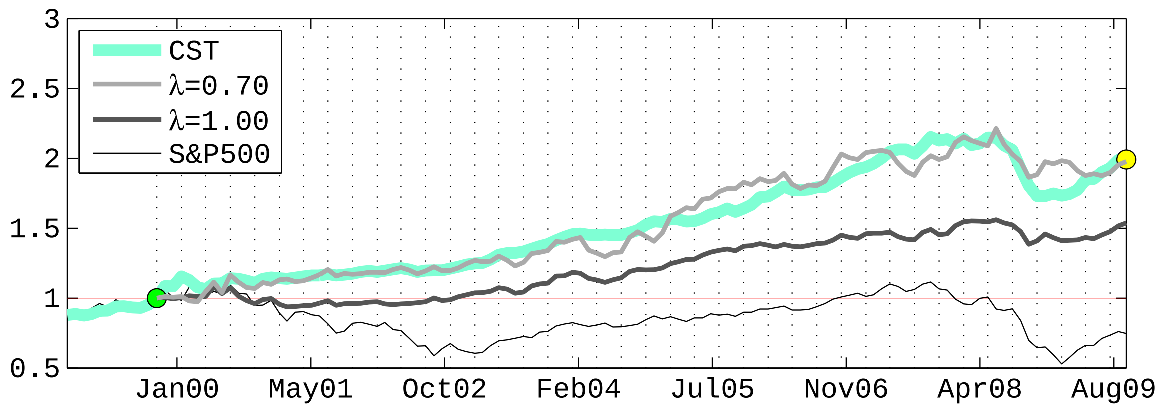


Figure 4: Median paths for portfolios minimizing the objective function (6) controlling correlation ρ_{r_P, r_M} with the market.

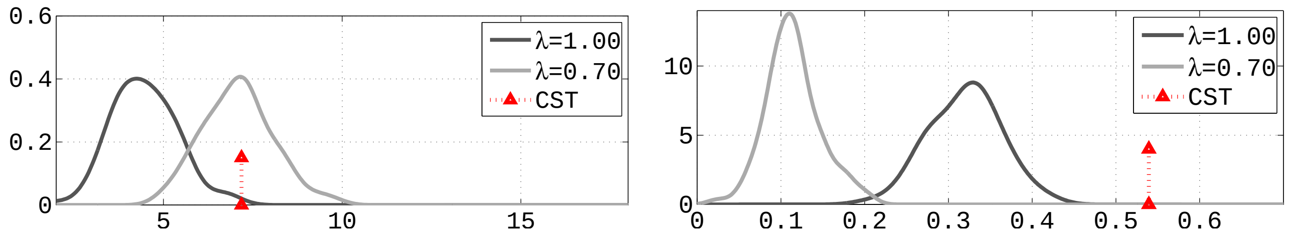


Figure 5: Density of mean yearly return $\bar{r}$ (left panel) and correlation ρ_{r_P, r_M} (right panel) for the objective function (6) controlling correlation with the market.

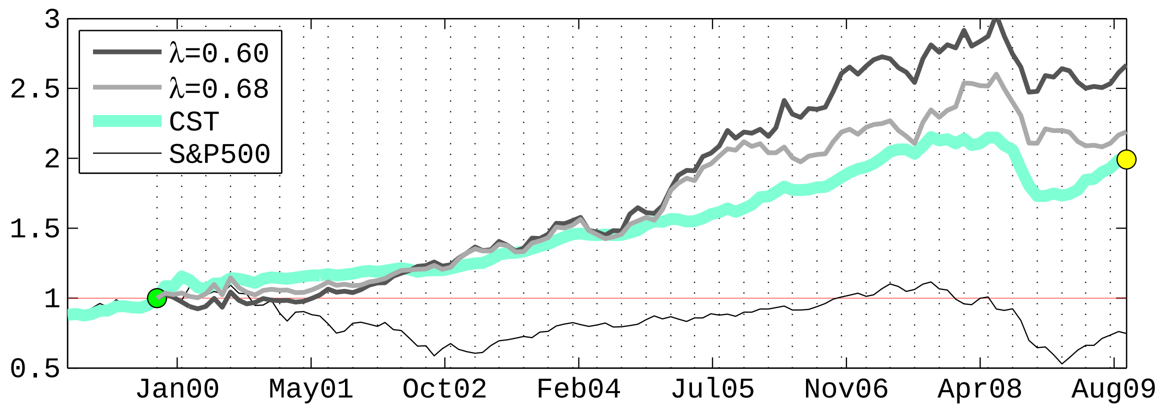


Figure 6: Median paths for portfolios minimizing the objective function (7) controlling correlation ρ_{r_P, r_M} with the market and ρ_{r_P, r_I} with the CST.

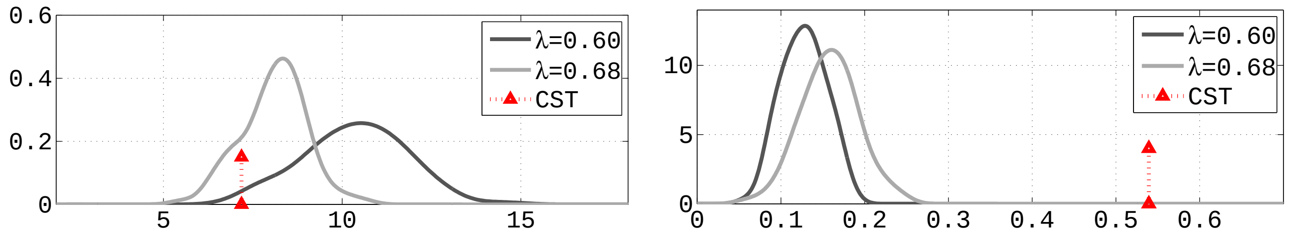


Figure 7: Density of mean yearly return $\bar{r}$ (left panel) and correlation ρ_{r_P, r_M} (right panel) for the objective function (7) controlling correlations with the market and the CST.

Table 3: Results for the median path of the simulated portfolios for objective function (7).

	TE	*S*	$\bar{r}$	*vol*	ρ_{r_P,r_M}
$\lambda = 0.60$	3%	0.88	10%	11%	0.09
$\lambda = 0.68$	3%	0.74	8%	10%	0.17
CST	–	1.04	7%	7%	0.54

Table 4: Results for the median path of the simulated portfolios for objective function (8).

	TE	*S*	$\bar{r}$	*vol*	ρ_{r_P,r_M}
$\lambda = 0$	4%	0.94	16%	17%	0.23
$\lambda = 0.7$	2%	0.97	9%	9%	0.17
$DD_{\max}$	2%	1.01	9%	9%	0.37
CST	–	1.04	7%	7%	0.54

the drawdown is defined as

$$\mathcal{D}_t = v_t^{\max} - v_t$$

where $v_t^{\max}$ is the running maximum, ie, $v_t^{\max} = \max\{v_s | s \in [0, t]\}$. Following this definition $\mathcal{D}$ is a vector for which we can compute the mean, standard deviation or the maximum element $\mathcal{D}_{\max} = \max(\mathcal{D})$ which is the one we use in our next objective function. In other words $\mathcal{D}_{\max}$ measures the largest drop of the portfolio value over the time horizon.

In a first step we consider only the minimisation of the maximum drawdown. In a second step we combine maximum drawdown minimisation with the objective function defined in (7) yielding

$$\lambda \, \|r_E\|^2 - (1-\lambda)\,\overline{r_E} + \rho_{r_P,r_M} - \rho_{r_P,r_I} + \mathcal{D}_{\max}\,. \qquad (8)$$

Figure 8 below reports results for pure maximum drawdown minimisation as well as results for objective function (8) with different weighting of tracking error and excess return. As previously, higher weights for excess return (low λ values) produce high final wealth portfolios.

Looking at the final wealth distribution given in Figure 9 we see that for $\lambda = 0$ we get an impressive shift to the right of the distribution.

In the light of these results portfolios obtained from this last model offer properties suitable to substitute the Credit Suisse/Tremont hedge fund index. In particular for $\lambda = 0.7$ we have comparable Sharpe ratio but higher average return and lower correlation with the market. However the portfolio can be modulated choosing different values for λ in order to meet different preferences or risk aversion of an investor.

It might be interesting to show how the different asset classes are represented in the median portfolios. This is plotted in Figure 10 and we notice how the model reacts to market conditions. For instance, in periods of distress, the weight of fixed income instruments gains importance at the expense of equities. Also, the portfolio where excess return is favored ($\lambda = 0$) has relatively higher weighted commodities and equities.

6 Conclusion

John Tukey once said that an analyst of data needs both tools and understanding. In this chapter, we have given a brief, selective introduction to heuristics in portfolio selection. Heuristics are tools. Powerful tools; but even powerful tools cannot replace understanding.

As we said in the Introduction, the tasks that the analyst faces can be classified into three broad topics: modelling, forecasting and optimisation. In our view, optimisation is the least problematic of these tasks. Of course, that has not always been the case, but on today's computers with today's software, we can handle models and amounts of data that people could not imagine twenty years ago. (Actually, the amounts of data are so large that even today it is difficult to *imagine* them.)

In our view, less progress has been made when it comes to modelling and forecasting. Students are still taught financial theories 'as is', with often only parenthetical reference to practical problems, most notably when it comes to forecasting.

Useful research in portfolio selection should put more emphasis on data handling and the empirical testing of models, thus better re-aligning financial theory with the nature of financial data. That means, in particular, that less emphasis should be put on obtaining numerically-precise solutions. As we said above, it is not reasonable to think that 'only the optimum' will work well: slightly changing the solution should not really change the results. (If it does, we should better not trust the results, anyway.) This is in line with the empirical evidence (eg, Gilli and Schumann, 2011a), which suggests that 'the optimum' is not required: good solutions are enough. And those are exactly the solutions that heuristics provide.

References

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

Ingo Althöfer and Klaus-Uwe Koschnick. On the convergence of "Threshold Accepting". *Applied Mathematics and Optimization*, 24(1):183–195, 1991.

J. Scott Armstrong. Forecasting with econometric methods: Folklore versus fact. *Journal of Business*, 51(4):549–564, 1978.

Richard S. Barr, Bruce L. Golden, James P. Kelly, Mauricio G. C. Resende, and William R. Stewart. Designing and reporting on computational experiments with heuristic methods. *Journal of Heuristics*, 1(1):9–32, 1995.

Frans van den Bergh and Andries P. Engelbrecht. A study of Particle Swarm Optimization particle trajectories. *Information Sciences*, 176(8):937–971, 2006.

Michael W. Brandt. Portfolio choice problems. In Yacine Aït-Sahalia and Lars P. Hansen, editors, *Handbook of Financial Econometrics*, volume 1. Elsevier, 2009.

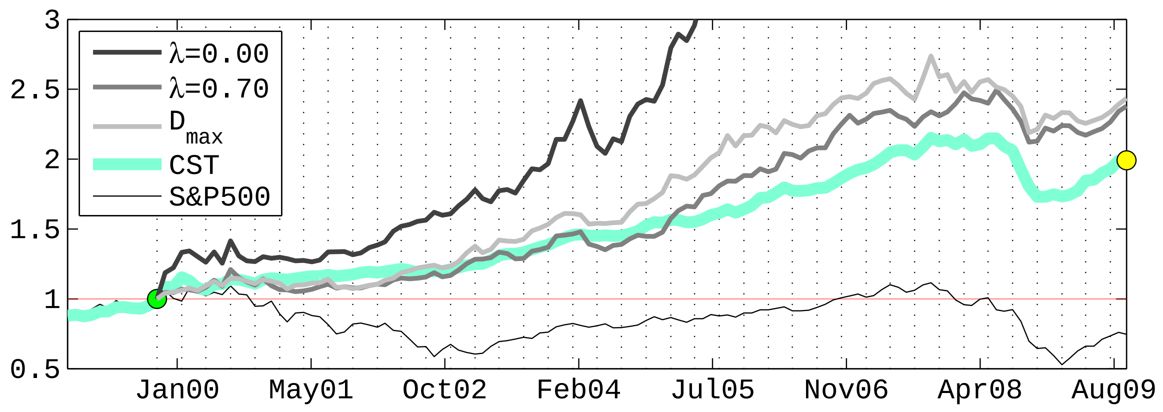


Figure 8: Median paths for portfolios minimizing the objective function (8) controlling correlations ρ_{r_P,r_M}, ρ_{r_P,r_I} and the maximum drawdown.

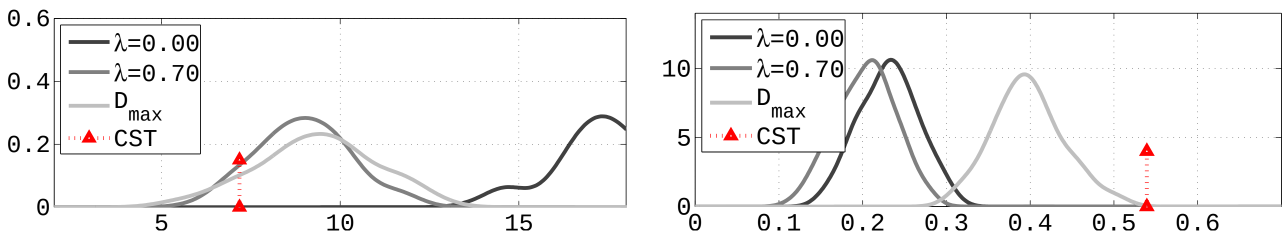


Figure 9: Density of mean yearly return (left panel) and density of correlation with the market (right panel) for the objective function (8) controlling correlations ρ_{r_P,r_M}, ρ_{r_P,r_I} and the maximum drawdown.

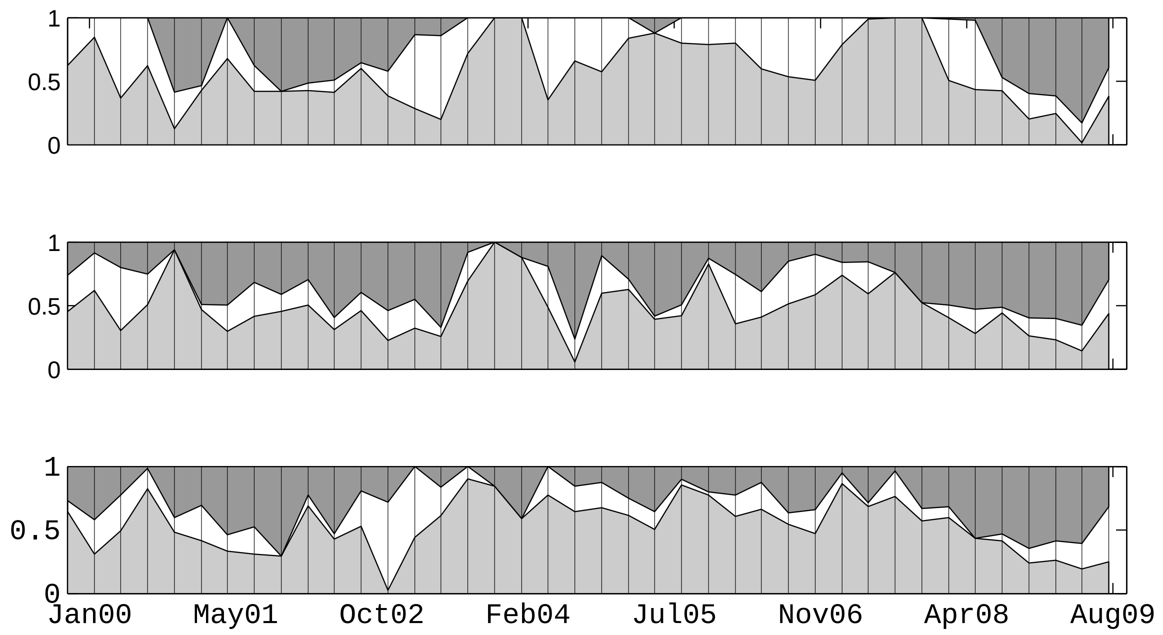


Figure 10: Median portfolio composition in terms of asset classes (equities in light gray, commodities in white and bonds in dark gray) for the objective function (8). From top to bottom $\lambda = 0$, $\lambda = 1$ and $\mathcal{D}_{\max}$.

Patrick Burns. Random portfolios for performance measurement. In Erricos John Kontoghiorghes and Cristian Gatu, editors, *Optimisation, Econometric and Financial Analysis*. Springer, 2010.

Jacob Cohen. The earth is round ($p < .05$). *American Psychologist*, 49(12):997–1003, 1994.

Robyn M. Dawes. The robust beauty of improper linear models in decision making. *American Psychologist*, 34(7): 571–582, 1979.

Robyn M. Dawes. *House of Cards – Psychology and Psychotherapy Built on Myth*. Free Press, 1994.

Ron S. Dembo. Scenario optimization. *Annals of Operations Research*, 30(1):63–80, 1991.

Gunter Dueck and Tobias Scheuer. Threshold Accepting. A general purpose optimization algorithm superior to Simulated Annealing. *Journal of Computational Physics*, 90(1): 161–175, 1990.

Gunter Dueck and Peter Winker. New concepts and algorithms for portfolio choice. *Applied Stochastic Models and Data Analysis*, 8(3):159–178, 1992.

Edwin J. Elton and Martin J. Gruber. Estimating the dependence structure of share prices – Implications for portfolio selection. *Journal of Finance*, 28(5):1203–1232, 1973.

Edwin J. Elton, Martin J. Gruber, and Thomas J. Urich. Are betas best? *Journal of Finance*, 33(5):1375–1384, 1978.

Peter A. Frost and James E. Savarino. For better performance: Constrain portfolio weights. *Journal of Portfolio Management*, 15(1):29–34, 1988.

Saul B. Gelfand and Sanjoy K. Mitter. Analysis of Simulated Annealing for Optimization. Technical Report LIDS-P-1494, MIT, 1985.

Gerd Gigerenzer. Fast and frugal heuristics: The tools of bounded rationality. In Derek J. Koehler and Nigel Harvey, editors, *Blackwell Handbook of Judgment and Decision Making*, chapter 4, pages 62–88. Blackwell Publishing, 2004.

Gerd Gigerenzer. Why heuristics work. *Perspectives on Psychological Science*, 3(1):20–29, 2008.

Philip E. Gill, Walter Murray, and Margaret H. Wright. *Practical Optimization*. Elsevier, 1986.

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

Manfred Gilli and Enrico Schumann. An Empirical Analysis of Alternative Portfolio Selection Criteria. *Swiss Finance Institute Research Paper No. 09-06*, 2009.

Manfred Gilli and Enrico Schumann. Optimization in financial engineering – an essay on 'good' solutions and misplaced exactitude. *Journal of Financial Transformation*, 28: 117–122, 2010.

Manfred Gilli and Enrico Schumann. Optimal enough? *Journal of Heuristics*, 17(4):373–387, 2011a.

Manfred Gilli and Enrico Schumann. Risk–reward optimisation for long-run investors: an empirical analysis. *European Actuarial Journal*, 1(1):303–327, Supplement 2 2011b.

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

Manfred Gilli, Enrico Schumann, Gerda Cabej, and Jonela Lula. Replicating Hedge Fund Indices with Optimization Heuristics. *Swiss Finance Institute Research Paper No. 10-22*, 2010.

Manfred Gilli, Dietmar Maringer, and Enrico Schumann. *Numerical Methods and Optimization in Finance*. Academic Press, 2011a.

Manfred Gilli, Enrico Schumann, Giacomo di Tollo, and Gerda Cabej. Constructing 130/30-portfolios with the omega ratio. *Journal of Asset Management*, 12(2):94–108, 2011b. URL `http://dx.doi.org/10.1057/jam.2010.25`.

Daniel G. Goldstein and Gerd Gigerenzer. Fast and frugal forecasting. *International Journal of Forecasting*, 25(4): 760–772, 2009.

Henk Grootveld and Winfried Hallerbach. Variance vs downside risk: Is there really that much difference? *European Journal of Operational Research*, 114(2):304–319, 1999.

Walter J. Gutjahr. A graph-based Ant System and its convergence. *Future Generation Computer Systems*, 16(9):873–888, 2000.

Ravi Jagannathan and Tongshu Ma. Risk reduction in large portfolios: Why imposing the wrong constraints helps. *Journal of Finance*, 58(4):1651–1683, August 2003.

David S. Johnson, Cecilia R. Aragon, Lyle A. McGeoch, and Catherine Schevon. Optimization by Simulated Annealing: An experimental evaluation; part I, graph partitioning. *Operations Research*, 37(6):865–892, 1989.

Eric Jondeau, Ser-Huang Poon, and Michael Rockinger. *Financial Modeling Under Non-Gaussian Distributions*. Springer, 2007.

Scott Kirkpatrick, C. Daniel Gelatt, and Mario P. Vecchi. Optimization by Simulated Annealing. *Science*, 220(4598): 671–680, 1983.

Blake LeBaron and Andreas S. Weigend. A bootstrap evaluation of the effect of data splitting on financial time series. *IEEE Transactions on Neural Networks*, 9(1):213–220, 1998. available from `citeseer.ist.psu.edu/lebaron98bootstrap.html`.

Alexander D. Lovie and Patricia Lovie. The flat maximum effect and linear scoring models for prediction. *Journal of Forecasting*, 5(3):159–168, 1986.

Spyros Makridakis and Michèle Hibon. The M3-competition: Results, conclusions and implications. *International Journal of Forecasting*, 16(4):451–476, 2000.

Spyros Makridakis, Michèle Hibon, and Claus Moser. Accuracy of forecasting: An empirical investigation. *Journal of the Royal Statistical Society. Series A (General)*, 142(2):97–145, 1979.

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

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

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

Oskar Morgenstern. *On the Accuracy of Economic Observations*. Princeton University Press, 2nd edition, 1963.

Pablo Moscato and José F. Fontanari. Stochastic versus deterministic update in Simulated Annealing. *Physics Letters A*, 146(4):204–208, 1990.

Judea Pearl. *Heuristics*. Addison-Wesley, 1984.

George Pólya. *How to Solve it*. Princeton University Press, 2nd edition, 1957.

Michael J.D. Powell. Problems related to unconstrained optimization. In W. Murray, editor, *Numerical Methods for Unconstrained Optimization*. Academic Press, 1972.

Günter Rudolph. Convergence analysis of canonical genetic algorithms. *IEEE Transactions on Neural Networks*, 5(1): 96–101, 1994.

Enrico Schumann. Take-the-best in portfolio selection. available from `http://enricoschumann.net`, 2013.

Andrea Scozzari, Fabio Tardella, Sandra Paterlini, and Thiemo Krink. Exact and heuristic approaches for the index tracking problem with UCITS constraints. *Annals of Operations Research*, 205(1):235–250, 2013.

Thomas Stützle and Marco Dorigo. A short convergence proof for a class of Ant Colony Optimization algorithms. *IEEE Transactions On Evolutionary Computation*, 6(4): 358–365, 2002.

Lloyd N. Trefethen. Numerical analysis. In Timothy Gowers, June Barrow-Green, and Imre Leader, editors, *Princeton Companion to Mathematics*. Princeton University Press, 2008.

Amos Tversky and Daniel Kahneman. Judgment under Uncertainty: Heuristics and Biases. *Science*, 185(4157):1124–1131, 1974.

John von Neumann and Herman H. Goldstine. Numerical inverting of matrices of high order. *Bulletin of the American Mathematical Society*, 53(11):1021–1099, 1947.

Peter Winker and Kai-Tai Fang. Application of Threshold-Accepting to the evaluation of the discrepancy of a set of points. *SIAM Journal on Numerical Analysis*, 34(5):2028–2042, 1997.

Peter Winker and Dietmar Maringer. The Threshold Accepting optimisation algorithm in economics and statistics. In Erricos John Kontoghiorghes and Cristian Gatu, editors, *Optimisation, Econometric and Financial Analysis*, volume 9 of *Advances in Computational Management Science*, pages 107–125. Springer, 2007.

Stelios H. Zanakis and James R. Evans. Heuristic "optimization": Why, when, and how to use it. *Interfaces*, 11(5):84–91, 1981.