

# Heuristic Optimization Methods in Econometrics\*

M. Gilli<sup>†</sup>      P. Winker<sup>‡</sup>

October 21, 2007

---

\*We are indebted to M. Meyer, G. di Tollo, and three anonymous referees for valuable comments on preliminary drafts. Both authors gratefully acknowledge financial support from the EU Commission through MRTN-CT-2006-034270 COMISEF.

<sup>†</sup>Department of Econometrics, University of Geneva and Swiss Finance Institute, Switzerland

<sup>‡</sup>Department of Economics, University of Giessen, Germany

# Contents

<b>1</b>	<b>Traditional numerical versus heuristic optimization methods</b>	<b>3</b>
1.1	Optimization in econometrics . . . . .	3
1.2	Optimization heuristics . . . . .	4
1.3	An uncomplete collection of applications . . . . .	7
1.4	Structure and instructions for use of the chapter . . . . .	8
<b>2</b>	<b>Heuristic optimization</b>	<b>9</b>
2.1	Basic concepts . . . . .	9
2.2	Trajectory methods . . . . .	11
2.2.1	Threshold methods (TM) . . . . .	11
2.2.2	Tabu search (TS) . . . . .	12
2.3	Population based methods . . . . .	12
2.3.1	Genetic algorithm (GA) . . . . .	13
2.3.2	Ant colonies (AC) . . . . .	13
2.3.3	Differential evolution (DE) . . . . .	14
2.3.4	Particle Swarm Optimization (PS) . . . . .	15
2.4	Hybrid meta-heuristics . . . . .	16
2.4.1	Basic characteristics of meta-heuristics . . . . .	17
2.4.2	Scheme for possible hybridization . . . . .	18
2.4.3	An example: Memetic algorithms (MA) . . . . .	20
<b>3</b>	<b>Stochastics of the solution</b>	<b>21</b>
3.1	Optimization as stochastic mapping . . . . .	21
3.2	Convergence of heuristics . . . . .	23
3.3	Convergence of optimization based estimators . . . . .	26
<b>4</b>	<b>General guidelines for the use of optimization heuristics</b>	<b>27</b>
4.1	Implementation . . . . .	28
4.2	Presentation of results . . . . .	34
<b>5</b>	<b>Selected applications</b>	<b>36</b>
5.1	Model selection in VAR models . . . . .	36
5.2	High breakdown point estimation . . . . .	38
<b>6</b>	<b>Conclusions</b>	<b>42</b>

# 1 Traditional numerical versus heuristic optimization methods

## 1.1 Optimization in econometrics

Before introducing heuristic optimization methods and providing an overview of some applications in econometrics, we have to motivate the use of such an optimization paradigm in econometrics. Obviously, optimization is at the core of econometric applications to real data sets, e.g., in model selection and parameter estimation. Consequently, econometrics is a field with a long tradition in applying the most up to date optimization techniques.

Maybe the most widely used technique is least squares estimation for linear models. The optimization problem in this case results in a system of linear equations which can be solved by standard techniques from linear algebra. However, in case of ill-conditioned matrices, e.g., due to very high multicollinearity, or huge models, even this simple model might pose some numerical problems.

A more demanding class of optimization problems stems from general maximum likelihood estimation. As long as the likelihood function can be considered as being globally convex, efficient numerical methods are available to solve the optimization problem. However, in this class of models, the number of notorious cases with flat likelihood functions or functions with several local optima is already quite substantial. Even a conceptually simple estimation problem such as GARCH(1,1) might sometimes result in a likelihood function which does not allow for the successful application of standard numerical approximation algorithms such as BHHH (Jerrell and Campione, 2001; Doornik and Ooms, 2003; Maringer and Winker, 2006). Some more examples will be discussed at the end of this section and in Section 5 together with the application of optimization heuristics.

Furthermore, there exist problem instances, e.g., in model selection, where traditional numerical methods cannot be applied at all due to the discreteness of the search space (Winker, 2000; Winker and Maringer, 2004; Maringer and Meyer, 2006). Again, discrete optimization problems might be categorized according to whether they can be easily solved, e.g., by simple enumeration of a small number of potential solutions, or not. In the latter case, the high inherent complexity of the problem might be a binding constraint to any deterministic approach. For an example of proven high complexity see the aggregation problem studied by Chipman and Winker (2005).

The few examples listed above and some more provided below just represent an indication for the increasing number of modelling and estimation problems in econometrics, which do not satisfy the necessary conditions for a guaranteed convergence of traditional optimization methods. For such problem instances,

the unreflecting application of standard methods might fail to provide sensible results. Unfortunately, for many applications of this type, there does not exist a simple way to find out whether a result provided by a standard method is a sensible one.

Thus, how can applied econometric work deal with the situation of optimization problems, which are highly complex from a computational point of view? The first approach would be to ignore the problem and to apply standard methods as if no problems did exist. Although this approach is certainly not advisable, it can often be observed in applications either due to ignorance about the real complexity of the optimization problem or due to a priori beliefs that the solution obtained by a standard method might nevertheless be useful in these cases. Second, one might try to simplify the specification or estimation problem until traditional methods can be applied successfully to the simplified problem. This option often had to be chosen in the past as a compromise due to a lack of computational resources. However, using this way out one sacrifices the potential gains resulting from the use of the more complex – and, hopefully, more adequate – model and risks instead to rely on a simplified – and, consequently, misspecified – problem formulation. In fact, the rapid increase in computing power allows the almost routinely use of optimization heuristics, which is the third – and often best – option.

As there is no free lunch in optimization either, this solution also comes at some cost. In fact, one trades away a deterministic solution for a stochastic one. Fortunately enough, econometricians are used to think in terms of stochastic outcomes and might take into account this additional source of randomness. Given that this aspect has been covered only marginally in reports on successful applications of optimization heuristics, the present contribution will provide an extended analysis on the implications of the additional stochastic component resulting from the use of heuristics for econometric analysis in Section 3.

## 1.2 Optimization heuristics

Before introducing a few more examples of optimization problems which cannot be tackled by classical methods, but might be suitable for the successful application of heuristics, the term “heuristic” or heuristic optimization method should be defined. Obviously, heuristic optimization methods differ from classical tools, but what exactly should we call an optimization heuristic?

Often the term heuristic is linked to algorithms mimicking some behavior found in nature, e.g., the principle of evolution through selection and mutation (genetic algorithms), the annealing process of melted iron (simulated annealing) or the self organization of ant colonies (ant colony optimization). In fact, a large number of heuristics results from such analogies extending to the “great deluge algorithm” introduced by Dueck (1993). An overview of some of these and other heuristics

as well as an attempt for their classification will be provided in Section 2.

Here, we follow a slightly more general definition of “heuristic” based on the properties of an algorithm (Winker and Maringer, 2007, p. 107). First, a heuristic should be able to provide high quality (stochastic) approximations to the global optimum at least when the amount of computational resources spent on a single run of the algorithm or on repeated runs is increased. Second, a well behaved heuristic should be robust to changes in problem characteristics, i.e. should not fit only a single problem instance, but the whole class. Also, it should not be too sensitive with regard to tuning the parameters of the algorithm or changing some constraints on the search space. In fact, these requirements lead to the third one, namely that a heuristic should be easily implemented to many problem instances, including new ones. Finally, despite of its name, a heuristic might be stochastic, but should not contain subjective elements.

Given the above definition of heuristics, one of their major advantages consists in the fact that their application does not rely on a set of strong assumptions about the optimization problem. In fact, for the implementation of most of the algorithms discussed in the following, it is sufficient to be able to evaluate the objective function for a given element of the search space. It is not necessary to assume some global property of the objective function, nor is it necessary to be able to calculate derivatives. In particular, several heuristics also allow to tackle discrete optimization problems or are even tailor made for this class of problems. On the other side, heuristics do not produce high-quality (or even exact) solutions with certainty, but rather stochastic approximations. However, when traditional methods fail, heuristics might still work in providing satisfying approximations.

Meanwhile, heuristic optimization tools have been used for some time in econometrics. Below we will provide a non exhaustive overview of typical applications. Some applications will be presented in more detail in Section 5. Nevertheless, the use of optimization heuristics for tackling highly complex problems cannot be considered yet as a well established part of econometric methodology. While several reasons might be relevant for this situation, three arguments appear to be most important:

1. Lack of knowledge about the real inherent complexity of optimization problems resulting in an inappropriate application of standard methods.
2. Lack of access to well documented implementations of heuristics. In fact, the increasing number of different heuristics and hybrids might rather confuse a potential user.
3. Difficulties in dealing with the stochastic nature of optimization results obtained by means of heuristics, e.g., in an estimation setting.

The first constraint is obviously highly problem specific. For discrete optimization

problems, complexity theory might help to check whether it might be expected that a problem can be tackled by classical approaches.<sup>1</sup> However, for discrete optimization problems in econometrics, not many results of this type are available so far. We are solely aware of a proof of NP-completeness for the optimal aggregation problem studied by Chipman and Winker (2005) in Winker (2001, pp. 261ff). For optimization problems on a continuous search space, the classical methods of complexity theory cannot be easily applied.<sup>2</sup> A formal analysis of complexity might become quite involved and, consequently, is often beyond the scope of applied econometrics. Therefore, a more heuristic approach is often used, namely grid search or restarting an optimization algorithm for different starting values. If this results in different values of the objective function, this finding might be taken as an indication of an optimization problem which is not well behaved enough for a standard deterministic algorithm. A slightly more general approach consists in using the implementation of an optimization heuristic to produce a benchmark. Running both, a standard algorithm and a heuristic, dominance of the heuristic in terms of solution quality is a clear indicator for non adequacy of the classical tool. Unfortunately, a similar conclusion cannot be drawn in the other direction, i.e. even if the classical method dominates the heuristic, this does not proof that the problem is adequate for being solved by classical techniques.

It might be easier to deal with the second issue, i.e. the access to heuristics, guidelines for the selection of a specific heuristic and their implementation. To this end, Section 2 will provide an overview of some of the most commonly used heuristics and a classification including hybrid algorithms. Furthermore, Section 4 introduces guidelines for a proper implementation of heuristics and for reporting the results obtained with a specific implementation. Given that the code of the core part of any optimization heuristic hardly exceeds a few lines, these guidelines together with a growing number of successful implementations might reduce the barrier to own applications.

With regard to the third issue, we do not think that we have to convince statisticians and econometricians about the rationality and feasibility of dealing with stochastic outcomes. However, by providing both, a formal framework and an empirical approach for the analysis of the randomness introduced by the application of heuristics in an econometric analysis, we might clarify the issues related to this argument. Section 3 will cover these topics which have not been considered so far in econometric applications of optimization heuristics to the best of our knowledge with the exception of the contributions by Maringer and Winker (2006) and Fitzenberger and Winker (2007).

---

<sup>1</sup>For a short introduction to complexity issues see Winker (2001, pp. 50ff).

<sup>2</sup>In the context of maximum likelihood estimation problems, asymptotic theory might allow to construct asymptotic tests for a global maximum (Gan and Jiang, 1999).

### 1.3 An uncomplete collection of applications of optimization heuristics in econometrics

The first applications of optimization heuristics in econometrics have been to problems with a continuous search space. One of the pioneering applications has been the study by Goffe *et al.* (1994). The authors apply an implementation of the simulated annealing heuristic to a set of test problems including switching regression, solving a rational expectation model, estimation of a frontier cost function and estimating the parameters of an artificial neural network. They demonstrate that using the optimization heuristic can improve the results both with regard to the best result out of a series of runs and with regard to the distribution of outcomes over this series. The switching regression problem considered by Goffe *et al.* (1994) has been analyzed by Dorsey and Mayer (1995) using both genetic algorithms and simulated annealing. Both Goffe *et al.* (1994) and Dorsey and Mayer (1995) reported results which improved the original solution proposed by Maddala and Nelson (1974). Nevertheless, the stochastic nature of the heuristics becomes quite obvious for this application as mentioned by Jerrell and Campione (2001) who repeated the application using genetic algorithms, evolution strategies, simulated annealing and a hybrid of the Nelder-Mead simplex and the simulated annealing algorithm. In fact, although all algorithms provide good results compared to the original solution, the algorithms never reproduced the same result for different runs.

Brooks and Morgan (1995) describe the application of simulated annealing (SA) to some low dimensional problems including the estimation of univariate normal mixtures. They find that SA performs better than classical algorithms.<sup>3</sup>

Jerrell and Campione (2001) discuss applications to a complex GARCH model. This problem is also considered by Adanu (2006) using among other methods genetic algorithms and differential evolution and by Maringer and Winker (2006) using the threshold accepting heuristic. While Jerrell and Campione (2001) just state that the algorithms do not converge repeatedly to the same value, Maringer and Winker (2006) provide an analysis of this stochastic component. The formal framework for such an approach is introduced in Section 3.

Other recent applications of optimization problems to continuous estimation problems in econometrics include fitting piecewise linear threshold autoregressive models (Baragona *et al.*, 2004), maximum likelihood estimation for threshold vector error correction models (Yang *et al.*, 2007), estimation of the parameters of stochastic ordinary differential equations (Alcock and Burrage, 2004), and calculating joint confidence bands for VAR and VEC models (Staszewska, 2007).

About the same time when the first applications to continuous estimation problems have been presented, Winker (1995) proposed the application of the thresh-

---

<sup>3</sup>Hawkins *et al.* (2001) use differential evolution for a similar problem.

old accepting heuristic for model selection in VAR models. More recently, heuristics have been repeatedly applied in the context of model selection. For example, Brooks *et al.* (2003) consider model selection and parameter estimation for different classes of models by means of simulated annealing, Acosta-González and Fernández-Rodríguez (2007) and Kapetanios (2007, 2007) use different heuristics for selecting variables and instruments in regression models, Winker and Maringer (2004) extend the approach from Winker (1995, 2000) to VEC models, i.e. for modelling partially nonstationary time series, while Maringer and Meyer (2006) consider model selection and parameter estimation in the logistic smooth transition autoregressive (LSTAR) model.

Further discrete optimization problems in econometrics which have been tackled by optimization heuristics include the identification of outliers (Baragona *et al.*, 2001) or the estimation of censored quantile regression (Fitzenberger and Winker, 2007).

Within the framework of econometric modelling many more problems are likely to be solved efficiently using heuristic optimization methods. Among these problems we have the expectation maximization (EM) introduced by Dempster *et al.* (1977) or the estimation of the parameters of a mixture of distributions such as a hidden Markov model, to mention a few. In principle, any problem in statistics and econometrics, which exhibits a high inherent complexity and cannot be solved with standard methods, is a candidate for the application of optimization heuristics. A full enumeration of all such problems being beyond the scope of this chapter, the examples provided might still provide an idea about their variety and omnipresence.

## 1.4 Structure and instructions for use of the chapter

This chapter is organized as follows. First, Section 2 will provide an overview of optimization heuristics which are used or might be used for econometric applications. Besides presenting several heuristics, this section also aims at classifying them. The following Section 3 is devoted to the issue of the additional randomness introduced to econometric modelling and estimation problems by any optimization heuristic. We will provide a formal framework for the analysis of these effects and demonstrate how they can be communicated in real applications. Section 4 aims at providing some key information for readers being interested in implementing their own heuristic to a specific econometric optimization problem. Although these guidelines focus on the threshold accepting heuristic, most of the arguments and all general comments on presenting results apply to other heuristics as well. In order to see the methods work, Section 5 provides a more detailed description of two specific applications, one with a discrete search space, the second one for a continuous parameter space related to an estimation problem.

For the reader primarily interested to learn how to implement a specific optimization heuristic, we recommend to have a short look at the specific subsection of the following section, and then to skip to Section 4. To obtain further details in the context of an actual implementation, the two examples in Section 5 might be helpful. Depending on whether the problem under study has a discrete or continuous search space, the reader might concentrate on Subsection 5.1 or 5.2, respectively. When the choice of an appropriate heuristic is also relevant, Section 2 should be covered completely. Section 3 will become relevant once an efficient implementation has been obtained if the complexity of the problem is found to high. Typically, this is indicated by differing results when restarting the algorithm with different initializations of the random number generator and improving (mean) results when increasing the number of iterations/generations.

## 2 Heuristic optimization

Heuristic optimization methods are essentially computational and therefore they have been naturally introduced following the development of electronic computing devices. First contributions go back to Bock (1958) and Croes (1958) who developed procedures to solve the traveling salesman problem, but the most significant advances in the domain have been made in the late 1980s and 1990s when the main techniques have been introduced. However, only very recently, desktop computers have reached the necessary performance to make the use of these methods really appealing.

It is beyond the scope of this introduction to present an overview of all the optimization heuristics developed during the last two decades. Osman and Laporte (1996) provide an extensive bibliography; see also Winker (2001, ch. 5). The next section will introduce the general concepts of optimization heuristics and provide a classification of the numerous existing variants and combinations. It should be noted that there does not exist a unique or generally accepted way of classifying optimization algorithms and heuristics. Nevertheless, the classification might help to identify methods which share common features and might be subject to similar strong and weak points.

### 2.1 Basic concepts

Optimization heuristics, which are sometimes also labeled approximation methods, are generally divided into two broad classes, *constructive methods* also called greedy algorithms and *local search methods*. Greedy algorithms construct the solution in a sequence of locally optimum choices (Cormen *et al.*, 1990, p. 329). We are interested in the second class. For long local search was not considered

as a mature technique and it is only recently that the method enjoys increasing interest. Some reasons for the resurgence of these methods are in particular the dramatic increase in computational resources, the ease of implementation and their flexibility as well as their successful application to many complex real-world problems. Also, recently more emphasis has been put to consider the solutions obtained with these tools as point estimates and, consequently, to provide information about their distribution. This contributes to a better understanding of the quality of a solution produced by heuristic methods. We will consider this aspect in more detail in section 3.

Local search uses only information about the solutions in the neighborhood of a current solution and is thus very similar to hill climbing where the choice of a neighbor solution locally maximizes a criterion. The classical local search method for minimizing a given objective function  $f(x)$  can be formalized as presented in Algorithm 1.

---

**Algorithm 1** Pseudo-code for the classical local search procedure.

---

```

1: Generate initial solution  $x^c$ 
2: while stopping criteria not met do
3:   Select  $x^n \in \mathcal{N}(x^c)$  (neighbor to current solution)
4:   if  $f(x^n) < f(x^c)$  then  $x^c = x^n$ 
5: end while

```

---

Hill-climbing uses information about the gradient for the selection of a neighbor  $x^n$  in statement 3 whereas local search algorithms choose the neighbors according to some random mechanism. This mechanism as well as the criteria for acceptance in statement 4, which are specific for a particular heuristic, define the way the algorithm walks through the solution space. The stopping criteria often consists in a given number of iterations.

Local search methods are generally divided into *trajectory methods* which work on a single solution and *population based* methods,<sup>4</sup> where a whole set of solutions is updated simultaneously. In the first class we find *threshold methods* and *tabu search* whereas the second class consists of *genetic algorithms*, *differential evolution* methods and *ant colonies*. All these local search methods have particular rules for either or both, the choice of a neighbor and the rules for acceptance of a solution. All methods, except tabu search, allow *uphill moves*, i.e. accept solutions which are worse than the previous one, in order to escape local minima. We give a brief sketch of these methods. For a description with more details see, e.g., Winker (2001).

---

<sup>4</sup>Some of them are also called *evolutionary algorithms* by some authors.

## 2.2 Trajectory methods

The definition of neighborhood is central to these methods and generally depends on the problem under consideration. Finding efficient neighborhood functions that lead to high quality local optima can be challenging. For some guidelines on how to construct neighborhoods see Section 4.1. We consider three different trajectory methods.<sup>5</sup> Jacobson and Yücesan (2004) present an approach for a uniform treatment of these methods under the heading of *generalized hill climbing algorithms*.

### 2.2.1 Threshold methods (TM)

The following two methods define the maximum slope for up-hill moves in the succeeding iterations. The first method uses a probabilistic acceptance criterion, while the maximum threshold is deterministic for the second.

**Simulated annealing (SA)** This refinement of the local search is based on an analogy between combinatorial optimization and the annealing process of solids. Similar to the classical local search an improvement of the solution for a move from  $x^c$  to  $x^n$  is always accepted. Moreover, the algorithm accepts also a move uphill, but only with a given probability (implemented using a uniformly distributed pseudorandom variable  $u$  in statement 6 of Algorithm 2). This probability depends on the difference between the corresponding function values and on a global parameter  $T$  (called temperature), that is gradually decreased during the process. In Algorithm 2, the rounds are implemented in statement 2 and the reduction of the probability by means of the parameter  $T$  is implemented in statement 8. The stopping criterion is defined by a given number of iterations or a number of consecutive iterations without change/improvement for the current solution  $x^c$ .

---

**Algorithm 2** Pseudo code for simulated annealing.

---

```
1: Generate initial solution  $x^c$ , initialize  $R_{\max}$  and  $T$ 
2: for  $r = 1$  to  $R_{\max}$  do
3:   while stopping criteria not met do
4:     Compute  $x^n \in \mathcal{N}(x^c)$  (neighbor to current solution)
5:     Compute  $\Delta = f(x^n) - f(x^c)$  and generate  $u$  (uniform random variable)
6:     if  $(\Delta < 0)$  or  $(e^{-\Delta/T} > u)$  then  $x^c = x^n$ 
7:   end while
8:   Reduce  $T$ 
9: end for
```

---

<sup>5</sup>Hoos and Stützle (2005) provide an introduction to stochastic local search algorithms.

**Threshold accepting (TA)** This method, suggested by Dueck and Scheuer (1990), is a deterministic analog of simulated annealing where the sequence of temperatures  $T$  is replaced by a sequence of thresholds  $\tau$ . As for SA, these threshold values typically decrease to zero while  $r$  increases to  $R_{\max}$ . Algorithm 2 becomes then

**if**  $\Delta < \tau$  **then**  $x^c = x^n$

and in statement 8 we reduce the threshold  $\tau$  instead of the temperature  $T$ .<sup>6</sup>

### 2.2.2 Tabu search (TS)

Tabu search<sup>7</sup> is particularly designed for the exploration of discrete search spaces where the set of neighbor solutions is finite. The method implements the selection of the neighborhood solution in a way to avoid cycling, i.e, visiting the same solution more than once. This is achieved by employing a short term memory, known as the tabu list and which contains the solutions which were most recently visited. In statement 3 of Algorithm 3 the construction of set  $V$  may or may not imply the examination of all neighbors of  $x^c$ .

---

**Algorithm 3** Pseudo code for tabu search.

---

```

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

```

---

The simplest way to update the memory in statement 6 is to remove older entries from the list. The stopping criterion is defined by a given number of iterations or a number of consecutive iterations without improvement for the current solution  $x^c$ .

## 2.3 Population based methods

In contrast to the trajectory methods, these approaches work simultaneously on a whole set of solutions called *population*. Therefore, population based methods might be more efficient with regard to exploring the whole search space at the cost of a higher computational load and more complex structures.

---

<sup>6</sup>See Winker (2001, pp. 137–147) for implementation details.

<sup>7</sup>See Glover and Laguna (1997) for a detailed presentation.

### 2.3.1 Genetic algorithm (GA)

This technique has been initially developed by Holland (1975).<sup>8</sup> Genetic algorithms imitate the evolutionary process of species that sexually reproduce. Thus, genetic algorithms might be considered the prototype of a population based method. New candidates for the solution are generated with a mechanism called *crossover* which combines part of the genetic patrimony of each parent and then applies a random *mutation*. If the new individual, called *child*, inherits good characteristics from his parents it will have a higher probability to survive. The fitness of the child and parent population is evaluated in function **survive** (statement 10) and the survivors can be formed either by the last generated individuals  $P''$ ,  $P'' \cup \{\text{fittest from } P'\}$ , only the fittest from  $P''$  or the fittest from  $P' \cup P''$ .

---

**Algorithm 4** Pseudo code for genetic algorithms.

---

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

---

The genetic algorithm first accepts a set of solutions (statement 3) and then constructs a set of neighbor solutions (statements 4–10, Algorithm 4). In general, a predefined number of generation provides the stopping criterion.

### 2.3.2 Ant colonies (AC)

This heuristic, first introduced by Colorni *et al.* (1992a, 1992b) imitates the way ants search for food and find their way back to their nest. First an ant explores its neighborhood randomly. As soon as a source of food is found it starts to transport food to the nest leaving traces of pheromone on the ground which will guide other ants to the source. The intensity of the pheromone traces depends on the quantity and quality of the food available at the source as well as from the distance between source and nest, as for a short distance more ants will travel on the same trail in a given time interval. As the ants preferably travel along important trails their behavior is able to optimize their work. Pheromone trails evaporate and once a source of food is exhausted the trails will disappear and the ants will start to search for other sources.

---

<sup>8</sup>For a more comprehensive introduction to GA see Reeves and Rowe (2003).

For the heuristic, the search area of the ant corresponds to a discrete set from which the elements forming the solutions are selected, the amount of food is associated with an objective function and the pheromone trail is modeled with an adaptive memory.

---

**Algorithm 5** Pseudo code for ant colonies.

---

```

1: Initialize pheromone trail
2: while stopping criteria not met do
3:   for all ants do
4:     Deposit ant randomly
5:     while solution incomplete do
6:       Select next element in solution randomly according to pheromone trail
7:     end while
8:     Evaluate objective function and update best solution
9:   end for
10:  for all ants do   Update pheromone trail (more for better solutions) end for
11: end while

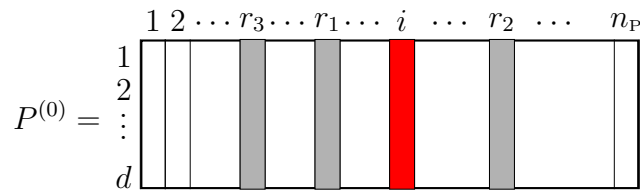
```

---

### 2.3.3 Differential evolution (DE)

Differential evolution is a population based heuristic optimization technique for continuous objective functions which has been introduced by Storn and Price (1997). The algorithm updates a population of solution vectors by addition, subtraction and crossover and then selects the fittest solutions among the original and updated population.

The initial population of  $n_P$  randomly chosen solutions can be represented by a matrix  $P^{(0)}$  of dimension  $d \times n_P$ , where  $d$  is the dimension of the domain of the function.



The algorithm constructs  $n_G$  generations of the population. A new generation is obtained in three steps. For each solution  $P_{\cdot,i}^{(0)}$ ,  $i = 1, \dots, n_P$ , represented by a column of matrix  $P^{(0)}$ , the algorithm constructs two intermediate solutions  $P_{\cdot,i}^{(v)}$  and  $P_{\cdot,i}^{(u)}$  from three randomly selected columns  $P_{\cdot,r_1}^{(0)}$ ,  $P_{\cdot,r_2}^{(0)}$  and  $P_{\cdot,r_3}^{(0)}$ .<sup>9</sup> The  $i$ th solution  $P_{\cdot,i}^{(1)}$  of the new generation is assembled from elements from  $P^{(0)}$ ,  $P^{(v)}$  and  $P^{(u)}$ . These particular steps are summarized in Algorithm 6.

---

<sup>9</sup>Note that the standard notation for one of the intermediate solutions  $P^{(u)}$  uses the symbol  $(u)$  which is not related to the uniform random variable  $u$  used in the acceptance step.

---

**Algorithm 6** Differential evolution.

---

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

---

The parameter  $F$  in statement 7 determines the length of the difference of two vectors and thus controls the speed of shrinkage in the exploration of the domain. The parameter  $CR$  together with the realization of the uniform random variable  $u \sim U(0, 1)$  in statement 9 determines the crossover, i.e. the probability for elements from  $P_{\cdot,i}^{(0)}$  and  $P_{\cdot,i}^{(v)}$  to form  $P_{\cdot,i}^{(u)}$ .<sup>10</sup> These two parameters are problem specific. Suggested values are  $F = 0.8$ ,  $CR = 0.8$  and  $n_P = 10d$  for the population size. For these values the algorithm generally performs well for a large range of problems. In statement 9 it is important to make sure that at least one component of  $P_{\cdot,i}^{(u)}$  comes from  $P_{\cdot,i}^{(v)}$ , i.e.  $\exists j$  such that  $P_{j,i}^{(u)} = P_{j,i}^{(v)}$ .

### 2.3.4 Particle Swarm Optimization (PS)

Particle swarm (PS) is a population based heuristic optimization technique for continuous functions which has been introduced by Eberhart and Kennedy (1995). The algorithm updates a population of solution vectors, called particles, by an increment called velocity. Figure 1 illustrates the updating of the position of a particle  $P_i^{(k)}$  from generation  $k$  to generation  $k+1$ . Two directions are considered, the direction from the current position of the particle to the best position so far found for the particle ( $P_i^{(k)} \rightarrow Pbest_i$ ) and the direction from the current position to the best position so far for all particles ( $P_i^{(k)} \rightarrow Pbest_{gbest}$ ). Both directions are then randomly perturbed by multiplying them with a parameter  $c$  and a uniform random variable  $u \sim U(0, 1)$  (c.f. statement 7 in Algorithm 7). A suggested value for the parameter  $c$  is 2. The sum  $\Delta v_i$  of these two randomized directions falls into a region characterized in the figure by a circle. The current velocity  $v_i^{(k+1)}$  is then obtained by updating  $v_i^{(k)}$  with the increment  $\Delta v_i$ .

---

<sup>10</sup>Different strategies for the crossover are suggested in [www.icsi.berkeley.edu/~storn/](http://www.icsi.berkeley.edu/~storn/).

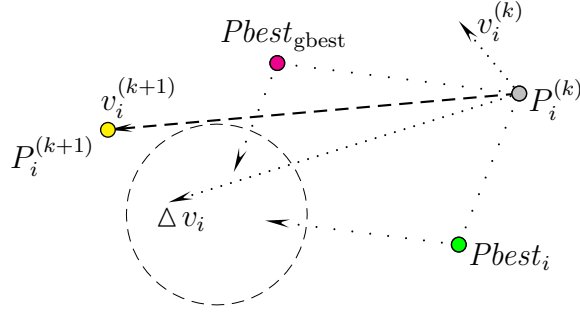


Figure 1: Updating the position of a particle  $P_i^{(k)}$  with velocity  $v_i^{(k)}$ .

Algorithm 7 summarizes the particular steps for the updating of the population of  $n_P$  particles in  $n_G$  succeeding generations.

---

**Algorithm 7** Particle swarm.

---

- 1: Initialize parameters  $n_P$ ,  $n_G$  and  $c$
  - 2: Initialize particles  $P_i^{(0)}$  and velocity  $v_i^{(0)}$ ,  $i = 1, \dots, n_P$
  - 3: Evaluate objective function  $F_i = f(P_i^{(0)})$ ,  $i = 1, \dots, n_P$
  - 4:  $Pbest = P^{(0)}$ ,  $Fbest = F$ ,  $Gbest = \min_i(F_i)$ ,  $gbest = \operatorname{argmin}_i(F_i)$
  - 5: **for**  $k = 1$  to  $n_G$  **do**
  - 6:   **for**  $i = 1$  to  $n_P$  **do**
  - 7:      $\Delta v_i = cu(Pbest_i - P_i^{(k-1)}) + cu(Pbest_{gbest} - P_i^{(k-1)})$
  - 8:      $v_i^{(k)} = v_i^{(k-1)} + \Delta v_i$
  - 9:      $P_i^{(k)} = P_i^{(k-1)} + v_i^{(k)}$
  - 10:   **end for**
  - 11: Evaluate objective function  $F_i = f(P_i^{(k)})$ ,  $i = 1, \dots, n_P$
  - 12: **for**  $i = 1$  to  $n_P$  **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**
- 

## 2.4 Hybrid meta-heuristics

In a general framework optimization heuristics are also called meta-heuristics<sup>11</sup> which can be considered as a general skeleton of an algorithm applicable to a wide range of problems. A meta-heuristic may evolve to a particular heuristic when it is specialized to solve a particular problem. Meta-heuristics are made up by different components and if components from different meta-heuristics are assembled we obtain a hybrid meta-heuristic. This allows us to imagine a large number of new techniques. The construction of hybrid meta-heuristics is

<sup>11</sup>Apart this section, we do not follow this convention, but use the term heuristic synonymously.

motivated by the need to achieve a good tradeoff between the capabilities of a heuristic to explore the search space and the possibility to exploit the experience accumulated during the search.<sup>12</sup>

In order to get a more compact view of the possibilities and types of hybrid meta-heuristics one can imagine, we will present a short and informal classification of the different meta-heuristics, describe the basic characteristics of the different components and give examples of hybrid approaches.<sup>13</sup>

### 2.4.1 Basic characteristics of meta-heuristics

In the previous sections, we demonstrated that different meta-heuristics use different strategies for the selection of a neighbor solution and for the acceptance of such a neighbor solution. We will now enumerate some of the most important features of these strategies and then provide a figure where we show what particular feature is used in the meta-heuristics presented.

- *Trajectory methods.* The current solution is slightly modified by searching within the neighborhood of the current solution. This is typically the case for threshold methods and tabu search.
- *Discontinuous methods.* Full solution space available for the new solution. The discontinuity is induced by genetic operators (crossover, mutation) as is the case for genetic algorithms, ant colonies and differential evolution and which corresponds to jumps in the search space.
- *Single agent method.* One solution per iteration is processed. Case for threshold methods and tabu search.
- *Multi-agent or population based method.* Population of searching agents who all contribute to the collective experience. Case of genetic algorithms, ant colonies and differential evolution.
- *Guided search or search with memory usage.* Incorporates some additional rules and hints on where to search. In genetic algorithms and differential evolution the population represents the memory of the recent search experience. In ant colonies the pheromone matrix represents an adaptive memory of previously visited solutions. In tabu search the tabu list provides a short term memory.
- *Unguided search or memoryless methods.* Relies perfectly on the search heuristic.

---

<sup>12</sup>A more general notion of a hybrid heuristic would also allow for combining a meta-heuristic with classical optimization tools, e.g., gradient methods.

<sup>13</sup>This presentation builds on Talbi (2002), Taillard *et al.* (2000) and Birattari *et al.* (2001).

Figure 2 summarizes the meta-heuristics and their features discussed in this section. An edge in the graph means that the feature is present in the meta-heuristic.

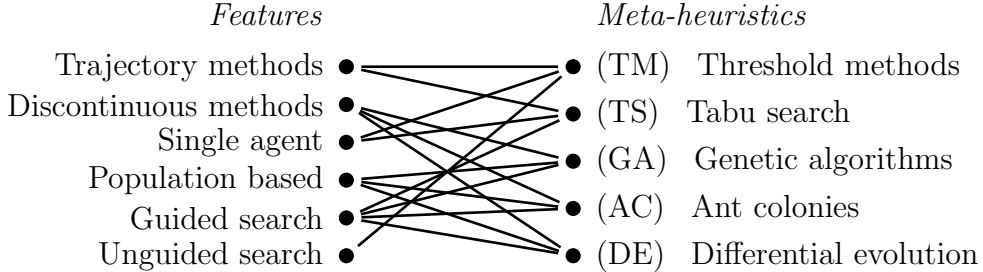


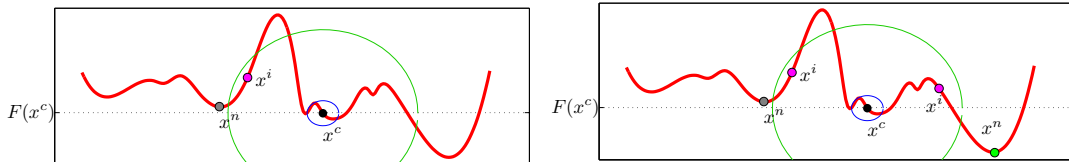
Figure 2: Meta-heuristics and their features.

### 2.4.2 Scheme for possible hybridization

We partially reproduce the classification presented in Talbi (2002). This classification is a combination of a hierarchical scheme and a flat scheme. The hierarchical scheme distinguishes between *low-level* and *high-level* hybridization and within each level we distinguish *relay* and *co-evolutionary* hybridization.

Low-level hybridization replaces a component of a given meta-heuristic by a component from another meta-heuristic. In the case of high-level hybridization different meta-heuristics are self-contained. Relay hybridization combines different meta-heuristics in a sequence whereas in co-evolutionary hybridization the different meta-heuristics cooperate. A few examples might demonstrate the corresponding four types of hybridizations:

- *Low-level relay hybrid.* As an example we could consider a simulated annealing where a neighbor  $x^n$  is obtained as following: Select a point  $x^i$  in the larger neighborhood of  $x^c$  and perform a descent local search. If this point is not accepted (left panel) we return to  $x^c$  (not  $x^i$ ) and continue (right panel).



Iterated local search (Lourenço *et al.*, 2002) and variable neighborhood search (VNS) (Mladenovic and Hansen, 1997) also fall into this class of hybrid meta-heuristics.

- *Low-level co-evolutionary hybrid.* Genetic algorithms and ant colonies perform well in the exploration of the search space but are weak in the exploitation of the solutions found. Therefore, for instance, an interesting hybridization would be to use in a genetic algorithm a greedy heuristic for the crossover and a tabu search for the mutation as indicated in Algorithm 8.

---

**Algorithm 8** Low-level co-evolutionary hybrid.

---

```

1: ...
2: Select  $P' \subset P$  (mating pool), initialize  $P'' = \emptyset$  (children)
3: for  $i = 1$  to  $n$  do
4:   Select individuals  $x^a$  and  $x^b$  at random from  $P'$ 
5:   Apply crossover to  $x^a$  and  $x^b$  to produce  $x^{\text{child}}$  (greedy algorithm)
6:   Randomly mutate produced child  $x^{\text{child}}$  (tabu search (TS))
7:    $P'' = P'' \cup x^{\text{child}}$ 
8: end for
9: ...

```

---

- *High-level relay hybrid.* Examples are the use of a greedy heuristic to generate the initial population of a genetic algorithm and/or threshold method and tabu search to improve the population obtained by the genetic algorithm as described in Algorithm 9.

---

**Algorithm 9** High-level relay hybrid.

---

```

1: Generate current population  $P$  of solutions (greedy algorithm)
2: Compute GA solution
3: Improve solution with threshold method (TM)

```

---

Another example is the use of a heuristic to optimize another heuristic, i.e. find the optimal values for the parameters.

- *High-level co-evolutionary hybrid.* In this scheme many self-contained algorithms cooperate in a parallel search to find an optimum.

For the flat scheme we distinguish the following hybridizations:

- *Homogenous* versus *Heterogeneous*. In homogenous hybrids the same meta-heuristics are used whereas heterogeneous hybrids combine different meta-heuristics.
- *Global* versus *Partial*. In global hybrids all algorithms explore the same solution space and partial hybrids work in a partitioned solution space.
- *Specialist* versus *General*. Specialist hybrids combine meta-heuristics which solve different problems whereas in general hybrids the algorithms solve all the same problem. For example a high-level relay hybrid for the optimization of another heuristic is a specialist hybrid.

Figure 3 illustrates this hierarchical and flat classification.

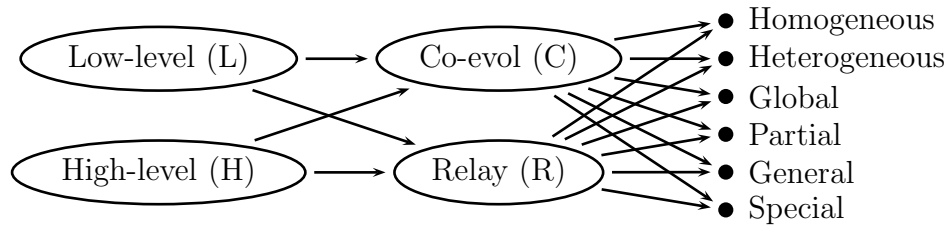


Figure 3: Scheme for possible hybridizations.

### 2.4.3 An example: Memetic algorithms (MA)

A typical example of a high-level co-evolutionary hybrid or – with regard to the flat scheme - a heterogeneous, global, general hybrid, is the so called memetic algorithm (MA) proposed by Moscato (1989). The goal of this hybrid is to combine the advantages of threshold methods and population based approaches. Each agent of a population individually runs a threshold method. However, in contrast to a simple restart scheme, the agents interact by competition and cooperation. Algorithm 10 provides the pseudo code of a memetic algorithm. The agents are positioned on a circle. Then, competition is always between neighbors on the circle. Thereby, the better solution replaces the worse neighbor. Of course, it is possible to use a more subtle acceptance criterion in this step. Cooperation is quite similar to the crossover step in genetic algorithms. In this step, the solutions of agent which are distant are combined to generate new offsprings replacing their parents. Again, the decision on whether to replace the parents or not might be based on some acceptance criterion.<sup>14</sup>

---

**Algorithm 10** Pseudo code for memetic algorithm.

---

```

1: Initialize population
2: while stopping criteria not met do
3:   for all agents do
4:     Perform optimization with threshold method
5:   end for
6:   Compete
7:   for all agents do
8:     Perform optimization with threshold method
9:   end for
10:  Cooperate
11:  Adjust acceptance criterion for threshold method
12: end while

```

---

<sup>14</sup>For a more detailed description and discussion of modified versions in the context of portfolio optimization see Maringer (2005, pp. 152ff).

### 3 Stochastics of the solution

Given the dominating classical optimization paradigm, it is not too surprising that the analysis of the results obtained by optimization heuristics concentrates on the probability to obtain the global optimum. In fact, some optimization algorithms allow to derive quite general convergence results as will be described in Subsection 3.2. However, for practical implementations, it might be more interesting to know the relationship between computational resources spent and the quality of the solution obtained.<sup>15</sup> Furthermore, often it is less the convergence of the objective function one is interested in, but, e.g., the convergence of the parameters estimated by optimizing the objective function. This aspect will be discussed in Subsection 3.3. Before turning to convergence issues, we start by providing a formal framework for the analysis of the results obtained by optimization heuristics.

#### 3.1 Optimization as stochastic mapping

Whenever repeated applications of an optimization method do not generate identical results, we have to deal with this type of stochastics, which is different from the stochastics resulting from random sampling. For the optimization heuristics introduced in the previous section, the generation of an initial solution (population), the selection of candidate solutions in each search step and sometimes also the acceptance decision are responsible for this type of randomness. It should be noted that the use of classical methods might also generate additional randomness, e.g., when using randomly generated starting values. In both cases, i.e. independent from the classification as classical or heuristic method, the outcome of a single run of the optimization procedure has to be interpreted as a random drawing from some a priori unknown distribution.

From the point of view of an applied econometrician, this additional randomness might be considered as a rather unwelcome feature of optimization heuristics as compared to standard optimization tools. However, a classical tool might well provide a deterministic solution which might be far away from the optimal solution if it is applied to a problem which does not meet the requirements for the method to converge to the global optimum. In such a situation, it is evident that a stochastic approximation to the true optimum might be preferable to a bad deterministic result. Furthermore, econometricians are well trained in dealing with the stochastics resulting from random sampling. Therefore, it seems sensible also to consider the stochastics of the outcome of optimization heuristics in some more detail.

Let  $\psi^{I,r}$  denote the result of a run  $r = 1, \dots, n_{\text{Restarts}}$  of an optimization heuristic

---

<sup>15</sup>For an example, see Brooks and Morgan (1995, p. 243).

$H$  for a given problem instance with objective function  $f$ . Thereby,  $I$  denotes a measure of the computational resources spent for a single run, e.g., number of local search steps for a local search heuristic or number of generations for a genetic algorithm. The value of the objective function obtained in run  $r$  amounts to  $f(\boldsymbol{\psi}^{I,r})$ . This value can be interpreted as a random drawing from a distribution  $D_I^H(\mu_I, \sigma_I, \dots)$ . It is assumed that the expectation and variance of this distribution exist and are finite.

Although, for most optimization heuristics  $H$ , specific knowledge about the distribution  $D_I^H(\mu_I, \sigma_I, \dots)$  is very limited for most applications,<sup>16</sup> some general properties of the distribution can be derived. First, for minimization problems, the distribution will be left censored at the global minimum of  $f$  denoted by  $f_{\min}$  in the following. Second, with increasing amount of computational resources  $I$ , the distribution should shift left and become less dispersed, i.e.  $\mu_{I'} \leq \mu_I$  and  $\sigma_{I'} \leq \sigma_I$  for all  $I' > I$ . Finally, for those applications, where asymptotical convergence in probability can be proven, the distribution becomes degenerate as  $I$  tends to infinity. It is beyond the scope of this contribution to develop a theory for  $D_I^H$ . Some ideas on how to build such a theory for the case of local search heuristics like simulated annealing and threshold accepting can be found in the convergence results presented by Aarts and Korst (1989) and Althöfer and Koschnik (1991).<sup>17</sup> A modelling approach for finite  $I$  is presented by Jacobson *et al.* (2006).

Applying an optimization heuristic repeatedly to the same problem instance makes it possible to calculate the empirical distribution function of  $f(\boldsymbol{\psi}^{I,r})$ ,  $r = 1, \dots, n_{\text{Restarts}}$ . This distribution function can be used to assess properties of  $D_I^H$  and to estimate its empirical moments. In particular, lower quantiles will be of highest interest as estimates of the best solutions which might be expected in an application to a minimization problem. Furthermore, extreme value theory might be used to obtain estimates of the unknown global minimum (Hüsler *et al.*, 2003). Finally, repeating the exercise for different amounts of computational resources  $I$  might allow estimation of an empirical rate of convergence.

Figure 4 presents results from an application of the threshold accepting heuristic to the problem of lag structure selection in VAR models.<sup>18</sup> Some details of the threshold accepting implementation are presented in Section 5.1 below. The upper left plot exhibits the empirical distribution functions of the objective function (Bayesian information criterion) for different values of  $I$  (from right to left for  $I = 100, 500, 1000, 5000, 10000$ ). As theoretically expected, the distributions move left ( $\mu_I$  is decreasing) and become steeper ( $\sigma_I$  is decreasing) with  $I$  grow-

<sup>16</sup>Pure random sampling might be an exception for problems with well-defined and easy to model search space.

<sup>17</sup>See also Rudolph (1997) and Reeves and Rowe (2003).

<sup>18</sup>For a discussion of VAR models – model selection and estimation – see Lütkepohl (2007) in this volume.

ing. The other three plots show kernel density estimates of the distribution of  $f(\psi^I)$  for different numbers of replications. Obviously, these plots confirm the findings from the empirical distribution function. In addition, they might provide information about specific properties of the objective function. For example, the multimodal shape for the lower right plot ( $I = 10\,000$ ) hints towards two local minima with a large domain of attraction, whereby the first one might be the global minimum.

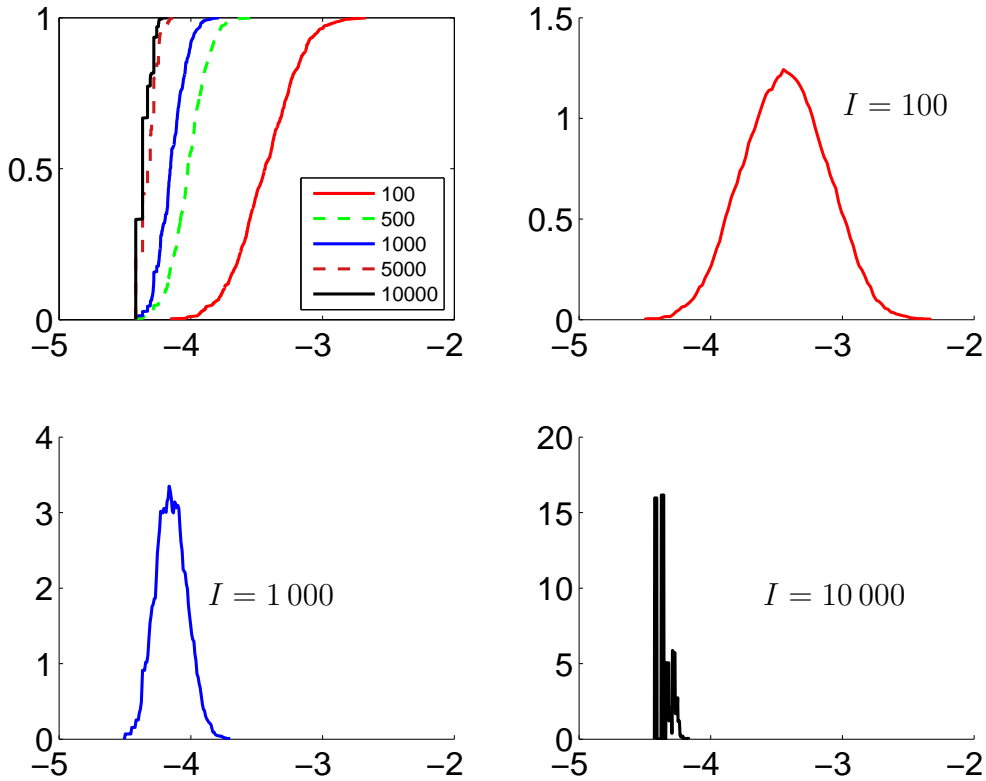


Figure 4: Empirical distribution of  $f$  for different values of  $I$  (based on  $n_{\text{Restarts}} = 1\,000$ ).

### 3.2 Convergence of heuristics

Given that the application of heuristics does not provide a deterministic result, it is of interest to analyze the factors determining the shape of the distribution of outcomes. Obviously, for a properly tuned heuristic, the amount of computational resources spent on a single run  $I$  should have a positive impact on the expected result.

For some optimization heuristics, asymptotic convergence results are available,

e.g., simulated annealing (Aarts and Korst, 1989), threshold accepting (Althöfer and Koschnik, 1991), and genetic algorithm (Reeves and Rowe, 2003, pp. 111ff). These results can be interpreted in the formal framework introduced in the previous subsection. Although the results indicate that  $\mu_I$  and  $\sigma_I$  decrease as  $I$  tends to infinity, they do not provide quantitative information, e.g., on the rate of convergence. Hence, further research is required regarding these convergence properties.

In the following, we will present the results for threshold accepting obtained by Althöfer and Koschnik (1991) as an example. Similar reasoning might apply for other convergent heuristics. The empirical assessment also focuses on threshold accepting, but, again, the approach might be used for other heuristics as well.

First, it has to be noted that the results depend on a number of assumptions regarding the problem instance. In particular, the objective function has to satisfy certain conditions, and the search space has to be connected, i.e. it should be possible to reach any point of the search space from any given starting point by performing local search steps. Second, the results are existence results. Althöfer and Koschnik (1991) prove that there exist suitable parameters for the threshold accepting implementation such that the global optimum of the objective function can be approximated at arbitrary accuracy with any fixed probability close to one by increasing the number of iterations. Unfortunately, finding the suitable parameter values is a different task.

Under the assumptions just mentioned and formalized in Althöfer and Koschnik (1991), the following convergence result is obtained: For any given  $\delta > 0$  and  $\varepsilon > 0$ , there exists a number of iterations  $I(\delta, \varepsilon)$  such that for any replication  $r = 1, \dots, n_{\text{Restarts}}$

$$P(|f(\boldsymbol{\psi}^{I,r}) - f_{\min}| < \varepsilon) > 1 - \delta, \quad (1)$$

where  $f_{\min}$  denotes the global minimum of the objective function. When considering only the best out of  $n_{\text{Restarts}}$  replications, a smaller number of iterations might be sufficient for given values of  $\delta$  and  $\varepsilon$ .

Although such a global convergence result might be considered a prerequisite for considering an optimization heuristic in an econometric context, it is not a sufficient property. In fact, at least two points deserve further attention. First, given that it is not realistic to spend an unlimited amount of computational resources, it is of interest to know at which rate  $\mu_I$  converges to  $f_{\min}$  and  $\sigma_I$  converges to zero as  $I$  tends to infinity. So far, we are not aware of any theoretical results on this issue, but we will discuss some empirical results in the following. Second, as long as  $\sigma_I$  is not zero, i.e. typically for any finite value of  $I$ , any deviation in fitting the global minimum of the objective function is linked to an error in the approximation of a specific estimator. The following subsection will provide some arguments on this issue.

As demonstrated in Figure 4, it is possible to approximate the unknown distribution function  $D_I^H$  by the empirical distribution function obtained from a number of replications  $r = 1, \dots, n_{\text{Restarts}}$ . Table 1 exhibits mean and standard deviation as well as some lower percentiles ( $p_{0.01}, p_{0.05}, p_{0.10}$ ) of these empirical distributions based on  $n_{\text{Restarts}} = 1000$  replications for the model selection problem described above.

Table 1: Statistics of empirical distributions.

$I$	$\hat{\mu}_I$	$\hat{\sigma}_I$	$p_{0.01}$	$p_{0.05}$	$p_{0.10}$
100	-3.435	0.244	-4.016	-3.835	-3.747
500	-4.019	0.138	-4.365	-4.246	-4.186
1000	-4.148	0.109	-4.418	-4.308	-4.285
5000	-4.326	0.067	-4.418	-4.418	-4.418
10000	-4.359	0.054	-4.418	-4.418	-4.418

The numbers support the descriptive results obtained from Figure 4. In particular, the value of  $-4.418$  might correspond to the global minimum. Beyond the purely descriptive approach, the empirical moments obtained allow to estimate convergence rates. For example, estimation of the model

$$\mu_I = \alpha_0 + \alpha_1 \frac{1}{I}$$

results in estimates  $\hat{\alpha}_0 = -4.291$  with an estimated standard deviation of 0.046 and a significant positive estimate of  $\alpha_1$ . These findings do not contradict a global minimum of  $-4.418$ . The  $R^2$  of this simple regression model amounts to 0.981. Of course, one might consider alternative functional forms for estimating the rate of convergence.

Although finding the global optimum is certainly preferred, this will not always be feasible for problems of high computational complexity given limited computational resources. Therefore, alternative measures of convergence might be also of interest. Jacobson *et al.* (2006) use the concept of  $\beta$ -acceptable solution probability in this context.

In particular, one might care about the number of iterations  $I$  and/or replications  $n_{\text{Restarts}}$  which is necessary to obtain a solution  $f_{\text{best}} = \min_{r=1, \dots, n_{\text{Restarts}}} f(\psi^{I,r}) \leq \beta$  with a certain probability. Although it is possible to model the  $\beta$ -acceptable solution probability as a function of the distributions  $D_I^H$ , concrete numbers have to be calculated for each particular application. Furthermore, also in this case, rates of convergence might be of interest, e.g., to determine the increase of  $I$  required to obtain a given improvement either in  $\beta$  or in the  $\beta$ -acceptable solution probability for given  $\beta$ .

Typically, an optimization heuristic is applied repeatedly to the same problem instance. Therefore, the result reported will correspond to the best out of  $n_{\text{Restarts}}$  replications, also called restarts. Obviously, the expected value for this best result will be better than for a single run. Extreme value analysis might be used to derive results on the distribution in this situation. Then, the results of the analysis might be used to derive an optimal number of restarts. For some ideas on this issue from the application point of view see the paragraph on restarts in Section 4.1 below.

Given that optimization heuristics start playing a more important role in econometrics, we argue that further research on these and similar aspects of their application is highly relevant and urgently needed.

### 3.3 Convergence of optimization based estimators

When optimization heuristics are applied to estimation problems like, e.g., censored quantile regression or (augmented) GARCH models, the stochastics of the optimization algorithm interferes with the stochastics of the estimators. We provide a formal description of this inference and demonstrate that, at least asymptotically, this interference favors the application of optimization heuristics. In fact, it is possible to derive joint convergence results.

Let us assume that the sample size  $T$  grows to infinity and that the theoretical estimator  $\hat{\psi}_T$  will converge to its “true” value  $\psi$  in probability. We consider the implementation of a convergent heuristic, i.e. we might assume that the heuristic converges in probability to the global optimum corresponding to  $\hat{\psi}_T$  with  $I$  going to infinity. Furthermore, we assume that the search space for  $\psi^T$  is a compact set and that the estimation function is continuous.<sup>19</sup> Then, given  $\delta > 0$  and  $\varepsilon > 0$  it is possible to choose the number of iterations  $I$  as a function of  $T$ ,  $\delta$ , and  $\varepsilon$  such that the estimate obtained using the heuristic  $\psi_T^I$  satisfies the following inequality:

$$\text{P}(|\psi_T^I - \hat{\psi}_T| < \varepsilon) > 1 - \delta. \quad (2)$$

Combining this result with the convergence in probability of the estimator, one obtains a joint convergence result: There exists a function  $I(T)$  for the number of iterations such that the estimate  $\psi_T^{I(T)}$  converges in probability to the true parameter vector  $\psi$ . Although this result is not precise with regard to the choice of  $I(T)$ , it bears some promises in cases where classical methods might not converge to  $\hat{\psi}_T$ . For a more detailed description see Fitzenberger and Winker (2007) and Maringer and Winker (2006).

As for the objective function itself, the convergence of parameter estimates might

---

<sup>19</sup>In passing note that these conditions are sufficient for the following results to hold but not necessary. One can probably obtain the results under much weaker assumptions.

be improved when considering the best result (with regard to  $f$ ) out of a number of  $n_{\text{Restarts}}$  restarts of the optimization heuristic. The analysis of this effect by means of extreme value theory is a topic of current research.

## 4 General guidelines for the use of optimization heuristics

The first questions to be answered for a specific application is whether to use an optimization heuristic at all and if so, which one to employ. Unfortunately, both questions do not allow for a general answer. Obviously, when a problem is known to be computational complex, e.g., due to several local minima, we recommend to apply an optimization heuristic as a benchmark for classical optimization procedures. Whenever the application of the heuristic generates better results at least for some replications, this is a clear indication for the use of such methods. Then, a more careful implementation analysis should follow. In fact, given a growing availability of code for some optimization heuristics on different software platforms, their use as a benchmark might become more standard.

The second choice is with regard to the heuristic itself. One selection can be based on the properties of the search space and the objective function given as some heuristics like DE are not well suited to tackle discrete optimization problems or problems with noise in the objective function. Another motivation for a specific optimization heuristic consists in previous (own) experience with a method for problems exhibiting similar characteristics. Finally, we argue that one should start with a simple general heuristic before turning to more problem specific hybrids.

Irrespective of the specific method chosen, any implementation of the algorithms presented in Section 2 needs particular attention with respect to a number of details, a task which is generally left to the user. For instance in the case of a trajectory method the neighborhood solution should be easy to generate, the definition of the neighborhood should not be too large and the topology of the objective function not too flat. Population based methods or evolutionary algorithms perform in each iteration a mechanism of co-operation and a mechanism of self-adaption. In a genetic algorithm information is exchanged among individuals during crossover which can be considered as co-operation, while mutation is a self-adaptation process. It is then important that pertinent information is transmitted during co-operation. The combination of two equivalent parent solutions should not produce an offspring that is different from the parents. The preservation of the diversity in the population is also very important for the efficiency of the algorithm.

Rather than covering the details of a large variety of methods this section aims

at providing some basic principles for the successful adaptation of heuristics in difficult optimization problems. We present these principles for the threshold accepting algorithm with its particularly simple structure.

We consider a minimization problem on a subset  $\Omega$  of  $\mathbb{R}^k$ :

$$\min_{x \in \Omega} f(x) \quad \Omega \subset \mathbb{R}^k. \quad (3)$$

For applications to discrete optimization problems see, e.g., Section 5.1.

## 4.1 Implementation

The implementation of the threshold accepting algorithm involves the definition of the objective function, the neighborhood and the threshold sequence. Moreover one has to specify the number of restarts  $n_{\text{Restarts}}$ ,<sup>20</sup> the number of rounds  $n_{\text{Rounds}}$  in which the threshold is reduced to zero and the number of steps  $n_{\text{Steps}}$  the algorithm searches neighbor solutions for a given value  $\tau_r$  of the threshold. Then, the number of iterations per restart is given by  $I = n_{\text{Rounds}} \times n_{\text{Steps}}$ .

### Objective function and constraints

Local search essentially proceeds in successive evaluations and comparisons of the objective function and therefore the performance of the heuristic crucially depends on its fast calculation. To improve this performance, the objective function should, whenever possible, be locally updated, i.e. the difference  $\Delta = f(x^n) - f(x^c)$  between the value of the objective function for a current solution  $x^c$  and a neighbor solution  $x^n$  should be computed directly by updating  $f(x^c)$  and not by computing  $f(x^n)$  from scratch. If possible, local updating will also improve the performance of population based algorithms. However, local updating requires a detailed analysis of the objective function. In the fields of statistics and econometrics, we are only aware of the applications to experimental design by Fang *et al.* (2003) and Fang *et al.* (2005) making use of local updating.

We use the classical traveling salesman problem from operations research to describe the idea of local updating. The problem consists in finding a tour of minimum length going through a given number of cities. Starting with some random tour, a local modification is given by exchange the sequence of two cities in the tour. Obviously, the length of the new tour has not to be calculated from scratch, but can be obtained from the length of the previous tour by subtracting the length of the removed links and adding the length of the new links. The speed up will be considerable as soon as the number of cities becomes large.

---

<sup>20</sup>Algorithm 12 below provides the pseudo code for the implementation with restarts.

In the presence of constraints the search space  $\Omega$  is a subspace  $\Omega \subset \mathbb{R}^k$ . The generation of starting and neighbor solutions which are elements of  $\Omega$  might be difficult, in particular if  $\Omega$  is not connected. Therefore,  $\mathbb{R}^k$  should be used as search space and a penalty term added to the objective function if  $x \notin \Omega$ . In order to allow the exploration of the whole search space, the penalty term is usually set at small values at the beginning of an optimization run. It is increased with the number of rounds to rather high values at the end of the run to guarantee that the final solution is a feasible one. If expressed in absolute terms, these scheme of penalty terms has to be adjusted for every application. Alternatively, one might use relative penalty terms allowing for more general implementations.

The handling of constraints is also an issue in population based algorithms unless all operators can be constructed in a way to guarantee that only feasible solutions can result.

### Neighborhood definition

The objective function should exhibit local behavior with regard to the closer neighborhood, denoted  $\mathcal{N}(x)$ , of a solution  $x$ . This means that for elements  $x^n \in \mathcal{N}(x)$ , the objective function should be closer to  $f(x)$  than for randomly selected points  $x^r$ . Of course, there is a trade-off between large neighborhoods, which guarantee non-trivial projections and small neighborhoods with a real local behavior of the objective function.

For real valued variables, a straightforward definition of the neighborhood is given by means of  $\varepsilon$ -spheres

$$\mathcal{N}(x^c) = \{x^n | x^n \in \mathcal{R}^k \text{ and } \|x^n - x^c\| < \varepsilon\},$$

where  $\|\cdot\|$  denotes the Euclidian metric. In the case of a discrete search space, one might use the Hamming metric instead (Hamming, 1950). A drawback of this definition in the Euclidian case is that the generation of elements in  $\mathcal{N}(x^c)$  might be computational costly. A simpler method consists in considering hyper-rectangles, possibly only in a few randomly selected dimensions, i.e. to select randomly a subset of elements  $x_i^c$ ,  $i \in J \subset \{1, 2, \dots, k\}$  for which  $\|x_i^n - x_i^c\| < \varepsilon$ . For many applications a choice with  $\#J = 1$ , i.e. where we modify a single element of  $x^c$ , works very well.

### Threshold sequence

The theoretical analysis of the threshold accepting algorithm in Althöfer and Koschnik (1991) does not provide a guideline on how to choose the threshold sequence. In fact, for a very small problem, Althöfer and Koschnik (1991, p. 194) even show that the optimal threshold sequence is not monotonically decreasing.

Nevertheless, for applications in econometrics, two simple procedures seem to provide useful threshold sequences. First, one might use a linear schedule decreasing to zero over the number of rounds. Obviously, for this sequence, only the first threshold value is subject to some parameter tuning. Second, one might exploit the local structure of the objective function for a data driven generation of the threshold sequence.

A motivation for this second approach can be provided for a finite search space. In this case, only threshold values corresponding to the difference of the objective function values for a pair of neighbours are relevant. Given the number of elements in real search spaces, it is not possible to calculate all these values. Instead, one uses a random sample from the distribution of such local differences. This procedure can also be applied to the cases of infinite and continuous search spaces.

Algorithm 11 provides the details of the procedure. One starts with a randomly selected point  $x^c \in \Omega$ , chooses a randomly selected neighbour  $x^n \in \mathcal{N}(x^c)$  and calculates the absolute value of the difference in the objective function  $\Delta = |f(x^c) - f(x^n)|$ . Next,  $x^c$  is replaced by  $x^n$  and the steps are repeated a certain number of times  $n_{\text{Deltas}} \geq n_{\text{Rounds}}$ . The resulting empirical distribution of  $\Delta$ s is shown in Figure 5. For very large values of the threshold, the search procedure resembles closely a pure random search. Thus, it is often useful to consider only a lower quantile of the empirical distribution function. Then, based on the  $n_{\text{Rounds}}$  smallest  $\Delta$ s, the threshold sequence can be computed as proposed by Winker and Fang (1997) and used in several applications afterwards. It is given by the quantiles corresponding to a vector of equidistant percentiles  $P_i = (n_{\text{Rounds}} - i)/n_{\text{Rounds}}$ ,  $i = 1, \dots, n_{\text{Rounds}}$ . Figure 5 also provides the threshold sequence  $\tau$  for  $n_{\text{Rounds}} = 10$  for the application presented in Section 5.2 below.

---

**Algorithm 11** Generation of threshold sequence.

---

- 1: Randomly choose  $x^c \in \Omega$
  - 2: **for**  $i = 1$  to  $n_{\text{Deltas}}$  **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  $\Delta_i$ ,  $i = 1, \dots, n_{\text{Steps}}$
  - 7: Compute threshold sequence  $\tau_r = F^{-1}\left(\frac{n_{\text{Rounds}} - r}{n_{\text{Rounds}}}\right)$ ,  $r = 1, \dots, n_{\text{Rounds}}$
- 

Instead of considering the local changes of the objective function  $\Delta$  along a path through the search space as described in Algorithm 11, one might consider several restarts to produce different trajectories of shorter length, or – in the limit – generate all  $x^c$  randomly. Obviously, when letting  $n_{\text{Deltas}}$  tend to infinity, all three methods should provide the same approximation to the distribution of local changes of the objective function. However, we do not have clear evidence which

method is best for small numbers of  $n_{\text{Deltas}}$ .

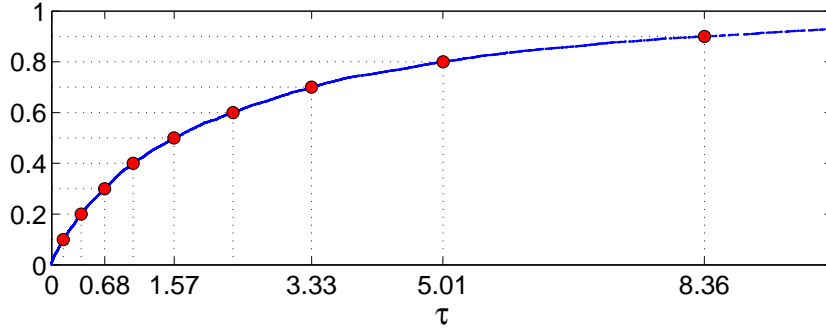


Figure 5: Empirical distribution of a sequence of  $\Delta$ s. To the  $n_{\text{Rounds}} = 10$  equally spaced percentiles on the  $y$ -axis we have the corresponding quantiles on the  $x$ -axis which constitute the threshold sequence  $\tau$ .

### Monitoring the algorithm

To gain insight into the way the algorithm explores the search space we recommend to produce a plot of the function values accepted in the succeeding rounds. This provides information about how the algorithm moves. Figure 6 exhibits such a plot for an optimization using the threshold determined in Figure 5. We observe that the amplitude of the up-hill moves diminishes in the succeeding rounds. In the last round no more up-hill moves exist.

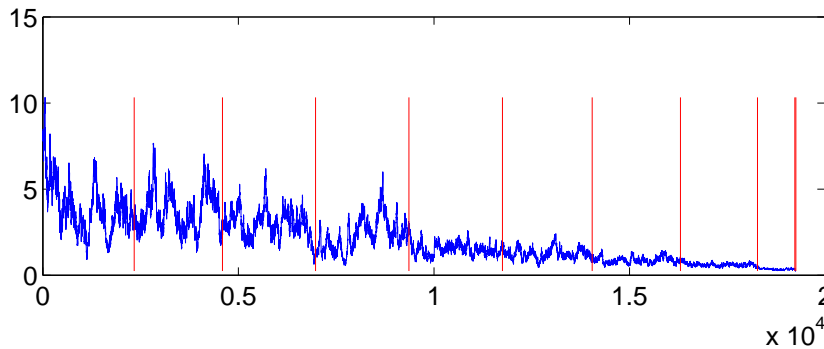


Figure 6: Function values for accepted steps in the local search. The vertical lines correspond to the beginning of a new round.

The definition of the threshold sequence is closely related to the definition of the neighborhood. We suggest the following rule of thumb: The standard deviation

of the generated distances  $\Delta$  in Algorithm 11 should be of the same order of magnitude as the standard deviation of the function values accepted in statement 8 of Algorithm 12. For the example illustrated in Figures 5 and 6 the standard deviations are respectively 2.3 and 1.5.

The graphics in Figures 5 and 6 give important information about whether the algorithm is appropriately parameterized. For instance an irregular shape (almost horizontal or vertical portions) of the cumulative distribution of the  $\Delta$ s is a clear signal for a bad local behavior of the objective function. The plot of the accepted function values for the objective function will among other also help to judge whether the choice for  $n_{\text{Rounds}}$  and  $n_{\text{Steps}}$  has been appropriate. Typically, the number of steps per round  $n_{\text{Steps}}$  exceeds the number of  $n_{\text{Rounds}}$  by far. Suggested minimum values for  $n_{\text{Rounds}}$  are about 10. However, when the total number of iterations  $I$  becomes very large,  $n_{\text{Rounds}}$  might be increased as well to obtain a closer approximation to the empirical distribution function. One might think to choose  $n_{\text{Rounds}}$  proportional to  $I$  with a low factor of proportionality in this case. Obviously, the choice of  $I$  and, consequently, for given  $n_{\text{Rounds}}$ , the choice of  $n_{\text{Steps}}$  is problem dependent. If the evaluation of the objective function is very expensive this parameter will be kept as small as possible.

## Restarts

Although the expected value  $\mu_I$  of the result improves with an increasing number of iterations  $I$ , the discussion above indicated that it might be reasonable to split available resources for several restarts  $n_{\text{Restarts}}$ . Although we are not aware of any theoretical result allowing for general conclusions, our experience with quite different problem instances indicate that a number of restarts  $n_{\text{Restarts}}$  ranging between 5 and 20 might be optimal for many applications. Optimality in this context means that for given total computational resources  $C$ , using  $n_{\text{Restarts}}$  restarts with  $I = C/n_{\text{Restarts}}$  iterations for each restarts will result in a smaller expected value of  $f$  for the best out of the  $n_{\text{Restarts}}$  runs than using all resources for a single run.

If the restarts are executed in a distributed computing environment, the optimal allocation of computing resources has to be considered differently. Again, the question is how to choose  $n_{\text{Restarts}}$  and  $I$  in order to minimize execution time for a given quality of the result  $f_{\text{sol}} = \min_{r=1, \dots, n_{\text{Restarts}}} f(\psi^{I,r}) \leq c$ .

To illustrate how this question could be answered let us consider in Figure 7 the empirical distribution of results  $f(\psi^I)$  of a heuristic for increasing values of  $I$ . We associate a Bernoulli random variable  $z$  to the solution  $f(\psi^I)$  of a single execution where  $z = 1$  if  $f(\psi^I) < c$  and  $z = 0$  else. For  $I = 10\,000$  and  $c = -1.265$  we have  $p = P(z = 1) = 0.727$  which corresponds to the percentile of  $f(\psi^I) = -1.265$  in the empirical distribution. We now consider the random variable  $x$  which

counts the number of solutions verifying  $f(\psi^I) < c$  out of  $n_{\text{Restarts}}$  solutions. We know that  $x \sim B(n_{\text{Restarts}}, p)$  is a binomial random variable. The probability for a solution being at least as good as  $f_{\text{sol}} = \min_{r=1, \dots, n_{\text{Restarts}}} f(\psi^{I,r}) \leq c$  is given by

$$\pi = 1 - P(x = 0) \quad (4)$$

from which it is possible to compute the number of restarts  $n_{\text{Restarts}}$  necessary to obtain the desired quality  $c$  with a given probability  $\pi$ . For  $x \sim \mathcal{B}(n, p)$  we have  $P(x = 0) = (1 - p)^n$  and the number of restarts  $n_{\text{Restarts}}$  we seek is given by the smallest positive integer  $n$  verifying  $\pi \leq (1 - p)^n$ .

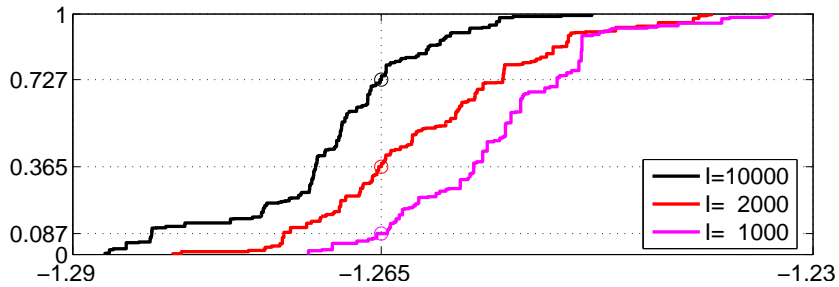


Figure 7: Empirical distribution of results  $f(\psi^I)$ .

Table 2 reproduces the values of  $n_{\text{Restarts}}$  retrieved from (4) for  $\pi = 0.99$ . We conclude that  $I = 1\,000$  and  $n_{\text{Restarts}} = 51$  would be the best choice in a distributed computing environment with at least 51 nodes, whereas  $I = 2\,000$  and  $n_{\text{Restarts}} = 11$  is the best choice if the restarts have to be executed serially.

Table 2: Number of restarts and iterations for a given quality of the result.

$I$	$R$	$C$
1 000	51	51 000
2 000	11	22 000
10 000	4	40 000

Finally, Algorithm 12 summarizes the implementation of the threshold accepting algorithm with restarts.

The algorithm behaves quite robust within a given class of problems and therefore in statement 2 the threshold sequence needs not to be recomputed for different executions with similar data.

---

**Algorithm 12** TA algorithm with restarts.

---

```
1: Initialize  $n_{\text{Restarts}}$ ,  $n_{\text{Rounds}}$  and  $n_{\text{Steps}}$ 
2: Compute threshold sequence  $\tau_r$  (Algorithm 11)
3: for  $k = 1 : n_{\text{Restarts}}$  do
4:   Randomly generate current solution  $x^c \in \mathcal{X}$ 
5:   for  $r = 1$  to  $n_{\text{Rounds}}$  do
6:     for  $i = 1$  to  $n_{\text{Steps}}$  do
7:       Generate  $x^n \in \mathcal{N}(x^c)$  and compute  $\Delta = f(x^n) - f(x^c)$ 
8:       if  $\Delta < \tau_r$  then  $x^c = x^n$ 
9:     end for
10:  end for
11:   $\xi_k = f(x^c)$ ,  $x_{(k)}^{\text{sol}} = x^c$ 
12: end for
13:  $x^{\text{sol}} = x_{(k)}^{\text{sol}}$ ,  $k \mid \xi_k = \min\{\xi_1, \dots, \xi_{n_{\text{Restarts}}}\}$ 
```

---

## 4.2 Presentation of results

Accepting the differences between the classical and the heuristic optimization paradigm has some implications on the presentation of results. In particular, all relevant information has to be provided allowing for an assessment of the quality and robustness of results. Given the novelty of heuristic optimization based approaches in econometric applications and the still limited number of publications in this field, we do not pretend to describe a generally accepted standard. In fact, many publications so far have provided only selective information on the procedure and its properties. Nevertheless, a few aspects seem of particular interest in this context and should be reported for any application. Therefore, the issues mentioned in the following might be considered as a guideline to ensure the distribution of a minimum of information required to assess econometric results which are based on a heuristic optimization method.

### Implementation details

The actual implementation of the heuristic should be completely described including all parameter settings. If a standard implementation is chosen, a reference to the literature might be sufficient. Otherwise, it is recommended to provide pseudo code of the implementation. Some examples are given in Section 2. Differences to standard implementations deserve a particularly detailed description. If possible, a reason should be given for any such departure from a standard approach.

## **Pretesting**

Typically, a preliminary test phase is run for choosing appropriate parameter values for the heuristic. Preferably, this testing phase should follow some structured approach, e.g., a grid search over some parameter values. The approach followed should be documented. In any case, it is relevant to report the number of runs in this preliminary phase as they will have an impact on the quality of the final outcomes.

## **Computational resources**

Authors should always clearly indicate how much computational resources have been spent for each of the results presented. In general, stochastic algorithms will be run repeatedly on a single problem instance. Then, the number of restarts and the number of iterations (generations) per restart should be indicated. Furthermore, information on the distribution of the results (in terms of values of the objective function and/or other variables of interest, e.g., parameter estimates) should be provided. This might be done using standard statistics such as mean, variance and quantiles if only a small number of restarts has been considered, or by density plots similar to the example provided in Figure 4 in Section 3.1.

## **Rate of convergence**

As long as an implementation of a heuristic does not provide the global optimum with probability close to one, an increase in computational resources spent (iterations/generations, restarts) should improve (expected) results. Since no theoretical results are available with regard to this convergence speed, empirical information is highly valuable. It might be provided by some graphical presentation or an econometric estimate of the rate of convergence as presented for an example in Section 3.2.

## **Generality of results**

Finally, for comparison with other methods, results on a single problem instance are not sufficient. Either multiple real problem instances should be discussed or a Monte Carlo simulation study should be conducted. In case of a Monte Carlo analysis, problem instances might be constructed in a way to allow for a response surface analysis of the results. Again, detailed information on the setup has to be provided.

## 5 Selected applications

The selected applications presented in the following subsections are not meant to present a complete analysis of the implementation of heuristics and the results obtained. Given the page constraint, they should rather provide an illustration of some of the issues addressed in the previous sections, while not all of the requirements mentioned in Section 4.2 can be met.

The first example describes the discrete problem of model selection in VAR models with an implementation of the threshold accepting heuristic. This example has been used to illustrate the stochastic dimension of working with optimization heuristics in Section 3. The second example compares implementations of threshold accepting and differential evolution for the continuous optimization problem stemming from high breakdown point estimation.

### 5.1 Model selection in VAR models

A  $p$ -dimensional vector autoregressive process  $\mathbf{X}_t$  is given by

$$\mathbf{X}_t = \sum_{k=1}^K \mathbf{\Gamma}_k \mathbf{X}_{t-k} + \boldsymbol{\varepsilon}_t, \quad (5)$$

where  $\boldsymbol{\varepsilon}_t \stackrel{iid}{\sim} N(0, \boldsymbol{\Sigma})$ . The matrices  $\mathbf{\Gamma}_k$  provide the autoregression coefficients. In the following, we assume that all component time series in  $\mathbf{X}_t$  are stationary. For a more detailed treatment of this type of multivariate time series models, estimation issues and interpretation of results, the reader is referred to Lütkepohl (2007) in this volume.

Here, we assume that a maximum lag length  $K_0$  can be derived either from economic theory or from some rule of thumb based on the number of available observations. Then, for a given realization of the process  $(\mathbf{X}_1, \dots, \mathbf{X}_T)$ , the model selection problem consists in identifying  $K$  along with those elements of  $\mathbf{\Gamma}_k$ ,  $k = 1, \dots, K$ , which are non zero in the process (5). Consequently, the search space can be described by the set  $\Omega = \{0, 1\}^{p^2 \times K_0}$ , where a zero corresponds to a parameter constraint to be zero, and a one to a parameter to be estimated freely.

For this model selection problem, different objective functions can be considered. Information criteria represent one standard approach by weighting model fit (as measured by the determinant of the fitted residuals covariance matrix  $\hat{\boldsymbol{\Sigma}}$ ) against the number of non zero parameters  $l$ . For the application presented in this subsection, the Bayesian or Schwarz information criterion  $BIC = \ln |\hat{\boldsymbol{\Sigma}}| + l \ln T/T$  is used (Schwarz, 1978). However, using a different information criterion does not require any change of the implementation besides replacing the objective function.

Given that  $\Omega$  is finite, the optimization problem can be solved by complete enumeration of all elements of  $\Omega$  and choosing the lag structure corresponding to the lowest value of the information criterion. However, even for modest values of  $p$  and  $K_0$ , the number of possible lag structures ( $2^{p^2 \times K_0}$ ) precludes this naive approach. In applied research, often only a small subspace of  $\Omega$  is considered by choosing only  $K_{\max}$  and to allow all elements in  $\Gamma_k$  to be estimated freely for  $k = 1, \dots, K_{\max}$ . Obviously, for a given  $K_0$  only  $K_0$  different models have to be compared in this constraint research space which is feasible with enumeration. However, there is no reason to expect that the optimal lag structure in  $\Omega$  happens to fall in this small subset. Hence, the application of a threshold method or another optimization heuristic suitable for discrete search spaces appears indicated.

The definition of neighborhoods on  $\Omega$  is straightforward using  $\varepsilon$ -spheres defined by the Hamming distance (Hamming, 1950). Intuitively, two lag structures with a Hamming distance of two just differ with regard to the constraints on two elements of the  $\Gamma$  matrices. One might refine these neighborhoods, e.g., by imposing further restrictions on the elements which might be exchanged in a single step (Winker, 2000, p. 92). Here, we stick to the standard case. Then, we just have to choose a value of  $\varepsilon$  large enough for not getting stuck in a bad local minima and small enough to result in a guided local search.

The second column of Table 3, labeled  $\sigma_{\Delta\mathcal{N}}$  exhibits the standard deviation of the generated distances  $\Delta$  in Algorithm 11 for the model selection problem and different choices of Hamming distances  $\varepsilon$ .<sup>21</sup> As expected, this standard deviation increases with an increasing size of the neighborhood. The third column labeled  $\sigma_{\text{accepted}}$  provides the standard deviation of the function values accepted during the runtime of the algorithm for  $I = 5\,000$ . According to the rule of thumb discussed in Section 4.1, a Hamming distance of 4 appears to be an adequate choice for this problem.

Table 3: Standard deviation of local changes and accepted solutions.

Hamming distance	$\sigma_{\Delta\mathcal{N}}$	$\sigma_{\text{accepted}}$
2	0.1706	0.9569
4	0.2578	0.5961
6	0.2916	0.7932

The threshold sequence is constructed as described in Algorithm 11 with  $n_{\text{Steps}} = 100$ . From the ordered absolute values of differences, only the lower 30%-quantile is used as threshold sequence resulting in  $n_{\text{Rounds}} = 30$ .

For the results presented in Section 3, a rather small problem instance with  $p = 3$

---

<sup>21</sup>For this simulation,  $n_{\text{Steps}}$  was set to 500 to obtain reliable estimates.

and  $K = 3$  has been considered. The problem instance exhibits different lag structures for the three equations and “holes” such that the standard approach cannot result in a proper model selection. The model selection procedure is applied to simulated data from this process of length  $T = 100$ . For the optimization, we set  $K_0 = 5$ .

When considering a single realization of the process, we find that the overall best solution is found 4 times for  $I = 500$  iterations, 18 times for  $I = 1\,000$ , 217 times for  $I = 5\,000$  and 332 times for  $I = 10\,000$  iterations when considering  $n_{\text{Restarts}} = 1\,000$  restarts. At the same time, the mean relative deviation from this best solution decreases from 22% for  $I = 100$  to 1.3% for  $I = 10\,000$  iterations. Given the low number of observations of the realization of the process, it should not be expected that the model with the lowest value of the information criterion found by the heuristic corresponds to the true data generating process. The heuristic just delivers the best or at least a good approximation for the given realization.

Of course, a test of the properties of the presented application has to be based on a much larger set of different model structures and realizations for a given model structure. However, this is beyond the present contribution. More details on model selection in VAR models including a much more comprehensive set of simulations can be found in Winker (2001, Ch. 12). A generalization of the approach to non stationary time series, i.e. in a VEC modelling context is presented by Winker and Maringer (2004) using a model selection criterion proposed by Chao and Phillips (1999) for (co)integrated series and the estimator proposed by Ahn and Reinsel (1990).<sup>22</sup> According to their findings, adequate modelling of the autoregressive part of the model, e.g., by means of optimization heuristics, might have a strong impact on the correct identification of long run relationships.

## 5.2 High breakdown point estimation

We consider<sup>23</sup> the linear regression model

$$y_i = [x_{i1} \cdots x_{ip}] \begin{bmatrix} \theta_1 \\ \vdots \\ \theta_p \end{bmatrix} + \epsilon_i \quad i = 1, \dots, n \quad (6)$$

where  $\theta \in \mathbb{R}^p$  are the parameters and  $\epsilon \sim N(0, \sigma^2 I)$  is the disturbance vector. The breakdown point of an estimator is defined as the smallest percentage of contamination in the data that may cause the estimator to be affected by an arbitrary

---

<sup>22</sup>Bauer and Wagner (2002) propose an alternative estimator which might also be used in the context of model selection for (co)integrated time series.

<sup>23</sup>Builds on a presentation given by M. Gilli and A. Marazzi at the 3rd IASC World conference in Limassol, Cyprus, 28–31 October 2005.

bias (Hampel *et al.*, 1986, chap. 2). The breakdown point for ordinary least squares is 0% but regression estimators with maximum breakdown point of 50% have been proposed. Unfortunately, the optimization problems corresponding to such high breakdown estimators cannot be solved as easily as is the case with least squares. Indeed, the objective functions are non convex and have multiple local minima. Over the last two decades, many algorithms have been proposed to solve the problem, e.g., Marazzi (1992) and Rousseeuw and Driessen (2000). These methods are based mainly on resampling techniques. More recently, the algorithms have been improved for faster execution, e.g., by Salibian-Barrera and Yohai (2004) and Rousseeuw and Driessen (2002). The resulting algorithms are complex ad hoc procedures. We demonstrate that standard heuristic optimization techniques can solve these problems easily.<sup>24</sup>

To keep this presentation as short as possible, we concentrate on the example of the *least median of squares* (LMS) estimator defined as

$$\hat{\theta}_{\text{LMS}} = \underset{\theta}{\operatorname{argmin}} Q_{\text{LMS}}(\theta)$$

where  $Q_{\text{LMS}}(\theta) = \operatorname{med}(r_1^2, \dots, r_n^2)$  is the median of the squared residuals  $r_i^2 = (y_i - x_i \cdot \theta)^2$ ,  $i = 1, \dots, n$ . Other estimators are the least trimmed squares and the S-estimator.

To illustrate the computation of an LMS estimator we use a data generation process which has been borrowed from Salibian-Barrera and Yohai (2004). We consider model (6) where 90% of the observed variables are i.i.d. standard normally distributed (gray zone in the scheme) and therefore  $\theta_i = 0$ ,  $i = 1, \dots, p$ . The remaining 10% are constituted by outliers corresponding to slopes  $\theta_2 = M/100$  and  $\theta_j = 0$  for  $j \neq 2$ . The structure of the data is summarized in Figure 8.

In the following, we generate such data for  $n = 100$ ,  $p = 10$  and  $M = 190$  and compute the LMS estimators which should not be affected by the outliers, i.e. none of the estimated parameters  $\hat{\theta}_i$ ,  $i = 1, \dots, p$  should be significantly different from zero. Figure 9 illustrates the non convex shape of the objective function to be minimized. The left panel corresponds to data generated with the artificial model described above. From the figure, one can recognize the local minimum corresponding to estimates of  $\hat{\theta}_2$  with value  $M/100 = 1.9$ . The objective function in the right panel is related to real data from a biomedical application (Brown and Hollander, 1977).

We now proceed with the description of the implementation of the threshold accepting algorithm for this optimization problem with a continuous objective function. The objective function  $F(\theta)$  has already been defined and is computed as

$$r_i = (y_i - X_{i \cdot} \theta)^2, \quad i = 1, \dots, n$$

---

<sup>24</sup>Atkinson and Weisberg (1991) have been among the first to use heuristics in this context.

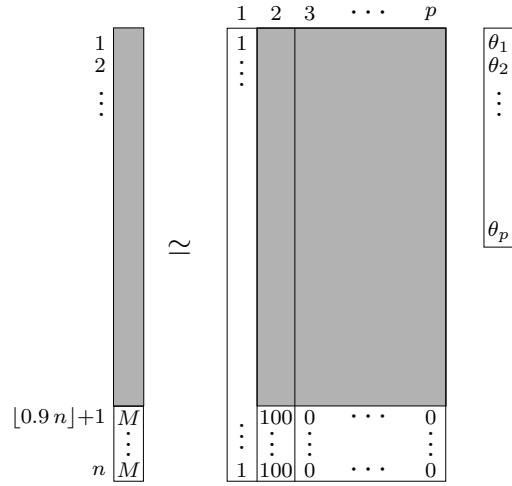


Figure 8: Structure of the data.

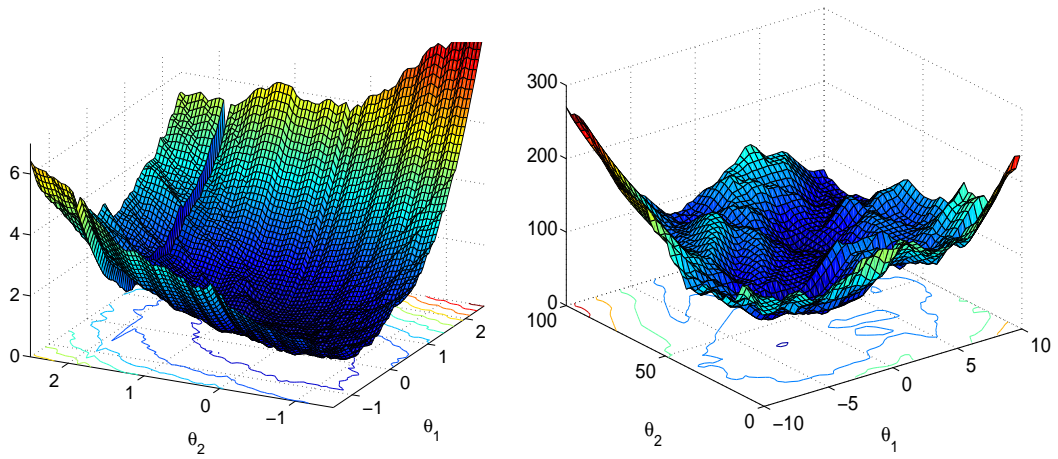


Figure 9: Shape of function  $Q_{\text{LMS}}$  for  $p = 2$ .

$$F = \text{med}(r_1^2, \dots, r_n^2) \quad ,$$

and a solution  $\theta^n$  in the neighborhood of a current solution  $\theta^c$  is generated with a mechanism defined as

$$\begin{aligned} \theta^n &= \theta^c \\ \text{Uniformly select } i &\in \{1, \dots, p\} \\ \text{Generate uniform random variable } u &\in [-h, h] \\ \theta_i^n &= \theta_i^c + u \times (1 + |\theta_i^c|) \quad , \end{aligned}$$

where  $h$  has been set to 0.40. The threshold sequence is then generated following Algorithm 11 with  $n_{\text{Rounds}} = 10$  and  $n_{\text{Steps}} = 2500$ . Figure 5 shows the computed threshold sequence and Figure 6 the accepted function values for an execution of the threshold accepting algorithm.

This continuous optimization problem can also be solved with a population based method such as differential evolution or particle swarm. For both algorithms the objective function remains unchanged. In order to make the comparison fair, the size of the population and the number of generations has been set to  $n_P = 100$ , respectively  $n_G = 250$  resulting in an overall number of function evaluations of 25 000 and which corresponds to the  $n_{\text{Rounds}} \times n_{\text{Steps}}$  evaluations in the TA algorithm.

The algorithm parameters for the differential evolution algorithm have been set to  $\text{CR} = 0.8$  and  $\text{F} = 0.75$  and  $c = 2$  for the swarm particle algorithm respectively. For both algorithms the initial population of solutions has been initialized in the domain  $\theta_i \in [-3, 3]$ ,  $i = 1, \dots, p$ .

Figure 10 shows the empirical distribution of the value of the objective function at the solution found by the threshold accepting, differential evolution and particle swarm algorithm for 200 executions.

All solutions identify the presence of outliers, i.e. none of the estimated  $\hat{\theta}_2$  approach the value  $M/100 = 1.9$ . From Figure 10 we conclude that for this case population based methods (and in particular particle swarm optimization) produces slightly better solutions than TA. The very small variance of the solutions also indicates that we only need a very limited number of restarts, say 10, to obtain with high probability a solution identifying the outliers.

Execution times per restart for our implementation in Matlab R2006a on a Intel U2500 1.20 GHz processor are 4 seconds for TA, 3.5 seconds for DE and 1.5 seconds for PS.

For the same parameter settings we also tried the Matlab function `devec3` of Rainer Storn, downloadable from <http://www.icsi.berkeley.edu/~storn/code.html>. With parameter `strategy` equal 2 the function produced similar results in same execution times.

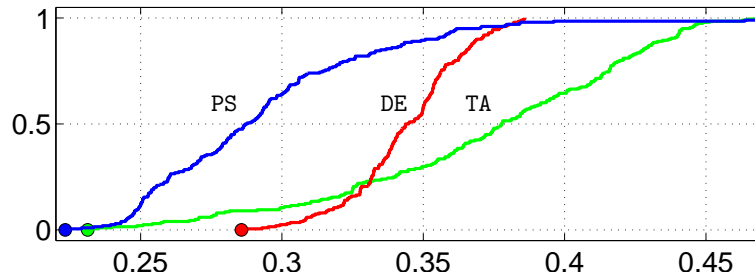


Figure 10: Empirical distribution of the value of the objective function of 200 solutions for TA, DE and PS.

## 6 Conclusions

In this contribution, the use of optimization heuristics in econometrics is motivated by the fact that many optimization problems in econometrics cannot be dealt with adequately using classical tools. Instead of simplifying the model, searching for alternative work arounds or using highly specified algorithms, we propose to employ general optimization heuristics.

The chapter provides an overview of some of the most commonly used optimization heuristics in fields close to econometrics, e.g., in statistics and finance. Furthermore, a classification of the basic algorithms and possible hybrids provides a base for selection of algorithms which might be most suitable for a specific problem instance.

In traditional applications of optimization heuristics in operations research, engineering and related fields, the inherent randomness of the results does not seem to play a major role. However, econometricians appear to be very anxious about this stochastic component. Therefore, a formal approach is proposed to deal with this additional source of randomness in the context of econometric applications. Obviously, our contribution provides only a few preliminary insights and might point towards issues deserving additional research. Nevertheless, the results presented indicate that it might be possible to include the randomness stemming from the application of optimization heuristics in the usual econometric analysis.

Another major obstacle to a more widespread use of optimization heuristics appears to be that few standard applications are available and most algorithms require some parameter tuning for a specific problem instance. In a chapter devoted to implementation issues, we tried to summarize our (limited) knowledge about these issues and to provide guidelines to the potential user of such methods. According to our understanding, it is equally important for a more general use of the heuristic methods that authors and editors become more careful about the presentation of the results. Again, we presented some guidelines without

claiming that they represent a complete set.

The chapter is complemented by the short description of two applications in econometrics, one for a discrete optimization problem in model selection and the second for a continuous optimization problem. The results indicate that these methods represent a valuable extension of the econometrician's toolbox when it comes to model and estimate slightly more complex problems than standard linear least squares. We are confident that given their performance, the methods will become a standard tool once a common standard for presentation and evaluation of the results and their randomness has been developed and is generally accepted.

## References

- Aarts, E. and J. Korst (1989). *Simulated Annealing and Boltzmann Machines*. J. Wiley and Sons. Chichester.
- Acosta-González, E. and F. Fernández-Rodríguez (2007). Model selection via genetic algorithms illustrated with cross-country growth data. *Empirical Economics* **33**, 313–337.
- Adanu, K. (2006). Optimizing the GARCH model – an application of two global and two local search methods. *Computational Economics* **28**, 277–290.
- Ahn, S.K. and G.C. Reinsel (1990). Estimation for partially nonstationary multivariate autoregressive models. *Journal of the American Statistical Association* **85**(411), 813–823.
- Alcock, J. and K. Burrage (2004). A genetic estimation algorithm for parameters of stochastic ordinary differential equations. *Computational Statistics and Data Analysis* **47**(2), 255–275.
- Althöfer, I. and K.-U. Koschnik (1991). On the convergence of threshold accepting. *Applied Mathematics and Optimization* **24**, 183–195.
- Atkinson, A.C. and S.W. Weisberg (1991). Simulated annealing for the detection of multiple outliers using least squares and least median of squares fitting. In: *Directions in robust statistics and diagnostics, Part I* (W.A. Stahel and S.W. Weisberg, Ed.). Springer-Verlag. New York.
- Baragona, R., F. Battaglia and C. Calzini (2001). Genetic algorithms for the identification of additive and innovation outliers in time series. *Computational Statistics & Data Analysis* **37**(1), 1–12.

- Baragona, R., F. Battaglia and D. Cucina (2004). Fitting piecewise linear threshold autoregressive models by means of genetic algorithms. *Computational Statistics & Data Analysis* **47**, 277–295.
- Bauer, D. and M. Wagner (2002). Estimating cointegrated systems using subspace algorithms. *Journal of Econometrics* **111**, 47–84.
- Birattari, M., L. Paquete, T. Stützle and K. Varrentrap (2001). Classification of metaheuristics and design of experiments for the analysis of components. Technical Report AIDA-2001-05. Intellektik, Technische Universität Darmstadt, Germany.
- Bock, F. (1958). An algorithm for solving “traveling salesman” and related network optimization problems. 14th ORSA meeting, St. Louis.
- Brooks, S.P. and B.J.T. Morgan (1995). Optimization using simulated annealing. *The Statistician* **44**(2), 241–257.
- Brooks, S.P., N. Friel and R. King (2003). Classical model selection via simulated annealing. *Journal of the Royal Statistical Society Series B* **65**, 503–520.
- Brown, B.W. and M. Hollander (1977). *Statistics: A Biomedical Introduction*. Wiley. New York.
- Chao, J.C. and P.C.B. Phillips (1999). Model selection in partially nonstationary vector autoregressive processes with reduced rank structure. *Journal of Econometrics* **91**, 227–271.
- Chipman, J. S. and P. Winker (2005). Optimal aggregation of linear time series models. *Computational Statistics and Data Analysis* **49**(2), 311–331.
- Coloni, A., M. Dorigo and V. Manniezzo (1992a). Distributed optimization by ant colonies. In: *Proceedings of the First European Conference on Artificial Life (ECAL-91)* (F.J. Varela and P. Bourguine, Ed.). The MIT Press. Cambridge MA. pp. 134–142.
- Coloni, A., M. Dorigo and V. Manniezzo (1992b). An investigation of some properties of an ant algorithm. In: *Parallel problem solving from nature, Vol 2*. (R. Männer and B. Manderick, Ed.). North-Holland. Amsterdam. pp. 509–520.
- Cormen, T.H., C.E. Leiserson and R.L. Rivest (1990). *Introduction to Algorithms*. MIT Electrical Engineering and Computer Science. MIT Press. Cambridge.
- Croes, G.A. (1958). A method for solving traveling salesman problems. *Operations Research* **6**, 791–812.

- Dempster, A.P., N.M. Laird and D.B. Rubin (1977). Maximum likelihood from incomplete data via the EM algorithm. *J. Roy. Statist. Soc. Ser. B* **39**, 1–38.
- Doornik, J.A. and M. Ooms (2003). Multimodality in the GARCH regression model. Technical Report 2003-W20. University of Oxford. Oxford.
- Dorsey, B. and W.J. Mayer (1995). Genetic algorithms for estimation problems with multiple optima, nondifferentiability and other irregular features. *Journal of Business and Economic Statistics* **13**, 53–66.
- Dueck, G. (1993). New optimization heuristics: The great-deluge algorithm and the record-to-record travel. *Journal of Computational Physics* **13**, 53–66.
- Dueck, G. and T. Scheuer (1990). Threshold accepting: A general purpose algorithm appearing superior to simulated annealing. *Journal of Computational Physics* **90**, 161–175.
- Eberhart, R.C. and J. 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.
- Fang, K.-T., D. Maringer, Y. Tang and P. Winker (2005). Lower bounds and stochastic optimization algorithms for uniform designs with three or four levels. *Mathematics of Computation* **75**(254), 859–878.
- Fang, Kai-Tai, Xuan Lu and P. Winker (2003). Lower bounds for centered and wrap-around  $l_2$ -discrepancies and construction of uniform designs by threshold accepting. *Journal of Complexity* **19**, 692–711.
- Fitzenberger, B. and P. Winker (2007). Improving the computation of censored quantile regressions. *Computational Statistics & Data Analysis* **52**(1), 88–108.
- Gan, L. and J. Jiang (1999). A test for global maximum. *Journal of the American Statistical Association* **94**(447), 847–854.
- Glover, F. and M. Laguna (1997). *Tabu Search*. Kluwer Academic Publishers. Boston, MA.
- Goffe, W.L., G.D. Ferrier and J. Rogers (1994). Global optimization of statistical functions with simulated annealing. *Journal of Econometrics* **60**(1-2), 65–99.
- Hamming, R.W. (1950). Error detecting and error correcting codes. *Bell System Technical Journal* **29**, 147–160.
- Hampel, F.R., E.M. Ronchetti, P.J. Rousseeuw and W.A. Stahel (1986). *Robust Statistics: The Approach based on Influence Functions*. Wiley. New York.

- Hawkins, D.S., D.M. Allen and A.J. Stromberg (2001). Determining the number of components in mixtures of linear models. *Computational Statistics & Data Analysis* **38**(1), 15–48.
- Holland, J.H. (1975). *Adaptation in Natural and Artificial Systems*. The University of Michigan Press. Ann Arbor, MI.
- Hoos, H.H. and T. Stützle (2005). *Stochastic Local Search: Foundations and Applications*. Elsevier. Amsterdam.
- Hüsler, J., P. Cruz, A. Hall and C.M. Fonseca (2003). On optimization and extreme value theory. *Methodology and Computing in Applied Probability* **5**, 183–195.
- Jacobson, S.H. and E. Yücesan (2004). Global optimization performance measures for generalized hill climbing algorithms. *Journal of Global Optimization* **29**, 173–190.
- Jacobson, S.H., S.N. Hall and L.A. McLay (2006). Visiting near-optimal solutions using local search algorithms. In: *COMPSTAT 2006, Proceedings in Computational Statistics* (Alfredo Rizzi and Maurizio Vichi, Ed.). Physica. Heidelberg. pp. 471–481.
- Jerrell, M.E. and W.A. Campione (2001). Global optimization of econometric functions. *Journal of Global Optimization* **20**(3-4), 273–295.
- Kapetanios, G. (2007). Variable selection in regression models using nonstandard optimisation of information criteria. *Computational Statistics & Data Analysis* **52**(1), 4–15.
- Kapetanios, George (2006). Choosing the optimal set of instruments from large instrument sets. *Computational Statistics & Data Analysis* **51**(2), 612–620.
- Lourenço, H.R., O. Martin and T. Stützle (2002). Iterated local search. In: *Handbook of Metaheuristics* (F. Glover and G. Kochenberger, Ed.). Vol. 57 of *International Series in Operations Research and Management Science*. Kluwer Academic Publishers. Norwell, MA. pp. 321–353.
- Lütkepohl, H. (2007). Econometric analysis with vector autoregressive models. In: *Handbook of Computational Econometrics* (E. Konthoghiorghes et al., Eds.). Elsevier. Amsterdam. p. forthcoming.
- Maddala, G.S. and F.D. Nelson (1974). Maximum likelihood methods for models of markets in disequilibrium. *Econometrica* **42**(6), 303–317.
- Marazzi, A. (1992). *Algorithms, Routines and S-functions for Robust Statistics*. Wadsworth & Brooks/Cole.

- Maringer, D. (2005). *Portfolio Management with Heuristic Optimization*. Springer. Dordrecht.
- Maringer, D. and M. Meyer (2006). Smooth transition autoregressive models – new approaches to the model selection problem. Technical Report WP010-06. University of Essex. Colchester.
- Maringer, D. and P. Winker (2006). The convergence of optimization based GARCH estimators: Theory and application. In: *COMPSTAT 2006, Proceedings in Computational Statistics* (Alfredo Rizzi and Maurizio Vichi, Ed.). Physica. Heidelberg. pp. 483–494.
- Mladenovic, N. and P. Hansen (1997). Variable neighborhood search. *Computers and Operations Research* **34**, 1097–1100.
- Moscato, P. (1989). On evolution, search, optimization, genetic algorithms and martial arts: Towards memetic algorithms. Technical report. Caltech.
- Osman, I. H. and G. Laporte (1996). Metaheuristics: A bibliography. *Annals of Operations Research* **63**, 513–623.
- Reeves, C.R. and J.E. Rowe (2003). *Genetic Algorithms – Principles and Perspectives*. Kluwer. Boston.
- Rousseeuw, P.J. and K. Van Driessen (2000). An algorithm for positive-breakdown regression based on concentration steps. In: *Data Analysis: Scientific Modeling and Practical Application* (W. Gaul, O. Opitz and M. Schader, Ed.). Springer-Verlag. Berlin. pp. 335–346.
- Rousseeuw, P.J. and K. Van Driessen (2002). Computing LTS regression for large data sets. *Estadística* **54**, 163–190.
- Rudolph, G. (1997). *Convergence Properties of Evolutionary Algorithms*. Kovač. Hamburg.
- Salibian-Barrera, M. and V.J. Yohai (2004). A fast algorithm for S-regression estimates. Technical report. University of British Columbia.
- Schwarz, G. (1978). Estimating the dimension of a model. *The Annals of Statistics* **6**, 461–464.
- Staszewska, A. (2007). Representing uncertainty about response paths: The use of heuristic optimisation methods. *Computational Statistics & Data Analysis* **52**(1), 121–132.
- Storn, R. and K. Price (1997). Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces. *Journal of Global Optimization* **11**, 341–359.

- Taillard, E.D., L.M. Gambardella, M. Gendreau and J-Y. Potvin (2000). Adaptive memory programming: A unified view of metaheuristics. *European Journal of Operational Research* **135**, 1–16.
- Talbi, E-G. (2002). A taxonomy of hybrid metaheuristics. *Journal of Heuristics* **8**, 541–564.
- Winker, P. (1995). Identification of multivariate AR-models by threshold accepting. *Computational Statistics and Data Analysis* **20**(9), 295–307.
- Winker, P. (2000). Optimized multivariate lag structure selection. *Computational Economics* **16**, 87–103.
- Winker, P. (2001). *Optimization Heuristics in Econometrics: Applications of Threshold Accepting*. Wiley. Chichester.
- Winker, P. and D. Maringer (2004). Optimal lag structure selection in VAR and VEC models. In: *New Directions in Macromodeling* (A. Welfe, Eds.). Elsevier. Amsterdam. pp. 213–234.
- Winker, P. and D. Maringer (2007). The threshold accepting optimisation algorithm in economics and statistics. In: *Optimisation, Econometric and Financial Analysis* (E. J. Kontoghiorghes and C. Gatu, Ed.). Vol. 9 of *Advances in Computational Management Science*. Springer. Berlin. pp. 107–125.
- Winker, P. 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**, 2028–2042.
- Yang, Z., Z. Tian and Z. Yuan (2007). GSA-based maximum likelihood estimation for threshold vector error correction model. *Computational Statistics & Data Analysis* **52**(1), 109–120.