Centre for Computational Finance and Economic Agents

WP009-06

Working Paper Series

Peter Winker
Dietmar Maringer

The Threshold Accepting Optimization Algorithm in Economics and Statistics

October 2006

University of Essex

CCFEA

www.ccfea.net

The Threshold Accepting Optimization Algorithm in Economics and Statistics*

Peter Winker† Dietmar Maringer‡

October 10, 2006

Abstract

Threshold Accepting (TA) is a powerful optimization heuristic. Using several examples from economics, econometrics and statistics, the issues related to implementations of TA are discussed and demonstrated.

A problem specific implementation involves the definition of a local structure on the search space, the analysis of the objective function and of constraints, if relevant, and the generation of a sequence of threshold values to be used in the acceptance-rejection-step of the algorithm.

A routine approach towards setting these implementation specific details for TA is presented, which will be partially data driven. Furthermore, fine tuning of parameters and the cost and benefit of restart versions of stochastic optimization heuristics will be discussed.
Keywords: Heuristic optimization; threshold accepting.

1 Introduction

Threshold accepting is an optimization heuristic. Reasonable features of such optimization heuristics include the following (Barr, Golden, Kelly, Resende, and Stewart, 1995, p. 12). Firstly, they should aim at good approximations to the global optimum. Secondly, they should be robust to changes in problem characteristics, tuning parameters and changes in the constraints. Thirdly, they

*We are indebted to Manfred Gilli, Erricos John Kontoghiorghes, Christian Gatu, and two anonymous referees for valuable comments on a preliminary draft of this contribution. A modified version of this paper will appear in E.J. Kontoghiorghes, C. Gatu (eds.), *Optimisation, Econometric and Financial Analysis*, Springer 2007.

†Faculty of Economics, University of Giessen

‡Centre for Computational Finance and Economic Agents (CCFEA), University of Essex

should be easy to implement to many problem instances, including new ones. Finally, a necessary requirement is that the solution approach consists of a procedure which does not depend on individual subjective elements. We will try to demonstrate that a suitable implementation of threshold accepting fulfills these requirements.

Threshold accepting is a modification of the more often used simulated annealing (Kirkpatrick, Gelatt, and Vecchi (1983)) using a deterministic acceptance criterion instead of the probabilistic one in simulated annealing. It also belongs to the class of local search methods (Aarts and Lenstra, 1997, p. 2). A classification of optimization heuristics can be found in Winker and Gilli (2004), and a more detailed description of the threshold accepting algorithm is provided by Winker (2001).

Classical or standard optimization techniques such as Newton's method are mostly based on differential calculus and first order conditions. However, this strategy requires the search space Ω to be continuous and to have just one global optimum. Many of the problems arising in statistics and economics exhibit objective functions with several local optima or discontinuities. A classification of optimization problems and some references to such cases are provided by Winker and Gilli (2004). Applied on these problems, classical optimization techniques might report the local optimum next to the starting point – provided it was able to converge in the first place. It therefore seems adequate to extend the portfolio of optimization techniques applied in these fields by optimization heuristics. There are a large number of problems in economics and statistics, including Maximum Likelihood Estimations, GMM, numerical models in economics, e.g., for computable general equilibrium models or quantitative game theory (Judd, 1998, pp. 133ff and 187ff, respectively), which are documented, for which standard optimization approaches may fail to provide solutions at all or would require tremendous amounts of computing resources. E.g., Brooks, Burke, and Persand (2001) found that commonly used econometric software may fail for a rather simple maximum likelihood estimation for the parameters of a GARCH model whereas threshold accepting is capable of finding significantly better results as Maringer (2005) reports. Therefore, the question as to whether new optimization paradigms could be useful in economics and statistics has to be answered by a clear–cut "yes".

During the last 15 years, threshold accepting has been successfully applied to many different problems ranging from classical operations research to economics and statistics. In fact, the algorithm has been introduced with an application to the famous traveling salesman problem by Dueck and Scheuer (1990). It appears that simulated annealing is still more widespread in its use, but there exist also a number of implementations of threshold accepting both in traditional operational research applications and for more specific problems from

economics and statistics. The second implementation of threshold accepting, described by Dueck and Wirsching (1991), covers multi–constraint 0–1 knapsack problems and has been included in a comparative study by Hanafi, Freville, and Abdellaoui (1996). Some further early applications in the area of operational research are cited in the bibliography provided by Osman and Laporte (1996, p. 547). A more recent survey is provided in Winker (2001).

Although we do not aim at providing a complete overview on applications of threshold accepting in statistics and economics, some further fields of application seem noteworthy. Dueck and Winker (1992) have applied threshold accepting to portfolio optimization for different risk measures, an approach taken up by Gilli and Këllezi (2002a, 2002b), recently. Winker (1995, 2000) introduces an application of threshold accepting to lag structure identification in VAR–models. Finally, threshold accepting has been applied with great success in the construction of low discrepancy experimental designs. First, Winker and Fang (1997a) obtain lower bounds for the star–discrepancy and Winker and Fang (1997b) use the approach to obtain low discrepancy U–type designs for the star–discrepancy. Next, Fang, Lin, Winker, and Zhang (2000) extend the analysis to several modifications of the L_2–discrepancy, while Fang, Ma, and Winker (2002), Fang, Lu, and Winker (2003), and Fang, Maringer, Tang, and Winker (2005) consider the centered and wrap–around L_2–discrepancy allowing to obtain lower bounds for the objective function (see subsection 2.3).

We will mention some further applications in the text when they are used as examples to demonstrate specific settings and approaches for a successful implementation of threshold accepting.

1.1 Basic features of threshold accepting

Much akin to its ancestor simulated annealing, threshold accepting is a typical local search heuristic that iteratively suggests slight random modifications to the current solution and by doing so gradually moves through the search space. TA is therefore well suited for problems where the solution space has a local structure and a notion of neighborhood around solutions can be introduced.

The second crucial property TA shares with simulated annealing (and most other heuristic search strategies) is that not only modifications for the better are accepted, but also for the worse in order to escape local optima. However, while simulated annealing uses a probabilistic criterion to decide whether to accept or reject a suggested "uphill move", TA has the deterministic criterion of a threshold value for impairments: Whenever the suggested modification improves the objective function or its degradation does not exceed a given threshold value, this modification is accepted; if the modification would degrade the objective function by more than the threshold, it is rejected. This threshold is not kept

fixed in the course of iterations, but forms a "threshold sequence" which usually makes the criterion rather tolerant in early iterations and increasingly restrictive in the later iterations. By this strategy, the algorithm can be shown to converge asymptotically to the global optimum Althöfer and Koschnik (1991).

1.2 Pseudo Code

Algorithm 1 provides the pseudo–code for a prototype threshold accepting implementation for a minimization problem.

Algorithm 1 Pseudo–code for Threshold Accepting

1: Initialize n_R, n_{S_r} and the sequence of thresholds τ_r, $r = 1, 2, \ldots, n_R$
2: Choose (randomly) feasible solution $x^c \in \Omega$
3: **for** $r = 1$ to n_R **do**
4: **for** $i = 1$ to n_{S_r} **do**
5: Choose (randomly) neighbor x^n of x^c
6: **if** $\Delta f = f(x^n) - f(x^c) < \tau_r$ **then**
7: $x^c = x^n$
8: **else**
9: leave x^c unchanged
10: **end if**
11: **end for**
12: **end for**

Thereby, f represents the objective function, which has to be minimized over the search space Ω. Of course, by replacing f with $-f$, the algorithm can also be applied to maximization problems.

Threshold accepting performs a refined local search on the search space Ω. It starts with a (randomly) generated feasible solution x^c (2:) and continues by iterating local search steps. For each step, a new candidate solution x^n has to be chosen in the neighborhood of the current solution x^c (5:). Then, the value of the objective function of both candidate solutions is compared (6:). The new candidate solution is accepted if it is better than x^c, but also if it is not much worse. The extent of an accepted worsening is limited by the current value of the threshold sequence (τ_r), which decreases to zero during the course of iterations.

The performance of the threshold accepting implementation depends on a number of settings. In particular, the definition of neighborhoods for the choice of x^n, the sequence of threshold values τ_r and, finally, the total number of iterations are most relevant. We will come back to all of these factors in the following sections of this contribution.

1.3 The basic ingredients

Approaching an optimization problem with TA demands two basic types of ingredients: ones that characterize the problem, and those that are needed for the heuristic search. The former group usually covers a proper problem statement by giving the decision variables, x, and the search space, Ω, the constraints the decision variables must meet, and the objective function, $f(x)$. Section 2 will present several relevant aspects in this respect.

For the TA implementation, the first basic ingredient is a concept of the local structure or the neighborhood of the current solution, $\mathcal{N}(x^c)$, within which new solutions are generated. What makes a suitable neighborhood depends on the optimization problem. Nonetheless there are some general requirements and approaches; section 3 addresses these issues. The second crucial ingredient is the design of the acceptance criterion. Section 4 describes how to find an appropriate threshold sequence and related issues. The third crucial ingredient for a TA is the decision of how to use the available computational time by setting the number of iterations per run and the total number of runs on the one hand and detecting when to halt a run and restart the search process on the other hand; section 5 has more details on this issue.

2 Objective Function and Constraints

2.1 Objective function

Obviously, the objective function f and the search space Ω are problem specific. Given that threshold accepting is an iterative local search procedure, it does not require the objective function f to be smooth or even differentiable. However, it has to evaluate f for many different elements $x \in \Omega$. Therefore, the efficiency of the algorithm will depend heavily on the fast calculation of $f(x)$ for any given $x \in \Omega$. Furthermore, if $f(x)$ cannot be calculated exactly, the quality of any approximation has to be taken into account.

This statement appears to be trivial for any optimization problem. However, in practice it is not. We will discuss two issues related to the objective function. First, although the objective function might be calculable in principle, the cost for doing so in terms of computational load can be quite high. Local updating, considered in more detail in subsection 3.2, often provides a remarkable speed up. The idea of local updating stems from the observation that if $x^n \in \mathcal{N}(x^c)$ is a neighbor of x^c, it is quite similar. Consequently, the objective function value for x^n could be similar to $f(x^c)$ as well. If it is possible to evaluate the difference directly, a tremendous speed up can result. For example, instead of recalculating a complete tour for the traveling salesman problem, if x^n and x^c differ only by

the ordering of a few cities, it is possible to calculate directly the difference in tour length resulting from these few differences. We will come back to this idea in subsection 3.2 as it is closely linked to the definition of local neighborhoods.

Second, there exist applications where the objective function itself cannot be easily evaluated. For example, in uniform design a given number of points has to be found in a discrete multi-dimensional space such that these points are as uniformly distributed as possible. A classical measure of the quality of such designs, i.e., the uniformity of these points, is the so called "star–discrepancy" which is described, e.g., in Winker and Fang (1997a). However, in order to evaluate this measure, a complex combinatorial problem has to be solved. Consequently, Winker and Fang (1997a) proposed to use threshold accepting to obtain a lower bound for this objective function. Now, if one would be interested in obtaining a low discrepancy design under the star–discrepancy, each evaluation of the objective function would correspond to a run of the threshold accepting heuristic itself. Fortunately, for this problem other measures of discrepancy have been developed which are much easier to compute. A similar problem comes up in the context of simulation models. If the value of the objective function is obtained by running a simulation model, again the computational complexity of the algorithm becomes quite substantial. In addition, the value of the objective function provided by the simulation will include some Monte Carlo variance. Gilli and Winker (2003) discuss how this Monte Carlo variance can be taken into account in a threshold accepting implementation.

2.2 Constraints

In most applications, the search space Ω is not a standard space like $\{0,1\}^k$ or $\mathbb{R}^k$, but only a subset of such a space resulting from some explicit constraints. If there is a large number of different constraints, this subspace might be not connected or it might prove difficult to generate elements in Ω. Also, the step of selecting $x^n \in \mathcal{N}(x^c)$ can become quite time consuming. Furthermore, the algorithm risks to get stuck in some part of the search space where no good solution can be found.

In these cases, a superior approach consists in considering the whole space $\{0,1\}^k$ or $\mathbb{R}^k$ as search space and to add a penalty term to the objective function if $x^c \notin \Omega$. If the penalty term is set at a very high level from the very beginning as sometimes suggested, this approach will just mimic the standard case, i.e., will face the same difficulties. Thus, it appears to be reasonable to start with a small penalty in order to enable the algorithm to access different parts of the search space. While the algorithm proceeds, the penalty term has to increase in order to make sure that the final solution obtained by the algorithm will be a feasible one.

2.3 Lower Bounds

Optimization heuristics like threshold accepting often provide high quality approximations to the global optimum for a given problem instance. However, given that the procedure is stochastic and convergence to the global optimum can only be expected asymptotically (see section 4), missing information about the quality of an actually found solution is often considered to be a major drawback of optimization heuristics. Of course, optimization heuristics share this potential drawback with classical optimization approaches. If, for example, a numerical procedure detects a solution of a maximum likelihood problem, this solution is determined by the first order condition. However, this condition does not guarantee a global optimum unless the function is globally convex which is rather a rare exception than the rule.

In particular for combinatorial optimization problems, lower bounds might provide a helpful tool in this context. For some problems it is possible to derive minimum values of the objective function for each instance without calculating an optimum solution. Provided that such a lower bound exists, any solution obtained by threshold accepting can be compared with this value. If the lower bound is met, the current solution is a global optimum. In this case, a further analysis of the problem instance is only required if one expects to have multiple global optima and one is interested in identifying the optimizing set instead of just a single optimum solution. If the lower bound is not met, the difference of the objective function to this lower bound provides an indicator of the maximum improvement which might be obtained by further runs of the algorithm. However, the existence of a lower bound does not imply that this lower bound can actually be reached. For the traveling salesman problem, e.g., a trivial lower bound for the round trip is the sum of the distances to the closest neighboring point for each point of the problem set. Obviously, no tour can be shorter than this sum, but in general, it has to be much longer.

Fang, Lu, and Winker (2003) provide theoretical lower bounds for some instances of the uniform design problem. Consequently, it is possible to prove that some of the designs obtained by threshold accepting represent global optima, while others differ to a small extent from the lower bounds. Again, it is not guaranteed that the lower bounds can be reached at all. Nevertheless, it has been shown that the designs obtained by the threshold accepting heuristic are not farther from a global optimum than a few percentage points in terms of the objective function.

To sum up this argument, although it is still a rare situation to have access to theoretical lower bounds for optimization problems arising in economics and statistics, such results are extremely helpful for evaluating the quality of the re-

sults obtained by optimization heuristics. Consequently, some effort should be devoted to the generation of lower bounds.

3 Local Structure and Updating

3.1 Neighborhoods

As the classification as a *local* search heuristic suggests, threshold accepting requires some notion of closeness or neighborhood for all elements of the search space Ω. For this purpose, for each element $x \in \Omega$ a neighborhood $\mathcal{N}(x) \subset \Omega$ has to be defined. Of course, given the typical size of Ω, this assignment of neighborhoods cannot be done element by element, but has to follow some algorithmic approach. Furthermore, in each iteration, an element x^n in the neighborhood of the current solution x^c has to be generated. Thus, the neighborhoods have to be constructed in a way which makes the search or construction of such neighboring elements a simple task in terms of computational complexity.

While for some of the classical combinatoric optimization problems like the traveling salesman problem there exist well–known standard concepts for constructing solutions which are neighbors to a current solution, this is not the case for most of the new optimization problems studied in economics and statistics during the last decade. However, most of these problems allow for the application of a general concept given that the search space Ω is either a subset of some real valued vector space $\mathbb{R}^k$ or of a discrete search space $\{0,1\}^k$. For these instances, ε–spheres provide a well–known concept of neighborhood on the vector space corresponding to a notion of distance provided by the Euclidean and Hamming metric Hamming (1950), respectively. Given a current solution x^c, a new element is considered a neighbor for a given ε if the distance between both elements is smaller than ε for the given distance measure. This concept is easily transferred to the subspace Ω: x^n is a neighbor to $x^c \in \Omega$ if it satisfies the distance condition and is an element of Ω itself (Winker, 2001, pp. 117ff). Thus, we define

$$\mathcal{N}(x^c) = \{x^n | x^n \in \Omega, \parallel x^n - x^c \parallel < \varepsilon\}, \tag{1}$$

where $\parallel \cdot \parallel$ stands for the distance measure.

Although being a quite general approach, the proposed construction of neighborhoods by projection of ε–spheres onto Ω will not always work. In particular, one has to check whether the resulting neighborhoods are non–trivial, i.e., contain more than a single element for reasonable choices of ε. Furthermore, the objective function should exhibit local behavior with regard to the chosen neighborhoods, i.e., for the elements in $\mathcal{N}(x^n)$, the mean value of the

objective function should be closer to $f(x^c)$ than for randomly selected elements in Ω. Both requirements result in a trade–off between large neighborhoods, which guarantee non–trivial projections, and small neighborhoods coming together with a real local behavior of the objective function.

A further argument with regard to the choice of (the size of) neighborhoods is closely connected to the features of the algorithm itself. While larger neighborhoods allow for fast movements through the search space, they also increase the peril that a global optimum is simply stepped over. Smaller neighborhoods, on the other hand, increase the number of iterations required to trespass a certain distance, e.g., in order to escape a local optimum: To escape a local optimum, a sequence of (interim) impairments of the objective function has to be accepted; the smaller the neighborhoods are, the longer this sequence is. Consequently, for smaller neighborhoods, the threshold sequence has to be more tolerant in order to be able to escape local optima.

In order to illustrate the idea of generating local neighborhoods, we consider the example of optimal aggregation of time series discussed by Chipman and Winker (2005). The authors analyze the aggregation of time series which is considered to be a central but still mainly unsolved problem in econometrics. In the specific setting considered in their paper, namely the international transmission of prices, aggregation boils down to the forming of groups of commodities and replacing the disaggregate time series by sums or weighted averages of the variables in each group. If one is interested in choosing the modes of aggregation, i.e., the composition of the groups, optimally with regard to a measure of mean–square forecast error, a highly complex integer optimization problem results. In fact, it has been shown that this problem is NP–complete (Winker, 2001, Ch. 13.9). Thus, it appears adequate to tackle the problem with an optimization heuristic like threshold accepting.

For this example, the search space is given by the set of all proper grouping matrices, which is a subspace of $\{0,1\}^{6\times 42}$ for the actual application. Thus, although the search space is finite, an exact solution by means of enumeration is not possible. The objective function is a measure of the aggregation bias in forecasting which results from using the model aggregated to only six aggregate groups as suggested by the official statistics as compared to the disaggregate data for 42 commodities. Unfortunately, the evaluation of the objective function requires some matrix inversion. Consequently, the number of iterations for this application has to be much smaller than for some of the other applications of threshold accepting mentioned.

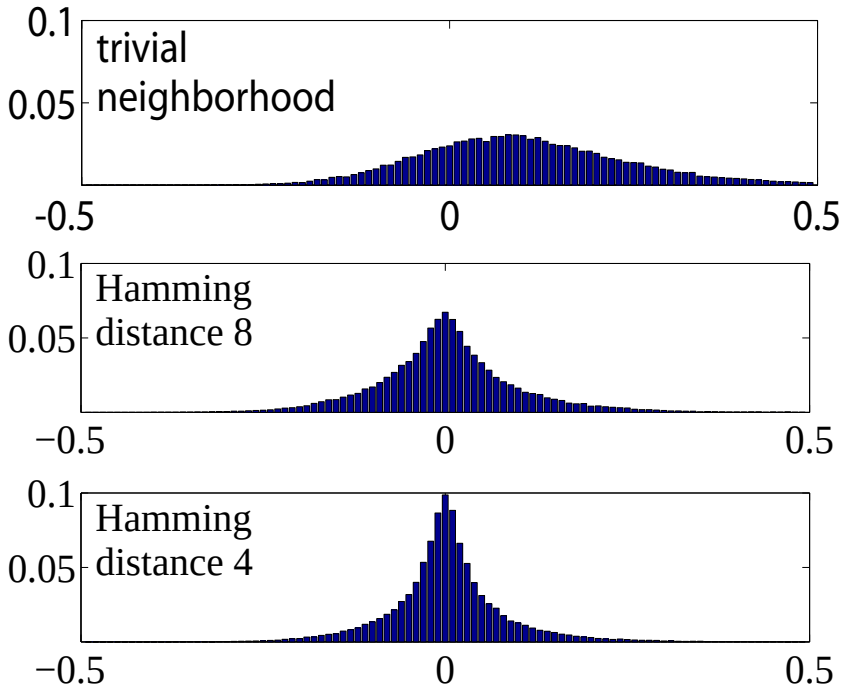
Given that the search space is defined as a subspace of some $\{0,1\}^k$, the neighborhood concept is based on the projection of ε–spheres with regard to the Hamming distance. In this example, the Hamming distance d_H between two

grouping matrices $H = (h_{ij})$ and $\tilde{H} = (\tilde{h}_{ij})$ is given by the number of differing entries:

$$d_H(H, \tilde{H}) = \sum_{i=1}^{m} \sum_{j=1}^{m^*} | h_{ij} - \tilde{h}_{ij} | . \tag{2}$$

Figure 1 shows the histogram of relative local differences of the objective function for three different neighborhood definitions, each based on 50 000 pairs of proper grouping matrices (H_k^1, H_k^2). For the trivial neighborhood represented by the top most panel, H_k^2 is randomly generated. This corresponds to setting $\varepsilon \to \infty$. The large dispersion of these relative deviations indicates that the probability of finding an acceptable new grouping in such a neighborhood is rather small unless the acceptance criterion becomes very loose, since no really local structure is imposed.

Figure 1: Local Differences for Different Neighborhood Definitions



In contrast, the lower two panels provide histograms for a Hamming distance of 4 and 8, respectively. In these cases, H_k^2 is selected randomly from $\mathcal{N}(H_k^1)$. Comparing the two lower panels it is worth noting that a shrinking

of the neighborhoods leads to a concentration of the empirical distribution of relative deviations around zero per cent, i.e., to a more locally oriented behavior of the algorithm, but at the same time reduces the number of feasible moves in each iteration. Consequently, the risk of being stuck in a local minimum increases with shrinking neighborhoods. In the application presented by Chipman and Winker (2005), the use of neighborhoods defined as spheres of radius 8 with regard to the Hamming distance proved to be a good choice, although the quality of the results did not decrease dramatically when choosing spheres of radii 4 or 12 instead.

3.2 Local Updating

In the standard version of the algorithm described so far, in each iteration, x^c and $f(x^c)$ is given. Then, an element $x^n \in \mathcal{N}(x^n)$ is generated. Finally, $f(x^n)$ is calculated in order to obtain the difference $\Delta = f(x^n) - f(x^c)$ required to test the acceptance criterion. Consequently, the total complexity of a TA implementation is given by a constant times the total number of iterations times the complexity of a single evaluation of the objective function f. While the constant might be influenced by an efficient coding of the algorithm and the choice of appropriate hardware, the total number of iterations typically will depend on the complexity of the problem at hand. Of course, a reasonable choice of neighborhoods and the threshold sequence (see the following section) might help to reduce the number of iterations required to obtain a predefined quality of the results. Here, we will concentrate on the last argument in the complexity function of the algorithm, the evaluation of the objective function (however, see also Ferrall (2004)).

At first glance, the performance of the algorithm with respect to the objective function depends solely on an efficient calculation of the objective function for given x. In fact, often considerable performance gains can be obtained by searching for more efficient code for calculating the objective function. However, sometimes a different approach is also feasible. During each iteration step, the algorithm does not require $f(x^n)$ and $f(x^c)$, but solely the difference of the objective function values $\Delta = f(x^n) - f(x^c)$. Given x^n and x^c, in some cases, a direct calculation of Δ becomes possible at much lower computational cost than the complete evaluation of f.

For example, for the traveling salesman problem, each element $x \in \Omega$ represents a tour through all N cities of the given problem. In order to calculate the length of such a tour, the sum of N distances of pairs of cities has to be calculated. If the problem is small enough, all distance pairs ($N(N-1)$) can be calculated once and for all before starting the algorithm, for larger problem instances, they have to be calculated on the fly. A typical definition of neighborhood for traveling salesman tours consists in assuming that two tours are neighbors if the

second one can be obtained from the first one by exchanging the position of two cities. Doing so, only the distances for four pairs of cities change. The rest of the tour remains unchanged. Thus, instead of calculating the sum of N distances, we have to consider only four in order to obtain Δ. The speed up resulting from this local updating idea amounts to $4/N$, i.e., it becomes the more important the larger the problem instance grows.

A similar idea is used by Fang, Lu, and Winker (2003) in an application to uniform design problems. Making use of a new representation of the objective function, they can avoid to recalculate the whole objective function when moving from one candidate solution to a neighboring one. When moving from one solution to a neighboring one, only two elements of the design matrix are exchanged, the updating requires $2(n-2)$ updates, where n denotes the number of design points, i.e., represents a measure of the problem size. In contrast, a complete evaluation would require $\frac{n(n-1)}{2}k$ comparisons, where k denotes the number of columns of the design matrix. Thus, even if up to four or even slightly more elements are exchanged in a single iteration, a tremendous speed up results which is proportional to $\frac{1}{nk}$, i.e., the larger the design under consideration, the higher the efficiency gain. For the implementation presented in Fang, Lu, and Winker (2003), the actual speed up resulting from the local updating idea ranges from around 50% for rather small problem instances ($n = 8$, $k = 10$), increasing to 80% for $n = 18$ and $k = 30$ and reaching more than 90% for $n = 100$ and $k = 8$.

4 Threshold Sequence

The final crucial ingredient of any threshold accepting implementation is the threshold sequence. By considering two extreme cases, a first intuitive idea of its influence might be gained. First, if all threshold values are set equal to zero, in each iteration, the algorithm will only accept new solutions which are at least as good as the current one. Consequently, a threshold accepting implementation with a zero threshold sequence would perform like a classical greedy local search algorithm. In general, it would converge much faster than with positive threshold values, but will get stuck in a local minimum with high probability unless the problem is a globally convex one. Second, if all values of the threshold sequence are set to a very large value which happens to be larger than any possible difference of objective function values, the algorithm will act like a random walk through the search space as any generated candidate solution will be accepted. The performance of this degenerated threshold accepting implementation will be similar to a pure random search heuristic.

Obviously, an intermediate setting is selected for any reasonable threshold accepting. Unfortunately, not much is known about how to choose this sequence in a way to improve the performance of the algorithm. The convergence result for threshold accepting provided by Althöfer and Koschnik (1991) states only the existence of an appropriate threshold sequence in order to obtain asymptotic convergence to the global optimum, but it does not provide any insights into the structure of the sequence. Consequently, the threshold sequence is often chosen in a rather ad hoc approach. Thereby, a linearly decreasing sequence appears to be preferred. The advantage of a linear cooling schedule consists in the fact, that for tuning purposes only the first value of the sequence has to be varied as it fixes the whole sequence. Existing experience with different functional forms for the threshold sequence suggests that the performance of the algorithm is quite robust with regard to the exact shape of the threshold sequence, while the size of the first threshold values has some impact. In fact, starting with too high threshold values makes the algorithm wandering around in the search space in a rather random fashion. In this case, computational resources are wasted. On the other hand, starting with a too small value for the threshold sequence, one risks to get stuck in a less favorable part of the search space. This trade–off has to be considered when conducting some tuning experiments with a threshold accepting implementation.

For discrete search spaces, a data driven method for the construction of the threshold sequence has been proposed by Winker and Fang (1997a). It is described in more detail in Winker (2001, p. 127f). It is based on the observation that for a finite (discrete) search space Ω, the set Δ of possible Δf is also finite (discrete). Obviously, for the algorithm, only the values of this set are relevant for the threshold sequence, as any value between two elements of the ordered set of possible Δf will have the same effect in the acceptance criterion. Although the size of Ω and, consequently, of Δ will exclude a complete evaluation even in the case of a finite search space, an empirical approximation to Δ can be obtained as follows. First, a large number of candidate solutions x_r^c is generated at random. Then, for each of these random designs a neighbor x_r^n is selected using the same neighborhood definition as for the optimization procedure. For each resulting pair of designs, the difference of the objective function values between the larger and the smaller value is calculated $\Delta_r = \left| f(x_r^c) - f(x_r^n) \right|$. Ordering these values provides an approximation to the distribution of local relative changes of the objective function. Finally, taking into account the trade–off between too large or too small values of the threshold sequence at the beginning of the optimization run, only a lower quantile of these sequence is actually employed as the threshold sequence. Typically, this lower quantile falls in the range of 10% to 50% for the applications considered in this contribution. Algorithm 2 provides the pseudo–code for this data driven generation of the threshold sequence.

Algorithm 2 Pseudo–code for data driven generation of threshold sequence

1: Initialize n_R, lower quantile α, $n_D[n_R/\alpha]$
2: **for** $r = 1$ to n_D **do**
3: Choose (randomly) feasible solution x_r^c
4: Choose (randomly) neighbor solution $x_r^n \in \mathcal{N}(x_r^c)$
5: Calculate $\Delta_r = |f(x_r^c) - f(x_r^n)|$
6: **end for**
7: Sort $\Delta_1 \leqslant \Delta_2 \leqslant \ldots \leqslant \Delta_{n_D}$
8: Use $\Delta_{n_R}, \ldots, \Delta_1$ as threshold sequence

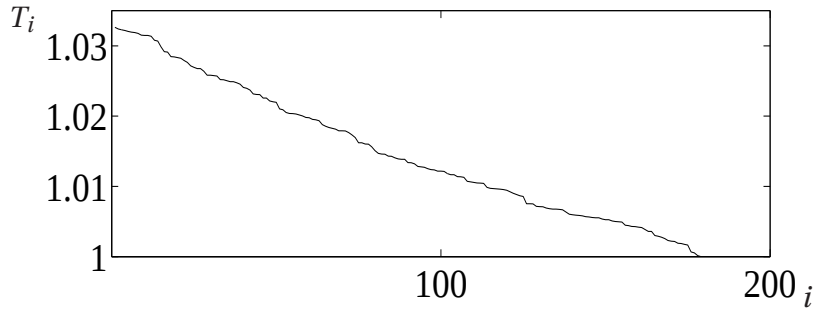
Before describing the construction of the threshold sequence for the example of the optimal aggregation of time series, a last possible modification is introduced. Instead of using an absolute definition of the thresholds, a relative version can be employed. Consequently, the decision criterion becomes $f(x^n) < f(x^c)(1 + \tau_r)$ instead of $\Delta f = f(x^n) - f(x^c) < \tau_r$. The data driven construction of a threshold sequence can be performed as before by replacing $\Delta_r = |f(x_r^c) - f(x_r^n)|$, e.g., by $\Delta_r = | f(x_r^c)/f(x_r^n) - 1 |$. The first advantage of this relative version of the threshold criterion is its independence from units of measurement. However, when employing the data driven method for constructing the threshold sequence, this advantage appears rather trivial. The second argument for employing the relative version comes into play, when the objective function takes values of widely differing orders of magnitude. Then, the threshold criterion exhibits some automatic scaling property. However, so far, there is no clear evidence for a superior performance of one or the other version of the criterion. This has to be left to future research.

As an example for the data driven generation of the threshold sequence, we refer again to the example of optimal aggregation introduced in subsection 3.1 Chipman and Winker (2005). The neighborhood definition uses the concept of the Hamming distance introduced before. As a final ingredient, a data generated threshold sequence is used which is obtained along the lines described in this section making use of the relative definition of the threshold criterion and a lower quantile $\alpha \in [0.3, 0.4]$.

Figure 2 shows a threshold sequence obtained by the data driven method for this application. The final values of the threshold sequence are equal to 1 since some of the simulated pairs of grouping matrices happen to belong to the same equivalence class of grouping matrices. Consequently, the threshold accepting algorithm degenerates to a classical greedy local search heuristic during the last iterations of the algorithm. This feature of the automatically generated threshold sequence increases the probability to finish with a local optimum. Given the

convergence result, this local optimum should be close to the global optimum and converge to it as the number of iterations of the algorithm tends to infinity.

Figure 2: A threshold sequence



5 Restart

Although it is not reported in many publications, most applications of optimization heuristics use a restarting framework, i.e., the algorithm is rerun with different seeds for the random number generator or with different tuning parameters. Then, the presented results stem from the run with best performance. Some theoretical arguments on restart implementations can be found in Fox (1994). A heuristic argument in the context of genetic algorithms is provided by Farley and Jones (1994).

In this section, we will present some rational for this approach in the context of TA. However, it will turn out that it is essential that publications report the restarting framework and provide additional information besides the "best" result. In fact, stochastic search heuristics like threshold accepting can be interpreted as a stochastic mapping

$$TA : \Omega \to f_{min},\ f_{min} \sim D_{TA}(\mu, \sigma), \tag{3}$$

where Ω is the search space and f_{min} the *random* realization of the minimum found by the algorithm for the given random number sequence. $D_{TA}(\mu, \sigma)$ denotes the distribution of f_{min} given the parameters used in the algorithm. Of course, this distribution is truncated from the left at the value of the global minimum $f_{min}^{glob} = \inf\{f(x) | x \in \Omega\}$. Consequently, D_{TA} will not be a normal distribution. It might be an interesting subject for future studies to analyze the prop-

erties of this distribution for different applications and different optimization heuristics.

However, for practical purposes it might suffice to know about the existence of such a distribution. Furthermore, it might be obvious that an increase in the total number of iterations of a local search heuristic like threshold accepting should reduce the expected value μ of the distribution and – due to the left truncation – probably the standard deviation σ, too. Instead of using a parametric distributional assumption, we might use the empirical distribution obtained from a simulation study. For this purpose, the threshold accepting implementation is run several times with differing initializations (seeds) of the random number generator. For each run i, $i = 1,\ldots,N$, the minimum f^i_{min} is stored. Using the set $\{f^i_{min}|i = 1,\ldots,N\}$, it is possible to calculate the empirical mean and standard deviation for the best solution found by the algorithm or to provide empirical quantiles. It is also possible to report the minimum $\min\{f^i_{min}|i = 1,\ldots,N\}$ of all runs. In fact, this is the typical value provided in publications – sometimes accompanied by the number of runs N. However, from the interpretation of *TA* as a stochastic mapping, it becomes evident that this value is not a robust statistic. Thus, it should not be the only information provided.

We recommend to provide at least the following information: The number of restarts N, the empirical mean and standard deviation or – alternatively – some quantiles of the empirical distribution. Furthermore, it should be reported for which parameter settings of the algorithm restarting has been considered.

Given these arguments about restarting, a further question has to be considered. Obviously, performing N restarts uses valuable computational resources. Instead, a smaller number of restarts – or a single run – with more iterations could be performed. Using more restarts provides a better approximation to the underlying distribution $D_{TA}(\mu_1,\sigma_1)$ for the given number of iterations. On the other hand, using less restarts and more iterations results in an approximation of lower quality to a different distribution $D_{TA}(\mu_2,\sigma_2)$ with a smaller expectation $\mu_2 < \mu_1$. In fact, given the convergence property of threshold accepting, D_{TA} will degenerate to a one point distribution in the global minimum with the number of iterations going to infinity. Unfortunately, not much is known about the rate of this convergence. Thus, it remains an empirical issue to decide about this trade–off.

To conclude this section, we present empirical findings for an implementation of threshold accepting to the well known traveling salesman problem. The problem instance with 442 points is described in more detail in Winker (2001, Ch. 8). For the analysis of a restarting situation, the following experimental setting is chosen. The threshold sequence is fixed for all runs to the same linear

Table 1: Restart threshold accepting

	Iterations per try		
	100 000	1 000 000	10 000 000
Restarts	10 000	1000	100
Mean	5317.07	5170.52	5138.22
SD	52.83	28.69	21.81
10%	5251.11	5135.45	5112.22
5%	5234.50	5124.83	5107.41
1%	5204.20	5109.90	5098.23

sequence. Then, the threshold accepting implementation is run with 100 000, 1 000 000 and 10 000 000 iterations. Obviously, in terms of computational resources, one run with 10 000 000 iterations corresponds to 10 runs with 1 000 000 or 100 runs with 100 000 iterations. In order to obtain good estimates of the lower percentiles, 100 runs were performed with the largest number of iterations. Consequently, the number of restarts with different random starting configurations was 1000 and 10 000 for 1 000 000 and 100 000 iterations, respectively. Table 1 summarizes the results obtained for a fixed threshold sequence, which is identical for all runs and numbers of iterations.

When considering these results, it turns out that the trade–off between more restarts with different seeds and a higher number of iterations per restart is in favour of the latter. As expected, both the mean and the lower percentiles become smaller as the number of iterations per run increases while holding the total use of computer resources (number of restart times number of iterations) constant. Given that users are typically interested in the very low percentiles of the distribution, it should be taken into account that the 1%–quantile for the runs with 10 000 000 iterations is estimated based on solely 100 observations. Consequently, this entry has to be interpreted with some care as it is estimated with less precision than other entries of the table.

The above analysis gives valuable information on the dependence of mean and percentiles on the number of iterations and restarts. However, it has not yet answered the practitioner's question whether it is preferable to perform a single run with a very high number of iterations, a few restarts with a moderate number of iterations, or many restarts with a low number of iterations. The outcomes of the experiment can also be used for this purpose. To this end, the results are grouped to 100 artificial sub–experiments each consisting of 100 tries with 100 000 iterations, 10 tries with 1 000 000 iterations and one try with 10 000 000 iterations. Now, only the overall best results obtained for each number of iterations are compared. Of course, considering this non–robust statistic

Table 2: Restart threshold accepting (comparison)

Iterations per try	100 000	1 000 000	10 000 000
Restarts	100	10	1
Times best in 100	0	65	35
Mean deviation from best	73.56	5.16	14.48

is not a valid approach from a pure statistical perspective. However, it corresponds closely to the typical proceeding in real applications. Table 2 shows the number of times, the respective combinations of iterations per restart and number of restarts gave the best result. In the lower part the mean deviation from the best results for the three categories from the best results of these three categories are depicted.

If one has only computer resources for a total of 10 000 000 iterations, these results clearly indicate that one should neither perform many restarts with a very low number of iterations, nor only one huge run. Instead, the best expected performance is given by a choice which falls between the two extremes, i.e., restarting the threshold accepting a few times with a moderate number of iterations. For recent results for a uniform design problem see also Winker (2005).

Summarizing the arguments of this section, two aspects seems noteworthy. Firstly, the common practice to report only the best outcome of several restarts is not adequate. A minimum requirement is to report also the number of restarts and some information on the empirical distribution D_{TA}, e.g., mean and standard deviations or quantiles. Secondly, the results indicate that mean value, standard deviation and low quantiles decrease, other things being equal, with an increasing number of iterations. Nevertheless, the combination of some restarts with a moderate number of iterations seems to be preferable in order to obtain high–quality results.

6 Conclusions

The concept of threshold accepting appears quite simple and yet powerful in its applications. By replacing a stochastic acceptance criterion by a deterministic one, it even reduces the complexity as compared to simulated annealing. Nevertheless, a more detailed presentation and discussion of the central ingredients of the algorithm highlights some aspects relevant for a successful implementation.

TA can be used to tackle highly complex optimization problems which are not accessible by classical optimization algorithms. The cost of implementation are small compared to more refined optimization heuristics, e.g., population

based approaches, or to tailor–made problem specific heuristics. The guidelines provided in this contribution might be helpful for the choice of settings and parameters resulting in a high quality outcome. Then, even if the TA implementation might not provide the global optimum, the best results obtained by this heuristic represent a benchmark which has to be beaten first by any potential challenger.

References

Aarts, E. H. L., and J. K. Lenstra (1997): "Local Search in Combinatorial Optimization: Introduction," in *Local Search in Combinatorial Optimization*, ed. by E. Aarts, and J. K. Lenstra, pp. 1–17, Chichester. Wiley.

Althöfer, I., and K.-U. Koschnik (1991): "On the Convergence of Threshold Accepting," *Applied Mathematics and Optimization*, 24, 183–195.

Barr, R. S., B. L. Golden, J. P. Kelly, M. G. C. Resende, and W. R. Stewart (1995): "Designing and Reporting on Computational Experiments with Heuristic Methods," *Journal of Heuristics*, 1, 9–32.

Brooks, C., S. P. Burke, and G. Persand (2001): "Benchmarks and the Accuracy of GARCH Model Estimation," *International Journal of Forecasting*, 17(1), 45–56.

Chipman, J. S., and P. Winker (2005): "Optimal Aggregation of Linear Time Series Models," *Computational Statistics and Data Analysis*, p. in press.

Dueck, G., and T. Scheuer (1990): "Threshold Accepting: A General Purpose Algorithm Appearing Superior to Simulated Annealing," *Journal of Computational Physics*, 90, 161–175.

Dueck, G., and P. Winker (1992): "New Concepts and Algorithms for Portfolio Choice," *Applied Stochastic Models and Data Analysis*, 8, 159–178.

Dueck, G., and J. Wirsching (1991): "Threshold Accepting Algorithms for 0–1 Knapsack Problems," in *Proceedings of the Fourth European Conference on Mathematics in Industry*, ed. by H. Wacker, and W. Zulehner, pp. 225–262, Stuttgart. B. G. Teubner.

Fang, K.-T., D. K. J. Lin, P. Winker, and Y. Zhang (2000): "Uniform Design: Theory and Application," *Technometrics*, 42, 237–248.

Fang, K.-T., X. 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.

Fang, K.-T., C.-X. Ma, and P. Winker (2002): "Centered L_2–Discrepancy of Random Sampling and Latin Hypercube Design, and Construction of Uniform Designs," *Mathematics of Computation*, 71, 275–296.

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.

Farley, A. M., and S. Jones (1994): "Using a Genetic Algorithm to Determine an Index of Leading Economic Indicators," *Computational Economics*, 7, 163–173.

Ferrall, C. (2004): "Solving Finite Mixture Models: Efficient Computation in Economics under Serial and Parallel Execution," Discussion paper, Queen's University, Canada.

Fox, B. L. (1994): "Random Restarting versus Simulated Annealing," *Computers, Mathematics and Applications*, 27(6), 33–35.

Gilli, M., and E. Këllezi (2002a): "The Threshold Accepting Heuristic for Index Tracking," in *Financial Engineering, E-Commerce, and Supply Chain*, ed. by P. Pardalos, and V. Tsitsiringos, Applied Optimization Series, pp. 1–18. Kluwer.

Gilli, M., and E. Këllezi (2002b): "A Global Optimization Heuristic for Portfolio Choice with VaR and Expected Shortfall," in *Computational Methods in Decision-making, Economics and Finance*, ed. by E. Kontoghiorghes, B. Rustem, and S. Siokos, pp. 165–181. Kluwer.

Gilli, M., and P. Winker (2003): "A Global Optimization Heuristic for Estimating Agent Based Models," *Computational Statistics and Data Analysis*, 42(3), 299–312.

Hamming, R. W. (1950): "Error Detecting and Error Correcting Codes," *Bell System Technical Journal*, 29, 147–160.

Hanafi, S., A. Freville, and A. E. Abdellaoui (1996): "Comparison of Heuristics for the 0–1 Multidimensional Knapsack Problem," in *Meta–Heuristics: Theory and Applications*, ed. by I. H. Osman, and J. P. Kelly, pp. 449–465, Boston, MA. Kluwer.

Judd, K. D. (1998): *Numerical Methods in Economics*. MIT Press, Cambridge, MA.

Kirkpatrick, S., C. Gelatt, and M. Vecchi (1983): "Optimization by Simulated Annealing," *Science*, 220, 671–680.

Maringer, D. (2005): *Portfolio Management with Heuristic Optimization*. Springer, forthcoming.

Osman, I. H., and G. Laporte (1996): "Metaheuristics: A Bibliography," *Annals of Operations Research*, 63, 513–623.

Winker, P. (1995): "Identification of Multivariate AR–Models by Threshold Accepting," *Computational Statistics and Data Analysis*, 20(9), 295–307.

——— (2000): "Optimized Multivariate Lag Structure Selection," *Computational Economics*, 16, 87–103.

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

——— (2005): "The Stochastics of Threshold Accepting: Analysis of an Application to the Uniform Design Problem," Discussion Paper 2005–003E, Staatswissenschaftliche Fakultät, Universität Erfurt.

Winker, P., and K.-T. Fang (1997a): "Application of Threshold Accepting to the Evaluation of the Discrepancy of a Set of Points," *SIAM Journal on Numerical Analysis*, 34, 2028–2042.

——— (1997b): "Optimal U–Type Designs," in *Monte Carlo and Quasi–Monte Carlo Methods 1996*, ed. by H. Niederreiter, P. Hellekalek, G. Larcher, and P. Zinterhof, Lecture Notes in Statistics 127, pp. 436–448, New York. Springer.

Winker, P., and M. Gilli (2004): "Applications of Optimization Heuristics to Estimation and Modelling Problems," *Computational Statistics & Data Analysis*, 47(2), 211–223.