Old Bachelor Acceptance: A New Class of Non-Monotone Threshold Accepting Methods

T. C. Hu†, Andrew B. Kahng and Chung-Wen Albert Tsao

UCLA Dept. of Computer Science, Los Angeles, CA 90024-1596 Tel. 310-206-7073
† UCSD CS&E Dept., La Jolla, CA 92093-0114 Tel. 619-534-3854
Email: hu@cs.ucsd.edu, abk@cs.ucla.edu, tsao@cs.ucla.edu

Abstract

Stochastic hill-climbing algorithms, particularly simulated annealing (SA) and threshold acceptance (TA), have become very popular for global optimization applications. Typical implementations of SA or TA use *monotone* temperature or threshold schedules, and are not formulated to accommodate practical time limits. We present a new threshold acceptance strategy called *Old Bachelor Acceptance* (OBA) which has three distinguishing features: (i) it is specifically motivated by the practical requirement of optimization within a prescribed time bound, (ii) the threshold schedule is *self-tuning*, and (iii) the threshold schedule is *non-monotone*, with threshold values even allowed to become negative. The standard implementation of the TA method of Dueck and Scheuer is a special case of OBA. Experiments using several classes of symmetric traveling salesman problem instances show that OBA can outperform previous hill-climbing methods for time-critical optimizations. A number of directions for future work are suggested.

Given a set S of feasible solutions and a real-valued cost function $f : S \rightarrow \Re$, *global optimization* may without loss of generality be formulated as the search for a global minimizer $s \in S$ such that $f(s) \leq f(s') \; \forall s' \in S$. Typically, $|S|$ is very large compared to the number of solutions that can be examined in practice. For small instances of certain global optimizations, implicit enumeration (e.g., branch-and-bound) or polyhedral approaches can prune the solution space and afford solutions within practical time limits; other problem formulations may be tractable to problem-specific methods. However, many important global optimization formulations (both discrete and continuous) are not only NP-hard [8], but also have no known problem-specific solution methods. Therefore, general-purpose heuristics are of interest. In this paper, we present a new class of stochastic hill-climbing heuristics for global optimizations; we call this new strategy *Old Bachelor Acceptance* (OBA).

Our discussion opens in Section 1 with a characterization of iterative, stochastic hill-climbing heuristics; these are usually superior to greedy methods in that they can probabilistically escape from locally optimal solutions. The leading examples of stochastic hill-climbing algorithms – the *simulated annealing* (SA) approach of Kirkpatrick et al. [25] and Cerny [6], along with the *threshold acceptance* approach of Dueck and Scheuer [7] – have gained wide popularity due to the quality of the

solutions that they return. Nonetheless, we observe that current SA and TA implementations usually have two shortcomings: (i) they are not formulated with respect to a *practical* CPU bound for the optimization, and (ii) they are respectively restricted to *monotone* schedules for their temperature and threshold parameters. In Section 2, we exhaustively determine *optimal*, *finite-time* threshold schedules for small (synthetic) problem instances. These schedules provide the motivation for a *non-monotone* strategy. We then propose our new Old Bachelor Acceptance (OBA) method, which affords a self-tuning, non-monotone approach to (bounded-time) hill-climbing optimization. After a discussion of parameters which can be used to tune the OBA approach, we state two promising variants, which we call OBA1 and OBA2. In Section 3, we present experimental results for our OBA variants over classes of traveling salesman problem (TSP) instances. Our simulations show that OBA can provide significant improvements over the previous TA method of Dueck and Scheuer [7], particularly in the regime of strongly time-bounded global optimization. The paper concludes in Section 4 with a number of directions for future research.

1 Preliminaries: Global Optimization Heuristics

1.1 Iterative Methods

We are interested in heuristics which iteratively apply the following two rules (Figure 1):

Iterative Global Optimization

Rule 1: Given the current solution s_i, generate a new trial solution s'
Rule 2: Decide whether to set $s_{i+1} = s_i$ or $s_{i+1} = s'$.

Figure 1: High-level template for iterative global optimization.

Rule 1 is *memoryless*, with generation of s' based only on the current solution s_i. Note that a "history-dependent" Rule 1′ might be, for example: Given the *history* of solutions evaluated thus far, generate a new trial solution s'. Such a Rule 1′ accommodates such methods as iterated descent [3] and tabu search [10], which more systematically exploit information about the recent history of solutions, e.g., whether the current solution has been previously visited, whether the current solution is known to be a local optimum, etc. (see also the heuristic search techniques used in artificial intelligence [29]). These latter methods are beyond the scope of the present discussion. Rule 1 also induces the notion of a *neighborhood structure* over S, where the neighborhood $N(s_i)$ of the current solution $s_i \in S$ is the set of possible trial solutions s' that can be generated from s_i. The

quality of solutions defines a *cost surface* over the neighborhood structure, and optimization is search for a global minimum in this cost surface. Typically, the set $N(s)$ consists of slight perturbations of the current solution s, for example, via the 2-interchange operator for the traveling salesman problem [18] or the pair-swap operator for graph bisection [17]. When the size of $N(s)$ is constant for all $s \in S$, we denote the neighborhood size by $|N|$. In practice, Rule 1 simply picks a random $s' \in N(s_i)$ from within "obvious" neighborhood structures such as those noted for the TSP and graph bisection problems [18]. Therefore, it is Rule 2 which determines the nature of an optimization heuristic as it traverses the cost surface.

A simple instance of Rule 2 is, "Replace s_i by s' if $f(s') < f(s_i)$," which corresponds to greedy optimization. Greed has been widely employed because of its simplicity and its acceptable success in a variety of implementations, e.g., Johnson et al. [16] [17] have documented the utility of greed for several hard combinatorial problems. However, the performance of greedy methods is erratic, and achieving "stable" – i.e., predictable – performance requires multiple random initial starting solutions. Johnson et al. [16] have determined that several thousand initial random starting configurations are necessary for greed to afford stable solution quality for graph bisection instances of size $n = 500$; this number grows rapidly with n and becomes hopeless for instance sizes of, e.g., $n = 100,000$ which arise in arenas such as VLSI circuit partitioning. Moreover, central limit phenomena in the cost surface [3] imply that as problems grow large, random local minima are almost surely of "average" quality, so that simple "multi-start" heuristics [31] fail. For details on this subject, the reader is referred to discussions by Baum [3] and Kirkpatrick and Toulouse [26] on traveling salesman structures; by Kauffman and Levin [23] on evolutionary optimization for "adaptive landscapes"; and by Bui et al. [5] and Hagen and Kahng [11]) for graph partitioning. In view of these factors, global optimization heuristics must escape from local minima to adequately explore the solution space of large problems.

1.2 Stochastic Hill-Climbing: SA and TA

Stochastic hill-climbing methods escape from local minima in the cost surface by probabilistically accepting disimprovements, or "uphill moves". The first such method, *simulated annealing* (SA), was proposed independently by Kirkpatrick et al. [25] and Cerny [6] and is motivated by analogies between the solution space of an optimization instance and microstates of a statistical thermodynamical ensemble. Figure 2 summarizes the SA algorithm, which uses the following criteria for Rule 2. If $f(s') < f(s_i)$, then $s_{i+1} = s'$, i.e., the new solution is adopted. If $f(s') \geq f(s)$, the "hill-climbing" disimprovement to $s_{i+1} = s'$ still has a nonzero probability of being adopted, determined by both

the magnitude of the disimprovement and the current value of a *temperature* parameter T_i. This probability is given by the "Boltzmann acceptance" criterion in Line 6 of Figure 2.

Algorithm SA(M)

$M \equiv$ limit on number of Rule 1 iterations

1. Choose (random) initial solution s_0;
2. Choose initial *temperature* T_0;
3. for $i = 0$ to $M - 1$
4. Choose (random) neighbor solution $s' \in N(s_i)$;
5. if $f(s') < f(s_i)$ then $s_{i+1} = s'$
6. else $s_{i+1} = s'$ with $Pr = exp((f(s_i) - f(s'))/T_i)$;
7. $T_{i+1} = next(T_i)$;
8. Return s_i, $0 \leq i \leq M$, such that $f(s_i)$ is minimum.

Figure 2: Bounded-time SA template.

Another stochastic hill-climbing heuristic called *threshold acceptance* (TA), which uses a different Rule 2 criterion (Figure 3), has recently been proposed by Dueck and Scheuer [7]. TA relies on a *threshold*, T_i, which defines the maximum disimprovement $f(s') - f(s_i)$ that is acceptable at the current iteration. All disimprovements greater than T_i are rejected, while all that are less than T_i are accepted. Thus, in contrast to the Boltzmann acceptance rule of annealing, TA offers a deterministic Rule 2.

Algorithm TA(M)

$M \equiv$ limit on number of Rule 1 iterations

1. Choose (random) initial solution s_0;
2. Choose initial *threshold* T_0;
3. for $i = 0$ to $M - 1$
4. Choose random neighbor solution $s' \in N(s_i)$;
5. if $f(s') < f(s_i) + T_i$ then $s_{i+1} = s'$
6. else $s_{i+1} = s_i$
7. $T_{i+1} = next(T_i)$;
8. Return s_i, $0 \leq i \leq M$, such that $f(s_i)$ is minimum.

Figure 3: Bounded-time TA template.

At timestep i, the SA temperature T_i allows hill-climbing by establishing a nonzero probability of accepting a disimprovement, while the TA threshold T_i allows hill-climbing by specifying a permissible amount of disimprovement. Typical SA practice uses a large initial temperature and a final temperature of zero. [Note that $T = \infty$ accepts all moves (i.e., a random walk in the cost surface);

$T = 0$ accepts only improving moves (i.e., greed).] The monotone decrease in T_i is accomplished by $next(T_i)$, which is a heuristic function of the T_i value and the number of iterations since the last cost function improvement. (Typically, $next(T_i)$ tries to achieve "thermodynamic equilibrium" at each temperature value.) Similarly, implementations of TA [7] begin with a large initial threshold T_0 which decreases monotonically to $T_M = 0$. Note that both SA and TA will in practice return the "best-so-far" (BSF) solution, i.e., the minimum-cost solution among $s_0, s_1, \ldots, s_M$; this is reflected in line 8 of each template in Figures 2 and 3, as well as in the experimental comparisons of Section 3 below.

The SA and TA algorithms both enjoy certain theoretical attractions. By using Markov chain arguments and basic aspects of Gibbs-Boltzmann statistics one can show for SA that with an appropriate $next(T_i)$ function, $Pr(s_M \in R) \rightarrow 1$ as $M \rightarrow \infty$, where R denotes the set of all global minimum solutions. In other words, SA is optimal in the limit of infinite time [1]. Althofer and Koschnick [2] argue that each execution of SA lies in some sense within the convex hull of a set of TA executions, and that TA is therefore also provably good. However, this convergence result is slightly weaker than those established for SA.

Finally, the practical utility of stochastic hill-climbing is well-documented, with the SA algorithm now being one of the most widely used heuristics for global optimization [1]. Thus, it is noteworthy that Dueck and Scheuer [7] claim that their TA method "yields better results than SA" with respect to both CPU time and the number of "new state choice steps" (i.e., applications of Rule 1), a standard measure of runtime complexity. Experimental results are presented in [7] which support this claim. A further practical advantage of TA is its greater simplicity of Rule 2, with no exponentiation or random number generation being required. With this in mind, our experimental results below compare OBA variants against the TA algorithm.

2 Non-Monotone Threshold Schedules: The OBA Approach

2.1 Motivations

Recall from the above discussion that SA and TA are traditionally implemented with *monotone* temperature or threshold schedules. For SA, the thermodynamic analogy, as well as the convergence proof based on Gibbs-Boltzmann statistics, together motivate the following intuition [12]: monotone temperature schedules allow annealing to explore "large features" of the cost surface at high T, and then perform finer optimization at lower T. For TA, the authors of [7] state that the "trivial" threshold schedule (linear in i, with $T_i = T_0 * (1 - \frac{i}{M})$) is "essentially best", and suggest that the

performance of TA is basically insensitive to the threshold schedule. Indeed, the successful results reported in [7] were obtained using monotone threshold schedules. However, despite the tremendous success of both SA and TA, certain observations motivate the study of alternative hill-climbing strategies.

First, standard implementations of SA and TA are not amenable to *a priori* specification of CPU limits. With respect to the templates of Figures 2 and 3, common practice will use $M = \infty$ and test for a stopping criterion (e.g., "equilibration" in SA) to terminate the algorithm. A finite time limit M will obviate the theoretical convergence results, and also reflect practical requirements for optimization. Experimental results [21] [19] for large discrete and continuous global optimizations show that optimal annealing schedules vary strongly with the time limit M, but it is not clear how $next(T_i)$ should be defined to accommodate finite M. [The results in [21] and [19] are with respect to single-temperature annealing schedules, which were used in order to reduce the number of degrees of freedom in the experiment. Recent work by Boese, Kahng and Tsao [4] has confirmed the dependence on CPU limit of general annealing schedules; the reader is also referred to the work of Strenski and Kirkpatrick [33] and Althofer and Koschnick [2], which we discuss later in this section.]

Second, current SA and TA implementations are "blind" to the specific features of the cost surface in any given optimization instance. Previous work [21] [19] [20] [32] has shown that large, real-world cost surfaces exhibit strong fits to models of self-similar random structure (e.g., VLSI placement problems have hierarchical scaling properties which resemble high-dimensional fractional Brownian motions [32]). The parameters of such fitted models vary with the individual problem instances, and again, evidence suggests that optimal hill-climbing schedules should be tuned to these parameters [20].

These two observations prompt a variety of questions and simple experiments. Consider the BSF performance of TA from random starting solutions in the one-dimensional cost surface of Figure 4 (six solutions s_i, each with two neighbors). We have exhaustively enumerated threshold schedules for $M = 2, 3, \ldots, 9$ for the cost surface of Figure 4. The value of the threshold will be the cost difference between any two neighboring states in Figure 4, i.e., ± 1 and ± 3. The optimal schedules are defined as the ones that maximize the probability of finding the optimal solution D within prescribed time bounds (starting from a random initial solution). Among all the optimal schedules thus found, many are non-monotone, and some even contain negative threshold values. Some examples are: $\{-1, X\}$, and $\{0, X\}$ for $M = 2$ ($Pr = 0.5000$), $\{-1, 0, X\}$ for $M = 3$ ($Pr = 0.5833$), $\{0, 3, -1, X\}$ for $M = 4$ ($Pr = 0.6250$); $\{0, -3, 3, 1, X\}$ for $M = 5$ ($Pr = 0.6875$); $\{0, 0, 3, 0, 0, X\}$ for $M = 6$ ($Pr = 0.7396$); $\{-1, -3, -3, 3, 1, -1, X\}$ for $M = 7$ ($Pr = 0.7656$); $\{0, 3, 0, 0, 3, 0, 0, X\}$ for $M = 8$ ($Pr = 0.8047$);

and $\{0, -3, 3, -1, -3, 3, 3, -1, X\}$ for $M = 9$ ($Pr = 0.8372$). Observe that the last value T_{M-1} (denoted as X) in the threshold schedule does not affect the best-so-far solution value at time M, as long as $X \geq -3$.

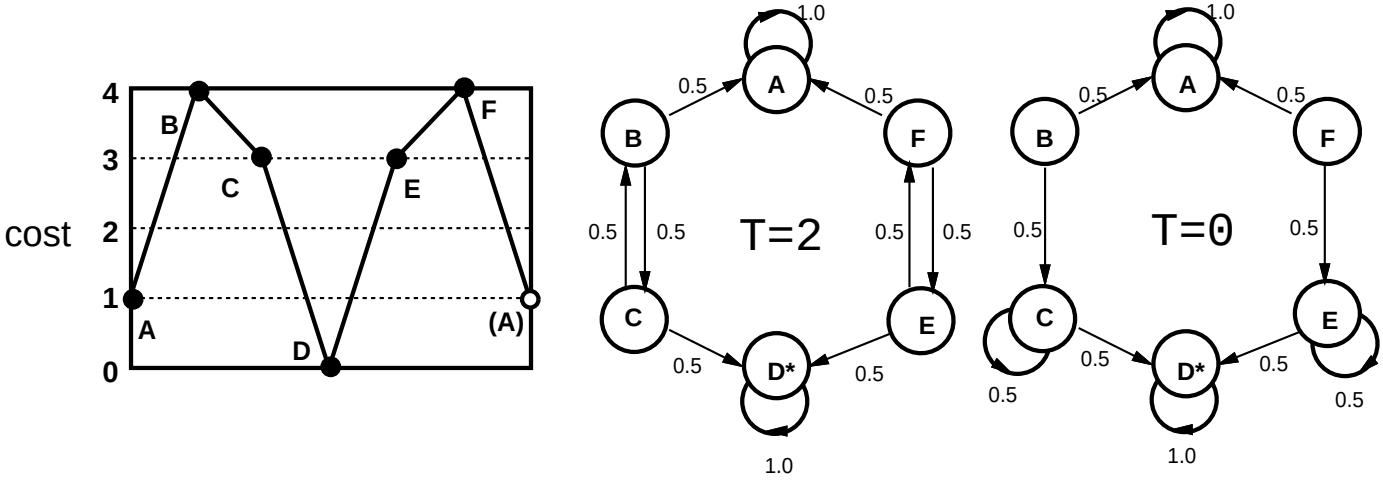

Figure 4: A simple cost surface, along with transition probabilities for $T = 2$ and $T = 0$.

A number of other authors have also touched on the issue of non-monotonicity in annealing. In particular, Glover [9] suggests in a very general way that some sort of nonmonotonicity would improve the performance of heuristic search procedures. Very recently, Osman [28] has reported very effective simulated annealing approaches with non-monotone cooling schedule; these significantly outperform other SA implementations that use cooling schedules. Strenski and Kirkpatrick [33] have shown that "locally optimal" annealing schedules can be non-monotone for a small instance of the graph bisection problem that is highly structured to reduce the size of the solution space. Hajek and Sasaki [13] show that a class of cost surfaces exists for which optimal schedules are non-monotone. Finally, Althofer and Koschnick [2] enumerate optimal TA schedules for a small cost surface and find clear evidence (Table 4.1 in [2]) of non-monotonicity; however, the authors surprisingly make no comment on this data. Our own investigations [4] support these previous studies. Moreover, we have found that the non-monotonicity of the known optimal schedules does not seem to be an artifact of the small size of $|S|$ relative to the available time M in these studies. Given these motivations, we have investigated a class of threshold accepting methods which use non-monotone threshold sequences.

2.2 The OBA Algorithm

Old Bachelor Acceptance uses a threshold criterion in Rule 2, but the threshold changes dynamically – up or down – based on the perceived likelihood of being near a local minimum. Observe that if the current solution s_i has lower cost than most of its neighbors, it will be hard to move to a neighboring solution; in such a situation, standard TA will repeatedly generate a trial solution s' and fail to accept it. OBA uses a principle of "dwindling expectations": after each failure, the criterion for "acceptability" is relaxed by slightly increasing the threshold T_i (this motivates the name "Old Bachelor Acceptance"). After sufficiently many consecutive failures, the threshold will become large enough for OBA to escape the current local minimum. The converse of "dwindling expectations" is what we call *ambition*, whereby after each acceptance of s', the threshold is *lowered* so that OBA becomes more aggressive in moving toward a local minimum. With this in mind, the basic OBA template is as shown in Figure 5.

Algorithm OBA(M)

$M \equiv$ limit on number of Rule 1 iterations

1. Choose (random) initial solution s_0;
2. Choose initial threshold T_0;
3. for $i = 0$ to $M - 1$ do
4. Choose (random) neighbor solution $s' \in N(s_i)$;
5. if $f(s') < f(s_i) + T_i$ then
6. $s_{i+1} = s'$;
7. $T_{i+1} = T_i - decr(T_i)$;
8. else
9. $s_{i+1} = s_i$;
10. $T_{i+1} = T_i + incr(T_i)$;
11. endif.
12. Return s_i, $0 \leq i \leq M$, such that $f(s_i)$ is minimum.

Figure 5: High-level OBA description.

Notice that if we use update functions $decr(T_i) = -incr(T_i)$, then OBA can be made equivalent to the TA method of Dueck and Scheuer (e.g., using the constant functions $decr(T_i) = -incr(T_i) = \frac{T_0}{M}$ yields the "trivial" threshold schedule recommended in [7]). Thus, special cases of OBA will enjoy the same convergence properties shown for TA in [2].

2.3 OBA Variants

Via the threshold update functions $incr(T_i)$ and $decr(T_i)$, the template of Figure 5 captures many possible strategies. We have typically based *decr* and *incr* on the following factors:

1. The *neighborhood size*, $|N|$, along with the *age* of the current iteration, which is the number of Rule 1 applications since the last move acceptance. The value of $|N|$ affects "reachability" between solutions, i.e., the diameter of and multiplicity of paths within the neighborhood structure. Intuitively, *age* reflects the OBA algorithm's current perception of local structure in the cost surface: increasing *age* implies greater likelihood that s_i is a local minimum, and that the threshold should increase faster.

2. The amount of time remaining, $M - i$. Since previous work [19] [20] has observed strong dependence of optimal hill-climbing strategies on the time bound M, we may allow *decr* and *incr* to depend on the proportion of time used, i/M.

3. The current threshold value T_i. We may allow different update rules depending on whether T_i is highly positive, highly negative, close to zero, etc.

Our initial experiments involved OBA variants that used the "obvious" choice of $decr(T_i) = \Delta_1$ and $incr(T_i) = \Delta_2$ both being constant functions, since TA is included among such variants. In Section 3 below we give experimental results for two slightly more sophisticated strategies, which we call OBA1 and OBA2. While OBA1 and OBA2 are only two out of a vast range of possible OBA variants, their empirical performance illustrates both the generality of the OBA approach as well as its practical utility.

OBA1. The OBA1 variant (see Figure 6) is highly tunable via the parameters Δ, a, b, and c. The algorithm has a core threshold update strategy of form

$$T_{i+1} = ((\frac{age}{a})^b - 1) * (\Delta) * (1 - \frac{i}{M})^c.$$

Whenever $age = 0$, OBA1 immediately sets the threshold to the most negative value allowable (i.e., as low as $-\Delta$ in line 8), thus giving the algorithm the "ambition" to improve rapidly. The threshold then rises from this negative value until the next move acceptance occurs. For $age > 0$, the update rule allows the OBA1 threshold growth rate to increase with *age*. More specifically, the parameters a, b and c afford the ability to fine-tune the growth rate $incr(T_i)$ as follows:

- a changes the threshold growth rate by a multiplicative factor;
- b allows a power-law growth rate; and
- c tunes a heuristic "damping" factor $(1 - \frac{i}{M})$ which is used to scale the magnitude of T_i as $i \rightarrow M$. [Here, the damping factor does not have any thermodynamic motivations as

Algorithm variant OBA1(M,Δ,a,b,c)

M $\equiv$ limit on number of Rule 1 iterations
Δ $\equiv$ threshold update granularity
a $\equiv$ multiplicative factor in growth rate
b $\equiv$ power-law in growth rate
c $\equiv$ coefficient in damping of threshold magnitude

1. $T_0 = 0$;
2. $age = 0$;
3. Choose (random) initial solution s_0;
4. for $i = 0$ to $M - 1$ do
5. Choose random neighbor solution $s' \in N(s_i)$;
6. if $f(s') < f(s_i) + T_i$ then
7. $s_{i+1} = s'$;
8. $age = 0$;
9. else
10. $s_{i+1} = s_i$;
11. $age = age + 1$
12. endif
13. $T_{i+1} = ((\frac{age}{a})^b - 1) * (\Delta) * (1 - \frac{i}{M})^c$;
14. endfor.
15. Return s_i, $0 \leq i \leq M$, such that $f(s_i)$ is minimum.

Figure 6: OBA1 variant, which incorporates fine tuning of *incr* threshold update function, along with a maximally ambitious *decr* threshold update strategy.

with annealing schedules that go to zero, but rather is intended to ensure that at some point during the optimization, the appropriate "granularity" in the threshold update is applied. The parameter c can be useful in determining the stage of the optimization at which a given granularity is applied.]

In this way, OBA1 can capture the gamut of strategies from "steep descent, mild ascent" to "steep descent, steep ascent" (cf. the terminology of Hansen and Jaumard [14]). Two important points should be noted. First, OBA1 demonstrates that the new threshold value T_{i+1} does not have to be a function of the previous T_i; here, T_{i+1} is completely determined by age, the current timestep i, and the parameterization of the algorithm. Second, OBA1 follows the intuition provided by the example of Figure 4, allowing threshold values to become negative so that the algorithm may prefer a *good* improving move over a random improving move. In our experiments below involving the traveling salesman problem, we use $a = 1 \cdot n$ (i.e., scaled to the size n of the problem instance), $b = 2$, and $c = 0.5$.

OBA2. The OBA2 variant (Figure 7) recalls the "steepest descent, mildest ascent" strategy proposed by Hansen and Jaumard described in [14]. In contrast to OBA1, the new threshold T_{i+1} in OBA2 depends on the previous value T_i. OBA2 uses a decrement $decr(T_i)$ whose magnitude grows quadratically, affording greater ambition; OBA2 also uses a linear $incr(T_i)$ function so that expectations do not "dwindle" too rapidly. In lines 9-10 of the algorithm template, we use the algorithm parameter d to establish the criterion for an *easy* move acceptance, namely, an acceptance $s_{i+1} = s'$ is easy if fewer than d move generations have elapsed since the last acceptance of s'. In practice, we believe that the choice of d should be dependent on the neighborhood size. In our experiments below, we use $d = \sqrt{2 * |N|}$, which happens to be the number of cities in the TSP instance. (Intuitively, d sets the threshold for consecutive failures at which the algorithm begins to assume that it is at a local minimum. Note also that the dependence of d on problem size is similar to the dependence of the OBA1 parameter a on problem size.)

Algorithm variant OBA2(M,Δ,d)

M $\equiv$ limit on number of Rule 1 iterations
Δ $\equiv$ threshold update granularity
d $\equiv$ "easy" move threshold
$count$ $\equiv$ number of consecutive "easy" move acceptances

1. $T_0 = 0$;
2. $count = 1$; $prev\_age = M$ (large initial value);
3. Choose (random) initial solution s_0;
4. for $i = 0$ to $M - 1$ do
5. Choose random neighbor solution $s' \in N(s_i)$;
6. if $f(s') < f(s_i) + T_i$ then
7. $s_{i+1} = s'$;
8. $age = 0$;
9. if $prev\_age < d$ then
10. $count = count + 1$;
11. else
12. $count = 1$;
14. endif
13. $T_{i+1} = T_i - count * \Delta * (1 - \frac{i}{M})$;
15. else
16. $s_{i+1} = s_i$;
17. $age = age + 1$;
18. $T_{i+1} = T_i + \frac{\Delta}{d} * (1 - \frac{i}{M})$;
19. endif
20. $prev\_age = age$.
21. endfor.

Figure 7: OBA2 variant, incorporating steeper descent and milder ascent, along with criterion for consecutive "easy" move acceptances.

Finally, we also report results for two further variants, which we call OBA1_N and OBA2_N. The OBA1_N (OBA2_N) result is obtained by executing the OBA1 (OBA2) algorithm, with the only difference being that the Rule 2 acceptance criterion treats negative threshold values $T_i < 0$ as if they are equal to zero. Our goal here is to obtain some indication as to whether "ambition" is practically useful, the example of Figure 4 notwithstanding.

We close this section with a qualitative portrayal of the differences in the threshold schedules produced by our four OBA variants. Each of the four algorithms was executed with $M = 400,000$ on a single 50-city Euclidean planar TSP instance with random city locations. Figure 8 shows in detail the 500-step threshold subsequence from $i = 100,000$ to $i = 100,500$ in each of these runs. The threshold sequences are superimposed against the linearly decreasing TA threshold sequence. Note that since the Figure shows only a very small portion of the total OBA execution, the linearly decreasing TA threshold sequence appears to be constant.

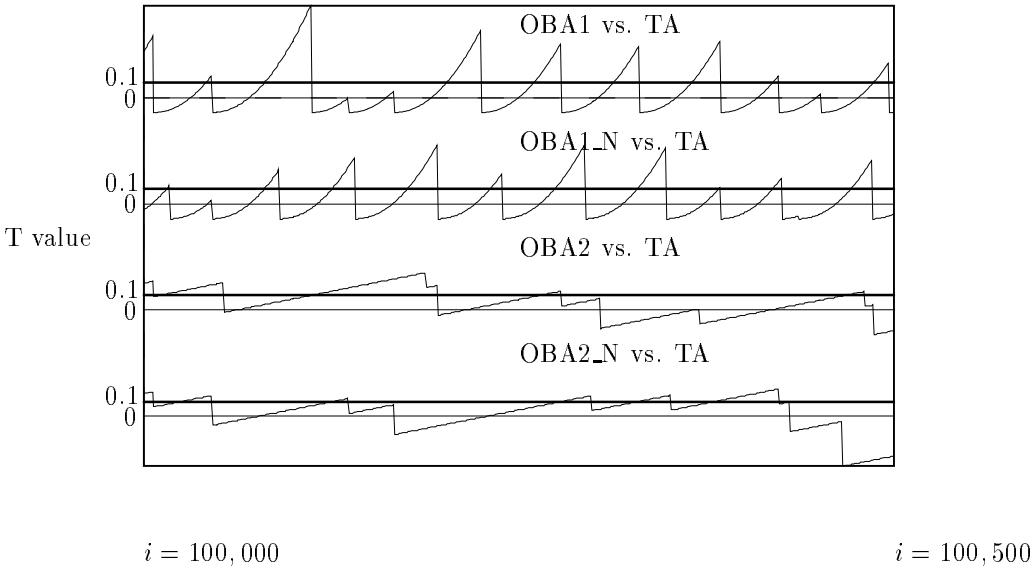

Figure 8: Detailed 500-step intervals, $i = 100,000$ to $i = 100,500$, taken from runs with $M = 400,000$. The non-monotone nature of the OBA threshold sequences clearly contrasts with the linearly decreasing TA sequence shown by the darkened solid line. Specific parameter values: $M = 0.4 \times 10^6$, $\Delta = 0.093$, $a = 1$, $b = 2$, $c = 0.5$ and $d = 50$.

3 Experimental Results

We tested OBA1, OBA1_N, OBA2, OBA2_N and TA on instances of the traveling salesman problem (TSP). The TSP is a well-studied NP-hard problem as well as a historically ubiquitous testbed for both SA and TA. Our experimental protocol was as follows:

1. Two classes of TSP instances were considered: (i) *Euclidean* planar instances corresponding to random pointsets drawn from a uniform distribution in the Euclidean unit square; and (ii) *random* instances having symmetric distance matrices with each intercity distance drawn from a uniform distribution in $[0, 1]$. These are the two most commonly treated classes of TSP, and in some sense represent limiting cases with respect to "metricity" of the TSP instance. (We also studied a class of *hierarchical* TSP instances, which reflect a clustered, non-uniform distribution of points in the Euclidean plane. Results for these instances were qualitatively similar to those for Euclidean instances which had n (the number of cities in the TSP instance) equal to the number of clusters in the hierarchical instances, so we do not report them here.)

2. Instance sizes ranged from $n = 50$ to $n = 200$; for these sizes, we considered CPU limits of between $M = 0.4 \times 10^6$ and $M = 1.0 \times 10^6$ applications of Rule 1, closely following the studies of Rossier et al. [30]. Since move generations and cost function evaluations dominate the runtimes of stochastic hill-climbing algorithms, the actual running time for OBA, SA, TA, etc. for any given value of M will be essentially identical. For all OBA1 runs, we used $\Delta = 0.093$, $a = 1 \cdot n$, $b = 2$ and $c = 0.5$. For all OBA2 runs, we used $\Delta = 0.093$ and $d = n$. (In our initial studies, the value of Δ was empirically set equal to the cost of the minimum spanning tree over the pointset divided by the size of the pointset. Intuitively, this is an appropriate granularity for the TSP optimization. The same Δ value was also used as the initial temperature of the TA. Here, we fix Δ at a representative value.)

3. Our Rule 1 corresponds to the popular Lin 2-opt neighborhood structure [24], wherein a neighbor solution s' is generated by deleting a random pair of edges in s_i and then reconnecting the two paths to achieve the "other" possible tour. The size of the neighborhood structure is $|N| = \frac{n(n-1)}{2} - n$ (all pairs of edges, except for adjacent pairs, which yield no change in the tour).

4. We report the best solution quality encountered during the execution of the algorithm, i.e., the minimum value among $f(s_0), f(s_1), \ldots, f(s_M)$, as indicated by the last line of the Figure 5 template.

Tables I and II compare our four OBA variants against TA for Euclidean and random instances of size $n = 50$ ($M = 0.4 \times 10^6$), $n = 100$ ($M = 0.7 \times 10^6$) and $n = 200$ ($M = 1.0 \times 10^6$). These are exactly the same time bounds used by Rossier et al. ([30], p. 162, Table 4) in their studies of the SA algorithm. In the tables, we measure the relative performance of the algorithms versus TA at intervals of $M/5$ move generations. We use $f(x)$ to denote the BSF solution quality at step x, normalized to the BSF TA solution quality. Thus, $f(x) \equiv 1.000$ for the TA algorithm. Each of our results represents a geometric average of single runs for each of 100 randomly generated instances. (Since the score of each run is a *ratio* of OBA performance to TA performance, we believe that using a geometric average is more intuitive. For instance, given two scores 2 and 0.5, the geometric average gives the more reasonable value of 1 while the arithmetic average returns the value 1.25. Figure 9 shows that the choice of averaging scheme does not materially affect the conclusions drawn from our experiments. Of course, for any given instance and any given execution, it is possible for OBA performance to fluctuate due to random seeds, etc.)

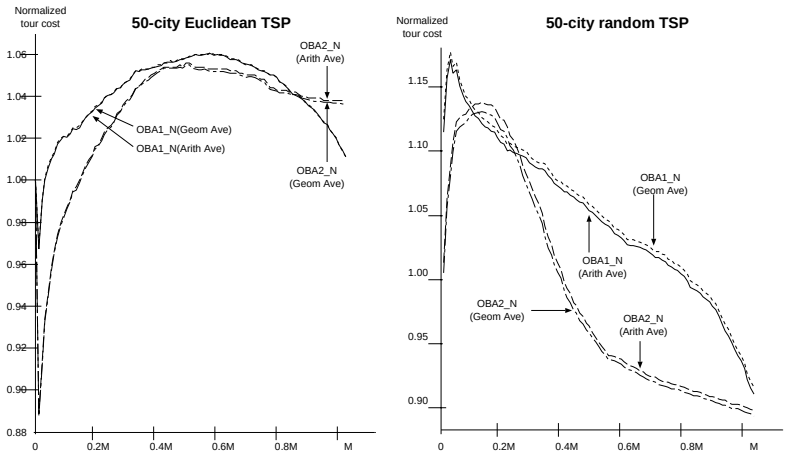

Figure 9: Example results showing that the choice of geometric over arithmetic averaging does not affect the conclusions drawn from experimental data.

In most cases, the OBA variants find significantly better solutions than TA within the early stages of the optimization; this is consistent with the original finite-time motivations for the OBA strategy. In addition, we note that the OBA2 variant seems particularly promising: it uniformly outperforms TA on the random instances and is very competitive with, if not better than, TA on the Euclidean instances. Finally, the OBAx_N variants are noticeably better than the corresponding OBAx variants; indeed, negative threshold values resulted in worse performance for all of the OBA variants that we tested [22]. This suggests that negative threshold values may not be useful in

practical optimization, despite the optimality of schedules that use negative T_i for such examples as Figure 4.

$n = 50$, $M = 5i = 0.4 \times 10^6$

Algorithm	$f(s_i)$	$f(s_{2i})$	$f(s_{3i})$	$f(s_{4i})$	$f(s_{5i})$
OBA1	1.002	1.016	1.017	1.007	1.004
OBA1_N	0.962	0.987	0.999	1.000	0.998
OBA2	0.971	0.995	1.004	1.004	1.002
OBA2_N*	0.958	0.986	0.999	1.001	1.000

$n = 100$, $M = 5i = 0.7 \times 10^6$

OBA1	1.001	1.025	1.033	1.029	1.023
OBA1_N	0.947	0.991	1.012	1.017	1.017
OBA2	0.968	1.003	1.016	1.018	1.016
OBA2_N*	0.939	0.986	1.006	1.013	1.012

$n = 200$, $M = 5i = 1.0 \times 10^6$

OBA1	1.010	1.044	1.060	1.055	1.048
OBA1_N	0.942	0.997	1.033	1.044	1.047
OBA2	1.103	1.017	1.032	1.036	1.030
OBA2_N*	0.931	0.981	1.014	1.025	1.026

Table I: Euclidean TSP results for $n = 50, 100, 200$ and respective time bounds $M = 5i = 0.4 \times 10^6$, 0.7×10^6, 1.0×10^6, normalized to TA solution quality (average taken over 100 randomly generated instances).

Finally, Figures 10 and 11 give a more detailed portrayal of how the OBA1 and OBA2 algorithms progress, in comparison with the TA algorithm. Again, we use the geometric average of performance ratio, averaged at each time step. Figure 10 portrays the four OBA variants over 100 Euclidean instances and one run per instance, using $n = 50$; the four variants are all within 0.4% of TA after $M = 400,000$ steps, but can also be significantly better than TA in the earlier stages of the optimization. Figure 11 similarly portrays the same OBA variants over 100 random instances with $n = 50$; the variants return results that are on average from 11.1% to 13.3% better than those of TA.

4 Conclusions

Our experimental results indicate that non-monotone threshold schedules are promising within hill-climbing approaches to global optimization. We believe that the OBA paradigm provides a powerful and general template for exploration of such non-monotone heuristics. Indeed, a number of previous strategies, including multi-start [31], steepest ascent-descent [27], and even the TA algorithm [7], may all be captured within the unifying OBA paradigm.

$n = 50$, $M = 5i = 0.4 \times 10^6$

Algorithm	$f(s_i)$	$f(s_{2i})$	$f(s_{3i})$	$f(s_{4i})$	$f(s_{5i})$
OBA1	0.966	0.928	0.908	0.893	0.889
OBA1_N	0.941	0.915	0.899	0.893	0.887
OBA2	0.925	0.895	0.884	0.876	0.869
OBA2_N*	0.914	0.889	0.879	0.871	0.867

$n = 100$, $M = 5i = 0.7 \times 10^6$

OBA1	1.055	0.998	0.967	0.952	0.945
OBA1_N	1.011	0.971	0.949	0.939	0.935
OBA2	0.985	0.943	0.926	0.920	0.914
OBA2_N*	0.978	0.937	0.922	0.916	0.911

$n = 200$, $M = 5i = 1.0 \times 10^6$

OBA1	1.066	1.075	1.055	1.041	1.030
OBA1_N	1.026	1.034	1.023	1.014	1.009
OBA2	1.131	1.031	1.005	0.983	0.972
OBA2_N*	1.035	1.017	0.990	0.976	0.968

Table II: Random TSP results for $n = 50, 100, 200$ and respective time bounds $M = 5i = 0.4 \times 10^6$, 0.7×10^6, 1.0×10^6, normalized to TA solution quality (average taken over 100 randomly generated instances).

The OBA variants that we have presented (particularly OBA2_N) perform well on both random and Euclidean TSP instances, and this good performance is furthermore achieved with respect to a *prescribed* time bound that is an parameter of OBA. Since random and Euclidean instances are extremal with respect to "geometricness" or "metricity" of symmetric TSP's, we surmise that the OBA variants we report here are fairly robust. We note that our broader experiments have also confirmed the strength of Dueck and Scheuer's original TA method: for example, our implementations of OBA variants corresponding to the steepest ascent-descent [27] and iterated descent [3] approaches exhibited noticeably poorer performance than TA [22]. This is an indication of the sensitivity of multi-start and iterated greedy approaches with respect to input parameters. In this sense, the *self-tuning* capabilities of the OBA approach seem quite valuable. (On the other hand, certain multi-start and iterated descent methods can be only approximately captured within the OBA template, since the requirement of a memoryless Rule 1 (i.e., move generation) makes checking local minima inefficient and/or inaccurate. The poorer performance of OBA variants which emulate such strategies should be assessed with this in mind.)

A number of research directions seem promising. (i) We would like to extend the non-monotone OBA approach to simulated annealing (using temperature T instead of threshold T). (ii) We hope

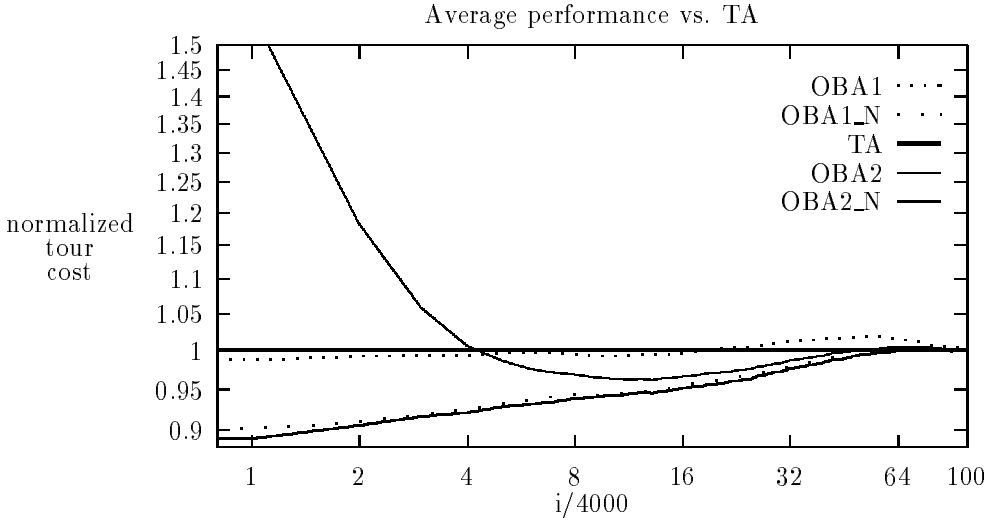

Figure 10: Ratios of OBA solution quality to TA solution quality, geometrically averaged at each time step over 100 Euclidean planar TSP instances. In the figure, data is sampled at every 4000 time steps, yielding 100 data points for $M = 400,000$. Note that the plot is on a log-log scale.

to confirm the robustness of our OBA variants on larger instances and on other problem classes, e.g., asymmetric TSPs and a wider variety of combinatorial optimizations. A related goal would be to derive natural relationships between the proper parameterization of OBA algorithm variants, the available CPU limit M, and the size n of the problem instance. A more careful investigation of the OBA2_N strategy seems warranted since OBA2_N is clearly the best variant that we report here; recall that OBA2_N is suggestive of the "steepest descent, mildest ascent" strategy proposed by Hansen and Jaumard [14] (see also the discussions in [3] [15]). (iii) Recall from above that OBAx_N will always dominate the OBAx variant on "real" cost surfaces, the example of Figure 4 notwithstanding. We hope to show that this dominance always holds within, e.g., the structural models of cost surfaces proposed in [32] [21]. (iv) We would like to extend the hill-climbing optimization template to include non-degenerate history (i.e., memory) in the Rule 1 generation of s'. Augmenting our existing OBA template in this way would provide a very general taxonomy of available hill-climbing approaches. (v) Finally, the most far-reaching outgrowth of this research may eventually stem from the motivating studies of Section 2.1, and specifically, from such studies of "BSF-optimal" schedules as those made with respect to Figure 4. Our preliminary work [4] indicates that such studies may lead to additional justifications for the non-monotone approach to stochastic hill-climbing.

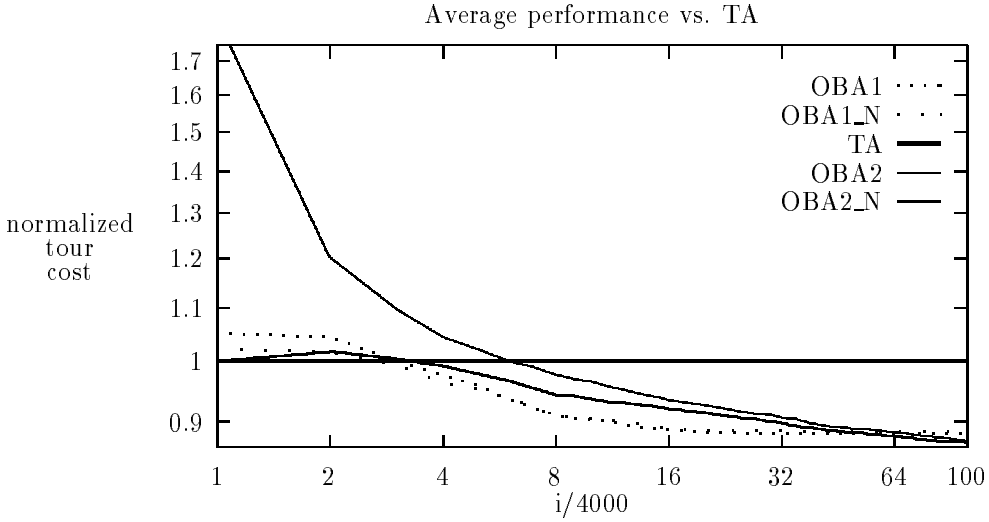

Figure 11: Ratios of OBA solution quality to TA solution quality, geometrically averaged at each time step over 100 random TSP instances. In the figure, data is sampled at every 4000 time steps, yielding 100 data points for $M = 400,000$. Note that the plot is on a log-log scale.

5 Acknowledgments

Partial support for this research was provided by NSF MIP-9110696, NSF Young Investigator Award MIP-9257982, ARO DAAK-70-92-K-0001 and ARO DAAL-03-92-G-0050. We thank the anonymous reviewers for their detailed and constructive comments.

References

[1] E. H. L. Aarts and J. Korst, 1989. *Simulated Annealing and Boltzmann Machines: a Stochastic Approach to Combinatorial Optimization and Neural Computing*, Wiley, New York.

[2] I. Althofer and K. U. Koschnick, 1991. On the Convergence of Threshold Accepting, *Applied Mathematics and Optimization* 24, 183-195.

[3] E. B. Baum, 1986. Iterated Descent: a Better Algorithm for Local Search in Combinatorial Optimization Problems, *Technical Report 164-30*, Crellin Laboratory, California Institute of Technology, Pasadena, CA 91125.

[4] K. D. Boese, A. B. Kahng and C.-W. A. Tsao, 1993. Best-So-Far vs. Where-You-Are: New Perspectives on Simulated Annealing for CAD, *Proc. European Design Automation Conference*, 78-83.

[5] T. N. Bui, S. Chauduri, F. T. Leighton and M. Sipser, 1987. Graph Bisection Algorithms with Good Average Case Behavior, *Combinatorica* 7(2), 171-191.

[6] V. Cerny, 1985. Thermodynamical Approach to the Traveling Salesman Problem: an Efficient Simulation Algorithm, *Journal of Optimization Theory and Applications* 45(1), 41-51.

[7] G. Dueck and T. Scheuer, 1990. Threshold Accepting: A General Purpose Optimization Algorithm Appearing Superior to Simulated Annealing, *Journal of Computational Physics* 90, 161-175.

[8] M. R. Garey and D. S. Johnson, 1979. *Computers and Intractability: A Guide to the Theory of NP Completeness*, W. H. Freeman, San Francisco.

[9] F. Glover, 1986. Future Paths for Integer Programming and Links to Artificial Intelligence, *Computers and Operations Research* 13(5), 533-549,

[10] F. Glover, 1989. Tabu Search - Part I, *ORSA J. on Computing* 1, 190-206.

[11] L. Hagen and A. B. Kahng, 1992. New Spectral Methods for Ratio Cut Partitioning and Clustering, *IEEE Trans. on CAD* 11(9), 1074-1085.

[12] B. Hajek, 1988. Cooling Schedules for Optimal Annealing, *Mathematics of Operations Research* 13, 311-329.

[13] B. Hajek and G. Sasaki, 1989. Simulated Annealing - To Cool or Not, *Systems and Control Letters* 12, 443-447.

[14] P. Hansen and B. Jaumard, 1990. *Algorithms for the Maximum Satisfiability Problem*, *Computing* 44, 279-303.

[15] D. S. Johnson, 1990. Local Optimization and the Traveling Salesman Problem, *Proc. of the 17th International Colloquium on Automata, Languages and Programming*, 446-460.

[16] D. S. Johnson and C. R. Aragon, L. A. McGeoch and C. Schevon, 1989. Optimization by Simulated Annealing: An Experimental Evaluation; Part I, Graph Partitioning, *Operations Research* 37(6), 865-892.

[17] D. S. Johnson, C. R. Aragon, L. A. McGeoch and C. Schevon, 1991. Optimization by Simulated Annealing: An Experimental Evaluation; Part II, Graph Coloring and Number Partitioning, *Operations Research* 39(3), 378-406.

[18] D. S. Johnson, C. R. Aragon, L. A. McGeoch and C. Schevon, 1995. Optimization by Simulated Annealing: an Experimental Evaluation; Part III, the Traveling Salesman Problem, *Operations Research*, forthcoming.

[19] A. B. Kahng, 1992. Exploiting Fractalness in Error Surfaces: New Methods for Neural Network Learning, *Proc. IEEE Intl. Symp. on Circuits and Systems*, 41-44.

[20] A. B. Kahng, 1992. Random Structure of Error Surfaces: New Stochastic Learning Methods, *Proc. SPIE Conf. on Neural Networks and Optimization* 1710(pt. 1, vol.2), 768-779.

[21] A. B. Kahng and G. Robins, 1990. On Structure and Randomness in Practical Optimization, *UCLA Computer Science Department 1990-1991 Annual*, 23-38.

[22] A. B. Kahng and C. W. Tsao, 1992. Old Bachelor Acceptance: A New Class of Non-Monotone Threshold Acceptance Methods, *UCLA CSD TR-920040*, Computer Science Department, UCLA.

[23] S. Kauffman and S. Levin, 1987. Toward a General Theory of Adaptive Walks on Rugged Landscapes, *Journal of Theoretical Biology* 128, 11-45.

[24] B. Kernighan and S. Lin, 1970. An Efficient Heuristic Procedure for Partitioning Graphs, *The Bell System Tech. Journal* 49(2), 291-307.

[25] S. Kirkpatrick, C. D. Gelatt, Jr. and M. Vecchi, 1983. Optimization by Simulated Annealing, *Science* 220(4598), 671-680.

[26] S. Kirkpatrick and G. Toulouse, 1985. Configuration Space Analysis of Traveling Salesman Problems, *Journal de Physique* 46, 1277-1292.

[27] J. B. Lasserre, P. P. Varaiya and J. Walrand, 1987. Simulated Annealing, Random Search, Multi-Start or SAD?, *Systems and Control Letters* 8, 297-301.

[28] I. H. Osman, 1983. Metastrategy simulated annealing and tabu search algorithms for the vehicle routing problem, *Annals of Operations Research* 41, 421-451.

[29] J. Pearl, 1984. *Heuristics: Intelligent Search Strategies for Computer Problem Solving*, Addison-Wesley, Reading, MA.

[30] Y. Rossier, M. Troyon and T. M. Liebling, 1986. Probabilistic Exchange Algorithms and Euclidean Traveling Salesman Problems, *OR Spektrum* 8(3), 151-164.

[31] F. Schoen, 1991. Stochastic Techniques for Global Optimization: A Survey of Recent Advances, *J. Global Optimization* 1, 207-228.

[32] G. Sorkin, 1991. Efficient Simulated Annealing on Fractal Energy Landscapes, *Algorithmica* 6(3), 367-418.

[33] P. Strenski and S. Kirkpatrick, 1991. Analysis of Finite Length Annealing Schedules, *Algorithmica* 6(3), 346-366.