Demon Algorithms and their Application to Optimization Problems

Ian Wood and Tom Downs

Abstract— **We introduce four new general optimization algorithms based on the 'demon' algorithm from statistical physics and the simulated annealing (SA) optimization method. These algorithms reduce the computation time per trial without significant effect on the quality of solutions found. Any SA annealing schedule or move generation function can be used. The algorithms are tested on traveling salesman problems including Grotschel's 442-city problem with results comparable to SA. Applications to the Boltzmann machine are considered.**

Keywords— **Demon algorithm, simulated annealing, optimization, traveling salesman problem, Grotschel's 442-city TSP, Boltzmann machine.**

I. Introduction

We present here a number of optimization algorithms based on the simulated annealing (SA) method. These new methods aim to speed up SA by reducing computation time per trial without sacrificing the quality of solutions. The choice of parameters is kept fairly simple, and applicability to other variations of SA is maintained.

The initial motivation for this study came from an interest in improving the speed of the Boltzmann machine - a recurrent neural net model [1] which requires Gibbs sampling of its internal states at a low 'temperature' equilibrium for both its learning and operational phases. Attainment of a low-temperature equilibrium has been achieved in the past via simulated annealing but is slow enough to deter most people from using the model. Sampling at low temperatures is desirable since the state probability density function is sharpened and learning speed increases.

As a means of improving the speed of the Boltzmann machine one might consider speeding up both the approach to equilibrium and the rate at which sampling can occur.

The issue of fast Gibbs sampling of equilibria is not only important for Boltzmann machines, but also for computational statistical physics. One approach to fast sampling due to Creutz [2], [3] is aimed at the 2-D Ising model of atomic spins in a ferromagnetic lattice. Conventionally, this is simulated using the Metropolis algorithm [4], but Creutz found he could use a computationally simpler algorithm to achieve similar results in far less time.

Creutz's method is known as microcanonical Monte Carlo simulation [2] or the 'demon' algorithm [5]. We prefer the latter term. In its original form the demon algorithm does not aim to generate low energy states, and hence is not directly useful for optimization. Optimization problems can usually be framed in terms of a cost or energy function which is to be minimized over a space of possible solutions. Here we propose four algorithms which vary the operation of the demon algorithm to encourage it to search for optimal or near-optimal solutions. The methods are tested on 200- and 442-city traveling salesman problems and results on the latter are compared with those reported using other optimization methods.

Intelligent Machines Laboratory, Department of Electrical and Computer Engineering, University of Queensland, Australia. E-mail: wood,td@elec.uq.edu.au .

II. The Metropolis Algorithm

The Metropolis algorithm was invented to allow computer simulation of equilibria in statistical physics. An initial state and a temperature are specified, and a Markov chain of system states is generated. Once equilibrium is reached at the required temperature, associated quantities can be approximated from the chain of states. The algorithm can be stated as follows.

1. choose an initial configuration (state) S
2. choose a temperature $T > 0$
3. repeat:
   (a) choose a new configuration S'
   (b) let $\Delta E = E(S') - E(S)$, where $E(S)$ is the energy of configuration S
   (c) if $\Delta E < 0$ accept new configuration, ie: $S = S'$
   (d) else if $rand[0,1] \leq \exp(-\Delta E/T)$ accept new configuration, ie: $S = S'$
   (e) else reject new configuration
4. until stop_condition

The simulation would normally only be stopped once the user has enough samples to calculate equilibrium properties to the desired accuracy.

Each new system state or configuration should be a small stochastic perturbation of the current state. The method of choosing the next state is called the generating function. For discrete parameters, such as those present in the Boltzmann machine, the generating function is usually a uniform random distribution over the closest possible states. For a space of continuous parameters, the generating function is usually a Gaussian, distributed around the current state.

New states are accepted according to an acceptance function which depends on the energy of the current state and the proposed state. The Metropolis algorithm accepts any state transition which will reduce the system energy, and accepts increases stochastically using the function in (3d).

III. Creutz's Demon Algorithm

Creutz's original demon algorithm can be stated as:

1. choose an initial configuration S
2. choose a demon energy $D > 0$
3. repeat:

(a) choose a new configuration S'
(b) let $\Delta E = E(S') - E(S)$
(c) if $\Delta E \leq D$ accept new configuration and update demon, ie: $S = S',\ D = D - \Delta E$
(d) else reject new configuration
4. until stop_condition

Generation of a new configuration (3a) is the same as in the Metropolis algorithm. Any new configuration which would reduce system energy is accepted, as in the Metropolis algorithm. However, the energy lost by the system is given to an artificial variable called a 'demon'. Increases in system energy are only allowed if the demon can provide the necessary energy, which it then loses. As a result, $E(S) + D = C$, a constant, for any state in the Markov chain. Temperature is not specified directly, but can be estimated from the chain of states. Its value is clearly governed by the value of C, which is the energy of the initial state plus the initial demon energy.

The acceptance function for Creutz's method is deterministic and computationally simpler than that of the Metropolis algorithm. It replaces an exponentiation and the generation of a random number with a comparison and a subtraction. The sequence of states produced remains stochastic, but derives its randomness from the generating function.

In [2] Creutz examines a fairly well-understood system from which he can choose initial states at any desired energy within the range of possible values. He does not appear to make a specific effort to generate results at low temperatures. Hence his algorithm as it stands is not directly useful for finding near-optimal solutions to optimization problems or low-energy states of a Boltzmann machine.

IV. Simulated Annealing

Kirkpatrick et al [6] modified the Metropolis algorithm to specifically aim for low energy states, whilst retaining its ability to escape local minima by occasional acceptances of moves which increase the energy.

The Metropolis algorithm already uses a temperature parameter. Kirkpatrick's innovation was to schedule reductions of the temperature each time the system reached quasi-equilibrium until the final temperature was near zero. The system will by then be stuck in a local minimum of the energy function. If the temperature was reduced slowly enough, this may be the global minimum for the system.

Simulated Annealing
1. choose an initial configuration S
2. choose an initial temperature $T = T_0 > 0$
3. repeat:
(a) choose a new configuration S'
(b) let $\Delta E = E(S') - E(S)$
(c) if $\Delta E \leq 0$ accept new configuration
(d) else if $rand[0,1] \leq \exp(-\Delta E/T)$ accept new configuration ie: $S = S'$
(e) else reject new configuration
(f) if quasi-equilibrium reached, reduce temperature according to schedule, eg: $T = \alpha * T$
4. until stop_condition

Simulated annealing is seen to consist of three procedures: a move generating function, a move acceptance function and an annealing schedule. The schedule of temperature reductions is labeled 'annealing' by analogy with the slow cooling of liquids to form large, low energy crystal structures in solids.

Kirkpatrick's original annealing schedule was to set

$$T(n) = \alpha\ T(n-1) \tag{1}$$

$\alpha \in (0,1)$ where n is the number of times annealing has been applied. This negative exponential (or geometric) schedule is quite commonly used in applications and has produced good results.

Another commonly used schedule reduces the temperature linearly from a starting value to 0 (or near zero) over the maximum number of annealing steps.

The starting temperature value is usually determined by steadily increasing the temperature from an initial guess until a value is reached at which most transitions are accepted [7].

Geman and Geman [8] proved that if an inverse log schedule were used, the system would be certain to eventually converge to the global minimum. However in practice this schedule is far too slow to be useful. For example, a drop in temperature from 10 degrees to 1 degree would take $10^{10} - 1$ annealing steps.

Many variations on the original generating function and annealing schedule [6] have been suggested in the 15 years since it was published [7]. Far less work seems to have gone into the acceptance function, and of the examples of which I am aware, only one [9] attempts to significantly reduce the computational complexity of the acceptance function, as we attempt here.

V. Demon Algorithms for Optimization

Here we have altered Creutz's algorithm to guide us from an initial state towards lower energy states as is required for optimization.

Each of our methods revolves around reducing the value of the demon. We employ two main methods for this:
- 'annealing' the demon value, much as Kirkpatrick et al [6] and others [7] have annealed the temperature in simulated annealing
- imposing a fairly low upper bound on the demon, which tends to truncate its value regularly, indirectly lowering system energy.

The two above methods can each be improved by introducing a stochastic demon value, which is normally distributed around a mean. The demon mean then operates in a similar manner to the demon value in the deterministic demon methods. The stochastic demon will occasionally take on high values allowing the system to escape from local minima that it might otherwise have been heavily delayed

or trapped by. This, of course, increases the computational cost of the methods.

The algorithms are as follows:

Bounded Demon Algorithm

1. choose an initial configuration S
2. choose an initial demon energy $D = D_0 > 0$
3. repeat:
   (a) choose a new configuration S'
   (b) let $\Delta E = E(S') - E(S)$
   (c) if $\Delta E \leq D$ accept new configuration and update demon, ie: $S = S', D = D - \Delta E$
   (d) else reject new configuration
   (e) if $D > D_0$, $D = D_0$ - enforce demon upper bound
4. until stop_condition

Randomized Bounded Demon Algorithm

1. choose an initial configuration S
2. choose an initial demon mean energy $Dm = Dm_0 > 0$
3. repeat:
   (a) choose a new configuration S'
   (b) let $\Delta E = E(S') - E(S)$
   (c) if $\Delta E \leq 0$ accept new configuration and update demon mean, ie: $S = S', Dm = Dm - \Delta E$
   (d) else
   i. $D = Dm$ + Gaussian noise value
   ii. if $\Delta E \leq D$, accept new configuration and update demon mean, ie: $S = S', Dm = Dm - \Delta E$
   iii. else reject new configuration
   (e) if $Dm > Dm_0$, $Dm = Dm_0$ - enforce demon mean upper bound
4. until stop_condition

Annealed Demon Algorithm

1. choose an initial configuration S
2. choose an initial demon energy $D = D_0 > 0$
3. repeat:
   (a) choose a new configuration S'
   (b) let $\Delta E = E(S') - E(S)$
   (c) if $\Delta E < D$ accept new configuration and update demon, ie: $S = S'$, $D = D - \Delta E$
   (d) else reject new configuration
   (e) if quasi-equilibrium reached, reduce demon according to schedule, eg: $D = \alpha * D$
4. until stop_condition

Randomized Annealed Demon Algorithm

1. choose an initial configuration S
2. choose an initial demon mean energy $Dm = Dm_0 > 0$
3. repeat:
   (a) choose a new configuration S'
   (b) let $\Delta E = E(S') - E(S)$
   (c) if $\Delta E \leq 0$ accept new configuration and update demon mean, ie: $S = S'$, $Dm = Dm - \Delta E$
   (d) else
   i. $D = Dm$ + Gaussian noise value
   ii. if $\Delta E \leq D$, accept new configuration and update demon mean, ie: $S = S'$, $Dm = Dm - \Delta E$
   iii. else reject new configuration
   (e) if quasi-equilibrium reached, reduce demon mean according to schedule, eg: $Dm = \alpha * Dm$
4. until stop_condition

In the randomized algorithms Gaussian noise is added to the demon mean value. This noise has mean 0 and variance specified by the user as a fraction of the Dm_0 value. This adds a stochastic element to the acceptance calculation and allows rare large increases in energy, ruled out by the deterministic algorithms. However this acceptance function has a computational complexity equal to that of simulated annealing.

Since these algorithms all use the same generating function as simulated annealing and two of the four use annealing, it should be possible to combine them with any of the alternative generating functions (eg: FSA [10], ASA [11]) or annealing schedules (eg: polynomial [7]) that have been proposed.

VI. Traveling Salesman Problem as a benchmark

As a test of the capabilities of the new algorithm, we chose the TSP. For large numbers of cities, this class of problems is recognized as being difficult to solve using general combinatorial optimization algorithms [12]. It has been widely studied and published results exist for many optimization techniques. Also, global optima are known for some large problem instances.

The algorithms tested included the four demon algorithms, as well as standard simulated annealing [6] and a greedy algorithm which only accepts improvements in the cost function. We initially ran comparative tests on randomly generated problem instances of 10, 20, 50, 100 and 200 cities. For these problems, only a little effort was made to choose suitable values for the 'user-defined' parameters. All algorithms were allowed to run to a maximum of 10^7 trials, and simulations were stopped before that if the number of consecutive rejected moves exceeded 50,000.

As a representative example, we show results on one instance of a 200-city TSP, averaged over 5 runs each starting from a random initial tour. City coordinates were chosen from a uniform random distribution over a 10*10 grid.

The move generation rule used was uniform 2-opt [13]. In 2-opt exchange, two cities along the route are chosen and the change considered is that of reversing the route segment lying between the two cities. In uniform 2-opt, the two cities are chosen from a uniform random distribution over all cities. The first city chosen is considered to be the starting point of the tour segment.

We then looked for well-known examples of TSPs with published results from other researchers using their own general optimization algorithms.

In searching for published results on large TSPs, we found a paper by Dueck and Scheuer [9] who use methods similar to some of those outlined here to carry out detailed tests on Grotschel's 442-city problem [14] and Padberg & Rinaldi's 532-city problem [15]. In 1990, these were two of the largest TSPs for which optimal solutions were known.

Knowledge of the global optimum of the cost function allows a more absolute evaluation of the performance of the various algorithms on these two problems.

The data for these two problems and many others is available in the TSPLIB archive at:

http://softlib.rice.edu/softlib/tsplib/ .

Dueck and Scheuer [9] also quote results by other authors on these problems, namely Rossier et al [16] using exhaustive Lin-2-opt and simulated annealing, and Muhlenbein et al [17] using genetic algorithms. 3-opt exchange [13] appears likely to prove a superior heuristic, but we are not aware of any published results in which it has been used for these problems.

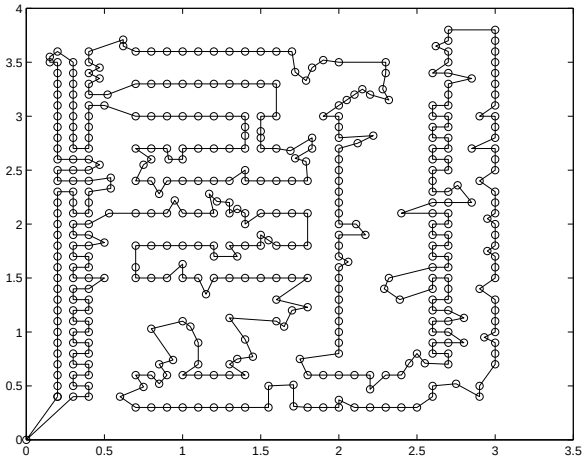

Fig. 1. An optimal solution to Grotschel's 442-city TSP

VII. Dueck and Scheuer's work

Dueck and Scheuer's [9] main algorithm is known as Threshold Accepting (TA) and takes the following form:

1. choose an initial configuration S
2. choose an initial threshold T
3. repeat:
   (a) choose a new configuration S'
   (b) let $\Delta E = E(S') - E(S)$
   (c) if $\Delta E \leq T$ accept new configuration, ie: $S = S'$
   (d) else reject new configuration
   (e) if annealing condition reached, reduce T according to schedule
4. until stop_condition

This algorithm is clearly similar to both the annealed demon algorithm and the bounded demon algorithm. The principal differences are:

- the threshold does not absorb and release energy, unlike the demon.
- the only annealing schedules considered are linear or a problem-specific variation of this.
- the upper bound on individual energy increases is fixed
- unlimited hill-climbing is possible, allowing eventual escape from any deep local minima, as well as unconstrained wandering.

Of the demon algorithms, only the two randomized algorithms allow the possibility of unlimited hill-climbing.

In the bounded demon algorithms, the demon upper bound demon can set higher than TA's threshold values can since the average demon value tends to be much lower than the initial level.

The demon value in the the bounded demon algorithms plays much the same role as the threshold in TA. However the average demon value is usually much lower than the demon upper bound, so this bound can be set much higher than the threshold in TA while maintaining a similar rate of energy reduction. The demon value's variation allows the algorithm to occasionally accept state transitions involving much larger increases in energy than can be allowed under TA.

In a later paper Dueck [18] presents two further algorithms labeled "The Great Deluge" and "Record to Record Travel". The first of these allows transitions in exactly the same way as the annealed demon algorithm. However, where we anneal the demon value, Dueck effectively anneals the sum of system energy and demon energy. The second method accepts transitions in the same way as the bounded demon algorithm.

Dueck does not report any results for either of these two algorithms with a stochastic generating function. He uses a deterministic generating function, developed specifically for the traveling salesman problem, which greatly increases the speed of the algorithm. This method considers only cities which are near neighbours for 2-opt updates, a method that will work well in many TSPs.

VIII. Results

All of the new algorithms require the choice of an initial demon value. This choice is quite important for the bounded demon algorithms since it is also an annealing control. Not surprisingly, testing has shown it is much less important for the annealed demon algorithms, in which the choice of α (see eqn. 1) is more important. We have however generally found that all the algorithms were less sensitive to the choice of these parameters than simulated annealing was to the choice of α. Other parameters, such as those involved in the determining of quasi-equilibrium, were chosen as advised in [7].

A. Random scatter 200-city problem

TABLE I
200-city TSP results

Algorithm	Average	Best	Avg. Trials
SA	106.28	104.10	1.09 M
Annealed Demon	104.71	103.52	4.22 M
Bounded Demon	106.15	105.02	10 M
R Annealed Demon	103.75	102.95	7 M
R Bounded Demon	105.66	105.30	10 M
Greedy	115.20	112.85	0.15 M

The fact that some of the algorithms terminated well short of the maximum number of trials is some cause for concern, since it is expected that more trials would lead to better results. The following tests showed that a very careful choice of parameters could result in runs of any desired length, and correspondingly better results.

B. Grotschel's 442 city problem

Fig 1 shows an optimal solution to Grotschel's 442-city problem [14]. The optimal tour has a length of 50.78 units. One can notice from the figure that the cities in this problem are not distributed in a random scattering. A great deal of clustering and lining up of cities is present. However, we are unsure as to whether this makes the problem easier or harder for optimization algorithms than a random scatter of 442 cities in the same grid.

Results from [16] using simulated annealing and from [17] using genetic algorithms are summarized in Table II. Both report only their best results.

Rossier et al [16] introduced a 'Distance' heuristic for the problem, which requires that the two cities chosen for consideration of a 2-opt move must lie within a .45 radius of each other. This is the maximum distance to a neighbour for any of the 442 cities in Grotschel's problem. Cities in this problem have on average around 20 neighbours within this radius, and examination of the optimal solution (fig. 1) shows that only one distance along the route exceeds .45, indicating the likely usefulness of this problem-specific heuristic.

Dueck and Scheuer [9], [18] use this heuristic extensively, and with good results. We show results for both our own work (Tables IV, V, VI, VII and VIII) and that of [9] (Table III) with the heuristic (Distance) and without (Standard) on the 442-city problem. Each line contains the average and best results over 25 random starting tours.

TABLE II
Simulated Annealing and Genetic Algorithms

Algorithm	Best result	Trials
Lin-2-opt	55.48	unlimited
SA Standard	53.30	2 M
SA Distance	51.765	2 M
Genetic Algorithm	51.21	unknown

TABLE III
Threshold Accepting - Dueck

Algorithm	Average	Best	Trials
TA Standard	52.96	51.94	2 M
TA Distance	51.53	51.07	1.5 M
TA Distance	51.51	50.97	2 M
TA Distance	51.36	50.95	4 M

TABLE IV
Standard Demon algorithms - 2 M trials

Algorithm	Average	Best
Simulated Annealing	54.35	53.34
Bounded Demon	53.69	52.28
R Bounded Demon	53.97	53.48
Annealed Demon	54.73	53.62
R Annealed Demon	54.44	53.16
Greedy 0.8 M	57.20	55.74

TABLE V
Distance Demon algorithms - 1.5 M trials

Algorithm	Average	Best
Simulated Annealing	51.73	51.23
Bounded Demon	52.31	51.60
R Bounded Demon	52.47	51.86
Annealed Demon	51.89	51.32
R Annealed Demon	51.81	51.27
Greedy 0.1 M	58.06	56.83

TABLE VI
Distance Demon algorithms - 2 M trials

Algorithm	Average	Best
Simulated Annealing	51.72	51.23
Bounded Demon	52.24	51.60
R Bounded Demon	52.36	51.77
Annealed Demon	51.74	51.19
R Annealed Demon	51.75	51.26

TABLE VII
Distance Demon algorithms - 4 M trials

Algorithm	Average	Best
Simulated Annealing	51.54	51.14
Bounded Demon	52.08	51.27
R Bounded Demon	52.18	51.55
Annealed Demon	51.74	51.19
R Annealed Demon	51.62	50.95

TABLE VIII
Distance Demon algorithms - 10 M trials

Algorithm	Average	Best
Simulated Annealing	51.32	50.85
Bounded Demon	51.91	51.23
R Bounded Demon	51.96	51.40
Annealed Demon	51.41	51.09
R Annealed Demon	51.39	50.93

The SA and demon algorithm simulations were done with fairly careful selection of the user parameters. For instance, the negative exponential schedule (eqn. 1) is governed by the parameter α. Although many texts, eg: [7], suggest

choosing α in the range .85 .. .99., we found that values around .99994 were the most successful for SA in the 10 M trial simulations. These reduced the system temperature from an initial value, commonly 20, to a final value of .01. This was low enough to ensure the rejection of most energy increasing moves, meaning the system was unlikely to evolve further.

IX. Comparison

TABLE IX
Algorithm Complexity - Acceptance Function

Algorithm	$\Delta E > 0$ reject	$\Delta E > 0$ accept
Metropolis Alg	m,e,r	m,e,r
Creutz' Demon	c	a,c
SA	m,e,r	m,e,r
Bounded Demon	c	a,c
R Bounded Demon	c,3m,e,r	a,c,3m,e,r
Annealed Demon	c	a,c
R Annealed Demon	c,3m,e,r	a,c,3m,e,r
Greedy	-	-
TA	c	c

When $\Delta E \leq 0$ all the algorithms will accept the move with very little computation - at most an addition and a comparison. If $\Delta E > 0$ the steps can be a little more demanding. Table IX compare the computational complexity of the algorithms. Operations are classed as a (addition and subtraction), c (compare), m (multiplication and division), e (exponentiation) and r (random number generation).

The annealing operation is only performed on average every N trials, where N is likely to be over 100, and requires just a multiplication or an addition for negative exponential schedules or linear schedules respectively. The generating function requires one random number to be generated in all cases.

X. Discussion

Dueck and Scheuer [9] admit that their 'annealing' schedule was optimized for this particular problem. The values chosen are not far removed from a linear annealing schedule, which would require only one parameter - the initial value. However, they choose 30 values for the threshold, each apparently held for 1/30 th of the total number of trials. This can be considered as the choosing of 30 parameters. Similar results are reportedly obtained [9] using a linear schedule, but no details are given.

Our exponential annealing schedule requires two parameters - the initial demon value and α (eqn. 1). A linear schedule is also possible, but has not yet been tried. We believe that an annealing schedule requiring any more parameter choices places an unnecessary burden on the user.

XI. Further work

We intend to test a linear annealing schedule on the demon in the annealed demon algorithms, as well as on the demon upper bound and the demon standard deviation in the randomized demon algorithms. Other distributions will also be considered for the randomized demon algorithms.

We intend applying the optimizing demon algorithms to the Boltzmann machine in place of simulated annealing for finding low-temperature equilibria. We will also try Creutz' demon algorithm for Gibbs sampling of Boltzmann machine equilibria as required by the learning rule.

References

[1] D. H. Ackley, G. E. Hinton, and T. J. Sejnowski, "A learning algorithm for Boltzmann machines," *Cognitive Science*, vol. 9, pp. 147–169, 1985. [Reprinted in Anderson.Rosenfeld.88].

[2] M. Creutz, "Microcanonical Monte Carlo simulation," *Physical Review Letters*, vol. 50, no. 19, pp. 1411–1414, 1983.

[3] G. Bhanot, M. Creutz, and H. Neuberger, "Microcanonical simulation of Ising systems," *Nuclear Physics B*, vol. 235, pp. 417–434, 1984.

[4] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller, "Equation of state calculations by fast computing machines," *Journal of Chemical Physics*, vol. 21, pp. 1087–1092, 1953.

[5] H. Gould, L. Spornick, and J. Tobochnik, *Thermal and Statistical Physics Simulations*. New York: Wiley, 1995.

[6] S. Kirkpatrick, C. Gelatt, and M. Vecchi, "Optimisation by simulated annealing," *Science*, vol. 220, pp. 671–680, 1983.

[7] E. Aarts and J. Korst, *Simulated Annealing and Boltzmann Machines*. Chichester: Wiley, 1989.

[8] S. Geman and D. Geman, "Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 6, pp. 721–741, 1984.

[9] G. Dueck and T. Scheuer, "Threshold accepting: A general purpose optimization algorithm appearing superior to simulated annealing," *Journal of Computational Physics*, vol. 90, pp. 161–175, 1990.

[10] H. Szu and R. Hartley, "Fast simulated annealing," *Physics Letters A*, vol. 122, pp. 157–162, 1987.

[11] L. Ingber, "Simulated annealing: Practice versus theory," *Mathematical and Computer Modelling*, vol. 18, no. 11, pp. 29–57, 1993.

[12] G. Reinelt, *The Traveling Salesman - Computational Solutions for TSP Applications*. Berlin: Springer-Verlag, 1995.

[13] S. Lin, "Computer solutions of the traveling salesman problem," *The Bell System Technical Journal*, vol. 44, pp. 2245–2269, 1965.

[14] M. Grötshel, Preprint no. 38, Polyhedrische Kombinatorik and Schnittebenverfahren, Universitat Augsburg, Germany, 1984.

[15] M. Padberg and G. Rinaldi, "Optimization of a 532-city symmetric traveling salesman problem by branch and cut," *Operations Research Letters*, vol. 6, pp. 1–7, 1987.

[16] Y. Rossier, R. Troyon, and T. Liebling, "Probabilistic exchange algorithms and Euclidean traveling salesman problems," *OR Spektrum*, vol. 8, pp. 151–164, 1986.

[17] H. Muhlenbein, M. Gorges-Schleuter, and O. Kramer, "Evolution algorithms in combinatorial optimization," *Parallel Computing*, vol. 7, pp. 65–85, 1988.

[18] G. Dueck, "New optimization heuristics - the great deluge algorithm and the record-to-record travel," *Journal of Computational Physics*, vol. 104, pp. 86–92, 1993.