Available online at www.sciencedirect.com

SCIENCE DIRECT®

ELSEVIER

Physica A 349 (2005) 649–666

PHYSICA A

www.elsevier.com/locate/physa

Weight annealing

Matan Ninio[a], Johannes J. Schneider[a,b,*]

[a]*School of Engineering and Computer Science, The Hebrew University of Jerusalem, Givat Ram, Jerusalem 91904, Israel*

[b]*Department of Physics, Johannes Gutenberg University of Mainz, Staudinger Weg 7, 55099 Mainz, Germany*

Received 5 August 2004; received in revised form 10 October 2004
Available online 11 November 2004

Abstract

Simulated Annealing has become a standard optimization technique for a wide variety of problems: starting at a random configuration and performing a sequence of moves, the system is optimized using a control parameter which partially allows for accepting a deterioration and therefore for climbing over barriers in the energy landscape. Our approach, Weight Annealing, changes the energy landscape by assigning variable weights to the single parts of the proposed problem. We describe the philosophies behind these algorithms and present results for the Traveling Salesman Problem and the Sherrington–Kirkpatrick-model for spin glasses.
© 2004 Elsevier B.V. All rights reserved.

PACS: 05.10.Ln; 02.70.Uu; 89.75.−k

Keywords: Optimization; Simulated annealing; Energy landscape; Traveling Salesman Problem; SK-model

1. Introduction

Many optimization heuristics have been developed over the last decades in order to solve problems approximately for which no analytical solutions exist.

*Corresponding author. Department of Physics, Johannes Gutenberg University of Mainz, Staudinger Weg 7, 55099 Mainz, Germany. Tel.: +49 6131 3923646; fax: +49 6131 3925441.
E-mail addresses: matan@cs.huji.ac.il (M. Ninio), jsch@cs.huji.ac.il, schneidj@uni-mainz.de (J.J. Schneider).

0378-4371/$ - see front matter © 2004 Elsevier B.V. All rights reserved.
doi:10.1016/j.physa.2004.10.022

Improvement heuristics start with an arbitrary configuration of the given problem instance and try to improve it gradually by applying a sequence of moves, which change the configuration slightly. The simplest of these heuristics is the Greedy algorithm (GRE), in which the moves to be applied are chosen at random and which only accepts the changes made by the move if the tentative new configuration is better than or at least equally good as the current one, otherwise it is rejected and one stays with the current configuration. Thus, the probability p for accepting the move from a configuration σ to a configuration τ is given by

$$p(\sigma \to \tau) = \begin{cases} 1 & \text{if } \Delta H = H(\tau) - H(\sigma) \leqslant 0\,, \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

with H being the cost or energy function of the proposed problem.

However, using this GRE, one usually gets stuck in high-lying local minima in the energy landscape from which the system cannot free itself as it is not allowed to climb up a hill in the energy landscape. Thus, one usually works with more elaborate acceptance functions which also allow for deteriorations but which reduce the probability of accepting such a deterioration during the optimization run by means of a control parameter. A widely used optimization algorithm of this kind is Simulated Annealing (SA) [1]. SA simulates a cooling process in which a system is transferred from a high-energetic unordered regime to a low-energetic ordered solution by decreasing the temperature T gradually. When using SA, the Metropolis acceptance probability

$$p(\sigma \to \tau) = \begin{cases} \exp(-\Delta H/T) & \text{if } \Delta H > 0\,, \\ 1 & \text{otherwise} \end{cases} \tag{2}$$

is typically used.

SA allows for climbing over barriers in the energy landscape and thus typically ends up at much better solutions than the GRE. The control parameter T governs the probability for accepting a deterioration, such that at high temperatures the system performs a quasi random walk and at low temperatures the system is in the Greedy mode. By the decrease of the control parameter, the system is transferred between these regimes.

In contrast to SA, there are also several approaches to change the energy landscape, such that also the GRE can end up in quite good solutions. One example for this procedure is Search Space Smoothing [2–4], in which one starts with a completely flat energy landscape. This energy landscape is then gradually desmoothed until the original energy landscape is achieved at the end. After each desmoothing step, a Greedy optimization run is performed, which starts with the final configuration of the previous Greedy optimization run and thus with a locally minimum configuration in the previous energy landscape. The minima are usually shifted such that formerly minimum configurations are no longer locally minimum. If the desmoothing process is performed carefully enough, one can trust in a guidance effect: it leads the formerly locally minimum configuration to the new slightly shifted minimum in the energy landscape. This is also true for the global optimum of the first quasi flat landscape, such that the hope is that when starting from this configuration, the optimization run will end up in an at least very good local minimum, perhaps even in the global optimum of the original landscape.

Another approach in this field of working with various energy landscape works as follows: one starts with the original problem instance, performs a Greedy optimization run for it, and then switches to a slightly differing instance [5–7]. One uses the final configuration of the Greedy run for the original instance as an input for the starting configuration of the slightly altered instance. Starting from this configuration, one again performs a Greedy optimization run. Now there are two possibilities: one either switches back to the original instance, taking the result of the Greedy run for the changed instance as a starting configuration for a new Greedy run, and thus switches between Greedy runs for the original instance and slightly changed instances. But one could also proceed with Greedy runs between a series of instances which are derived from the original instance by changing it slightly. Of course, in both cases the series has to end with a Greedy run for the original instance. This approach is based on the belief that similar problem instances exhibit similar energy landscapes. The large structures in the energy landscapes should be roughly the same, especially the broad deep valleys. Thus, one will more likely end up in a local minimum inside such a deep valley and therefore in a better configuration as if working with the GRE on the original landscape only.

2. The weight annealing method

The methods sketched above have in common that they want to find a way out of being trapped in a high-lying local optimum. But none of them use any additional knowledge about where the considered problem instance is easy and where it is hard to solve. Instead, they consider the energy landscape as a constant property of the proposed optimization problem even if they change it during the optimization process in order to overcome barriers in the energy landscape. But basically they want to solve the original problem with the original energy landscape by applying some external parameters to it without having a closer look at the specific local properties of the problem instance.

When looking at different solutions achieved for the proposed problem instance, one often finds that they are rather well solved for some local parts whereas other local parts are solved or at least seem to be solved rather badly. Like this eye-measure, one often neglects or tries to neglect long-range interactions within the complex system and tries to write down the energy or cost function as a sum of local Subhamiltonians, each of them measuring how well a local part is solved:

$$H = \sum_{i=1}^{N} H_i \, . \tag{3}$$

Of course, one wants to improve the parts which are solved worse, and therefore wants the optimization algorithm to put stronger emphasis on these parts. To achieve this, one can assign a weight W_i to each part i of the system, depending on how well this part of the system is solved. This set of weights defines a new weighted cost function H_W over the N parts of the system,

$$H_W = \sum_{i=1}^{N} W_i \times H_i \, . \tag{4}$$

Now the question arises how to choose these weights. Of course, this can be done by looking at various solutions of the problem instance, but it might be even better if one could use internal properties of the problem instance. Such a knowledge of the internal can be achieved by deeper insight in the problem, but also by looking at various solutions for a problem instance: first of all, the nature of the problem has to be considered: of course, one can often map a problem onto another problem, the nature of which has already been quite well understood. But such a mapping procedure often leads to the additional problem that some constraints of the original problem are considered by using penalty functions which add some virtual costs to the Hamiltonian of the problem if the corresponding constraints are violated and which should vanish at the end of the optimization run, as then one has to get a feasible solution. An example for this is the mapping of the Traveling Salesman Problem (TSP) onto a problem with binary variables [8]: the traveling salesman has the task to find the shortest closed roundtrip through a given set of nodes. The constraint that each node is visited exactly once has to be introduced via penalty functions if working with binary variables $\eta_{i,a}$ with $\eta_{i,a} = 1$ if node i is the ath stop in the tour and 0 otherwise. A better modelling of a configuration of the Traveling Salesman Problem is a permutation of the numbers $1, \ldots, N$. Each node is represented by a number. The constraint that there is exactly one roundtrip containing each node once is automatically fulfilled in this representation. Generally, it is preferable to model a problem in a way that as many hard constraints as possible are automatically fulfilled in each configuration. Moves should—wherever possible—lead only to such feasible configurations which fulfill these constraints and should ensure that each such configuration can be reached from any other configuration by applying a finite sequence of moves in a Random Walk.

When considering this nature of the problem, one can often determine why the proposed optimization problem is not trivially solvable: for example, the cost function H of the problem can consist of several functions, $H = \sum_i H_i$, which compete with each other in the sense that a move leading to an improvement according to some part H_j might lead to an overall deterioration due to the other parts of the cost function. Another example is that a problem instance might have to be solved rather badly somewhere locally in order to get an overall good solution. There are also systems in which competing interactions do not allow solutions in which all interactions are fulfilled. Such competing cost functions, parts, and interactions lead to so-called frustration effects in the system [9].

This deeper insight in the internal properties of the problem can be used for getting estimates how well the system can be solved; for example, if the cost function of the system can be written as a sum of competing addends H_i, then one can, e.g., solve the system according to each of the H_i separately, which provides a lower bound for the optimum solution. Analogously, one can have a look at the pessima according to the various H_i for getting an upper bound. If one has understood the frustration effects due to competing parts or interactions in the system, one can have a look at the corresponding unfrustrated system and often get an exact cost value for its ground state, which can also serve as a lower bound for the optimum of the original problem. An example for this is the Held–Karp-bound for the TSP, in which the problem instance is modelled with binary variables which are then relaxed to real numbers in the interval $[0; 1]$ in order to get a problem which can be solved exactly with the Simplex algorithm [10,11].

If such analytical considerations do not help any further, one has to perform several optimization runs and have a close look at various solutions: when looking at the various parts with the eye, one often sees what parts are well and what are rather badly solved. The cost value can be split according to the single parts of the problem or according to the addends in the cost function, providing best-so-far energy values for these parts. This information can also be used for choosing the weights.

Regardless of the specific choice of the weights, the general outline of the Weight Annealing (WA) algorithm is as follows:

(1) One starts off with an initial configuration, which can, e.g., be achieved by using the GRE on the original landscape.
(2) Then one determines a new set (W_i) of weights, based on the result of the previous optimization run and possibly also on further insight into the problem.
(3) Then one performs a new Greedy optimization run starting with the resulting configuration of the previous optimization run and using the new weight values.
(4) Finally, one returns to Step 2 until some stopping criterion is met.

Step 1 of the algorithm can be considered to be a special case of an iteration (as described in Step 3) of this WA algorithm with the initial values $W_i \equiv 1$ for all weights.

This outline leaves a wide range of different possible ways to choose the weights. There might also be some meta-parameters, like some control parameter T, which govern the amount by which each single weight can be changed. For example, one wants to start off allowing significant changes to the weights and wants to finish the run with the original weights $W_i = 1$, so the output solution would be optimized within the energy landscape of the original problem instance. This imposes—in a physicists' language—a "cooling schedule" on the weights W_i which has to end with all weights being equal to 1.

Any weighting scheme must comply to some requirements in order to be used in WA. The first requirement is that all the weights must remain non-negative, as a negative weight will cause the system to look for the worst possible local solutions for the matching part. The second requirement is that the weighting scheme depends on the control parameter T in such a way that when T is large, weights can be chosen freely, while as T approaches zero all the weights approach one. For normalization, one can generally choose the weights in such a way that the sum of the weights remains constant, e.g., that the sum of the weights divided through the number of the weights is one.

There are several ways to choose the weights. As a first step, we want to consider the case in which the single weights W_i are chosen randomly and thus independently of how well the system is solved locally. Next one may want the weights to consider how difficult it is to solve a local part of the system optimally.

2.1. Random reweighting

The natural distribution to use for T-dependent weights is the Gaussian distribution with mean 1 (around the original weights) and variance monotonically dependent on the temperature. In our case, this is an inadequate solution, as some of the weights may end up negative. Instead, we follow the approach in Ref. [12] and use the Γ-distribution:

$$\Gamma_{K,b}(x) = \frac{x^{K-1}\exp(-x/b)}{b^K \Gamma(K)} . \tag{5}$$

This distribution is particularly appropriate:

- The Gamma distribution is continuous.
- The Gamma distribution is non-negative.
- The sum of l independent, $\Gamma_{K,b}$-distributed variables is $\Gamma_{l\times K,b}$-distributed. In order to normalize the expected weight to 1, we set the scale parameter b to $1/K$. The variance therefore equals the inverse of the shape parameter: $\sigma^2 = 1/K$.
- For small variances, this form of the Gamma distribution approaches the Gaussian distribution around the mean.

To allow for a "cooling schedule" as mentioned above, we used the temperature as the variance of the Gamma distribution.

2.2. Adversarial reweighting

As an alternative approach, one might also want to consider how well the system is solved locally, and place more emphasis on the parts that are farther from optimal. Again following the approach in Ref. [12], we suggest the following Min–Max approach to choose the weights:

- For the current configuration, estimate how well each part is solved.
- Write H_W as a function of these local estimates.
- Use a distance $\delta(W, W^0)$ over the weight vectors and define the T-neighborhood of the original weights as the set of all weight vectors within T-distance.
- Search this T-neighborhood with some local search algorithm for a set of weights (W_i^*) such that H_{W^*} is as large as possible.

Instead of searching the T-neighborhood for a maximum of H_{W^*}, we can perform a search—without a neighborhood restriction—for a maximum of $H_{W^*} - \beta\delta(W^*, W^0)$, where β controls as a penalty factor the relative significance of δ.

Following the work in Ref. [12], we chose $\beta = 1/T$ and the distance δ to be the Kullback–Leibler measure [13] for the divergence between two distributions

$$\delta(W, W^0) = \mathrm{KL}(W||W^0) \equiv \sum_i W_i \log(W_i/W_i^0)\,, \tag{6}$$

where W_i^0 are the original weights (usually set to 1).

The reweighting step attempts to "worsen" the formerly minimum configuration, in which the previous search process ended up, as much as possible. This step produces a new set of weights. They define a new weighted Hamiltonian, which is a new challenge for the search algorithm, which restarts at the previous stopping point. If the temperature is high, the reweighting step has enough freedom to change the energy landscape in a way that the previous local minimum is typically no longer locally minimal. As the temperature approaches zero, the reweighting step has less possibilities to change the weights. As the changes of the weights must be smaller for smaller temperatures, we can assume that when the temperature approaches zero, a good local minimum will remain with high probability locally minimal, allowing the algorithm to converge to a good solution.

2.3. *Variations of the reweighted Hamiltonian*

Instead of using the weighted Hamiltonian H_W, we can use some measure related to it as long as the two measures reach the same minima as the weights W_i approach the value 1.

For example, one might have some deeper insight in the problem and know that there are lower bounds $H_i^{\text{lower bound}}$ for the values of the local parts H_i of the Hamiltonian. These lower bounds can be calculated in various ways, e.g., by relaxing the proposed integer problem into a non-integer problem or by looking at the corresponding problem without frustration. The original Hamiltonian does not change if we subtract these lower bounds from the corresponding addends,

$$\tilde{H} = \sum_i (H_i - H_i^{\text{lower bound}}) = H - \sum_i H_i^{\text{lower bound}} = H - \text{const.} \tag{7}$$

as then only a constant is subtracted from the Hamiltonian. However, if reweighting this Hamiltonian,

$$\tilde{H}_W = \sum_i W_i \times (H_i - H_i^{\text{lower bound}}) \tag{8}$$

and giving this reweighted Hamiltonian to the search process for the reweighting technique, the outcome of this technique has to be quite different: Here now more emphasis is not given to those addends which are absolutely large but to those which could—from the point of view of the bounding function—be solved much better. If the bounding functions provide good estimates, then this approach is surely superior to the original approach.

However, it is not always possible to easily find some good bounding functions. Furthermore, many complex problems have to be solved at some local part in a bad way in order to get to an overall excellent solution, whereas their unfrustrated counterparts do not exhibit this property. For at least these problems, another approach is even better: again one needs some measure for how well the system is solved locally. One can store the values H_i^{bsf} which represent the best cost function values for the single parts found so far in the previous iterations of the WA approach. Given these values instead of the $H_i^{\text{lower bound}}$, one does not necessarily know how well the parts are solved, as one could have solved one part more or less badly in all iterations. If one has to solve it in a bad way in order to get to an overall good solution, then there is nothing to say against that. However, it might be that this approach is not able to introduce a large enough pressure on the system to solve this part in a much better way. A second problem which occurs within this variation is that it might be rather unstable: imagine that some iteration of our algorithm leads to a new record for some local part i. Then the current value for H_i and H_i^{bsf} coincide. The reweighting process then thinks that this part is solved in an optimum way, that there is nothing which can be improved there, such that it puts much less emphasis on this part. Therefore, the next iteration of the algorithm will probably lead to a solution which is solved much worse in this local part. The algorithm will then put more emphasis again on this part, such that in the following iteration, the algorithm may return to a solution with $H_i = H_i^{\text{bsf}}$. This cycle may even be iterated again and again.

In order to overcome this instability problem, one can e.g., subtract a small amount from all H_i^{bsf} values. Of course, this is “lying to the algorithm”: the algorithm might be in a solution with all $H_i = H_i^{\text{bsf}}$ which happens also to be the globally optimum of the problem. Then we tell the algorithm that there is still something to do.

However, we want to stay first with the “first step” approach to work with randomly chosen weights.

3. Application to the TSP and the SK-model for spin glasses

The question is now how to apply this algorithm to practical problems. We want to sketch the possible applications at two prominent examples, namely the TSP and the Sherrington–Kirkpatrick-model (SK-model) for spin glasses.

The TSP is given by a $N \times N$ distance matrix D between N nodes [14,15]. It is the task of the traveling salesman to find the shortest closed tour through these N nodes, touching each node exactly once and returning to the initial node at the end. Each configuration can be coded as a permutation σ of the numbers $1, \ldots, N$. The Hamiltonian of this problem is thus given by

$$H(\sigma) = D(\sigma(N), \sigma(1)) + \sum_{i=1}^{N-1} D(\sigma(i), \sigma(i+1)) \,. \tag{9}$$

Usually, the symmetric TSP is considered in which $D(i,j) = D(j,i)$.

We implemented two possible moves for changing a configuration, namely the Lin-2-Opt, which turns around a part of the tour, and that possibility of the Lin-3-Opt that exchanges two successive parts of the tour without changing their direction [16,17]. Both moves are shown in Fig. 1. The Lin-2-Opt is called with a probability of $\frac{1}{6}$. Of course, for both moves the energy difference ΔH can be easily calculated by adding up the lengths of the new edges and subtracting the lengths of the cut old edges.

The application of SA to the TSP is straightforward: one starts with a rather large temperature T_0, decreases T logarithmically by a factor of, e.g., 0.95, and ends at a final temperature at which the system is frozen in a local minimum or hopefully the global optimum. Each move is accepted or rejected with the Metropolis acceptance criterion.

In the case of WA, we have first to state that the local parts are the nodes of the TSP. We have to rewrite the Hamiltonian as

$$H(\sigma) = \sum_{i=1}^{N} \frac{1}{2}(D(\sigma(i), \sigma(i+1)) + D(\sigma(i), \sigma(i-1))) \tag{10}$$

(with $\sigma(0) \equiv \sigma(N)$ and $\sigma(N+1) \equiv \sigma(1)$) in order to get the local cost functions

$$H_i(\sigma) = \tfrac{1}{2}(D(\sigma(i), \sigma(i+1)) + D(\sigma(i), \sigma(i-1))) \tag{11}$$

for the local parts. A weight W_i is then assigned to each node. Rewriting the reweighted Hamiltonian in just the contrary way, one achieves the weighted distance values

$$D_W(i,j) = \frac{W_i + W_j}{2} D(i,j) \,. \tag{12}$$

Each move is accepted or rejected with the Greedy acceptance criterion, however, based on the weighted cost function H_W, which works with the weighted distance matrix D_W instead of the original distance matrix D. Thus, the application of WA is also very straightforward if a program for the Greedy algorithm already exists. After

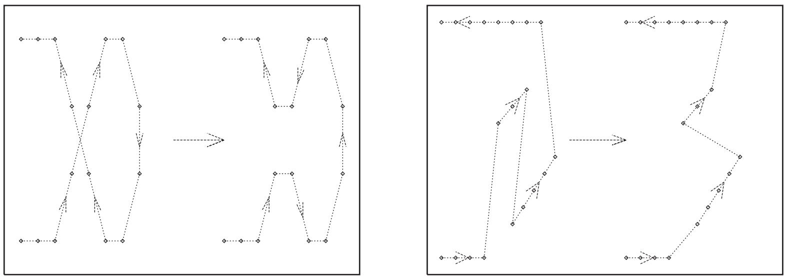

Fig. 1. Two possible moves for the TSP: the Lin-2-Opt (left) removes two edges from the tour, turns around one of the two tour parts and connects them in a new way. There are four possibilities to perform a Lin-3-Opt, i.e., a move which removes three edges and connects the three tour parts in a new way. We implemented that type which exchanges two parts of the tour without changing their directions (right).

the end of a Greedy optimization run, the weight values W_i are changed, the distance matrix D_W is recalculated, and a new Greedy run is performed, starting with the resulting configuration of the previous run and based on the new weight values.

The SK-model is one of the standard models for spin glasses with long range interactions between the single spins. Each of the N Ising-spins σ_i, which can only take the values $+1$ and -1, interacts with all other spins via an interaction matrix J, such that the Hamiltonian of this SK-model [18] is given by

$$H(\sigma) = -\frac{1}{2N} \sum_{i,j=1}^{N} J_{i,j}\sigma_i\sigma_j \, . \tag{13}$$

The interactions $J_{i,j}$ are Gaussian distributed. We implemented the single spin flip, i.e., the move $\sigma_i \to -\sigma_i$, as move for jumping from one configuration to another one.

The application of SA to this problem is as straightforward as for the TSP. Again one accepts or rejects the move with the Metropolis acceptance criterion and reduces the temperature logarithmically from a high value in which the system nearly moves at random to a low value at which it is frozen.

The WA method assigns weights to the single spins, such that a weighted interaction matrix J_W can be derived:

$$J_W(i,j) = \frac{W_i + W_j}{2} J(i,j) \, . \tag{14}$$

In each step of the algorithm, a Greedy optimization run based on a newly weighted interaction matrix is performed.

4. Computational results

In order to demonstrate the quality of our approach, we will apply it to five benchmark instances of the TSP, namely the BEER127, LIN318, PCB442, ATT532, and NRW1379 instances, which can be downloaded from Reinelt's TSPLIB95 [19]. Thus, we cover the range between 100 and more than 1000 nodes which consists of non-trivial and still real-life TSP instances. Furthermore, we generated three instances of the SK-model with 100, 200, and 400 spins, respectively, which we call here SK100, SK200, and SK400. For each instance, 1000 optimization runs were performed, each for the Greedy algorithm, SA and WA. The cooling schedules for SA and WA were chosen in such a way that the number of temperature steps was the same, that the system was frozen at the final temperature, and that the initial temperature was large enough to be far beyond the transition temperature. The overall number of sweeps performed in these simulations was thus generally the same, in order to make the results comparable.

Fig. 2 shows the decrease of the energy with decreasing temperature both for a TSP and a SK instance if applying SA. We find in both cases a nice sigmoidal decrease from the unordered high-energy region to an ordered low-energy solution, in which the system finally freezes.

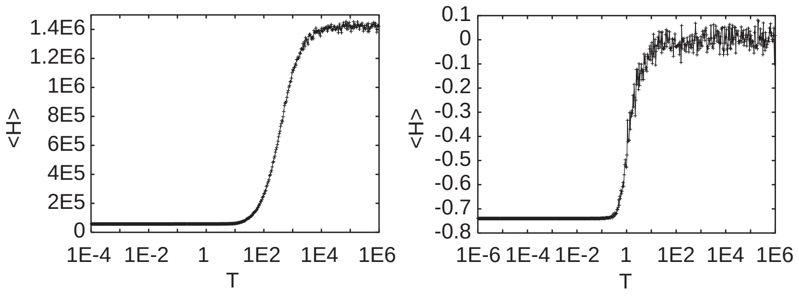

Fig. 2. Decrease of the energy $\langle H \rangle$ with decreasing temperature T if using SA: curves for the NRW1379 TSP instance (left) and for an instance of the SK-model with 400 spins (SK400, right).

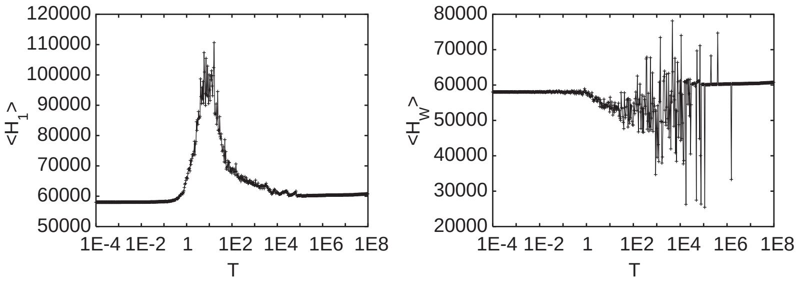

Fig. 3. Change of the energy $\langle H \rangle$ with decreasing temperature T if using WA: curves of the NRW1379 TSP instance for the original Hamiltonian (left) and for the reweighted Hamiltonian (right).

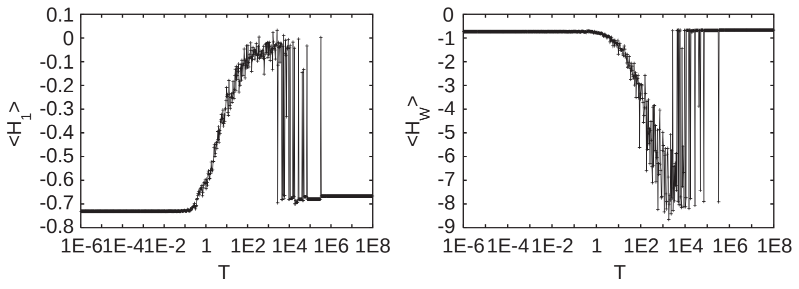

Fig. 4. Change of the energy $\langle H \rangle$ with decreasing temperature T if using WA: curves of the SK400 spin glass instance for the original Hamiltonian (left) and for the reweighted Hamiltonian (right).

The curves for WA, which are shown in Figs. 3 and 4, look completely different from those for SA. First of all, there are two curves, one for the original Hamiltonian $H = H_1$, for which all weights are set to 1, and secondly for the weighted Hamiltonian H_W. We find for the NRW1379 and the SK400 instance roughly the same behavior and can divide the cooling process in four steps: (a) for very large temperatures, our WA implementation lets the system drop into some local minimum. Here the weights stay all the same. Therefore, in this range, the WA approach works in rather the same way as the GRE. The system is frozen in some local minimum. (b) With decreasing temperature, an increase of $\langle H_1 \rangle$ can be observed, which is rather smooth for the TSP instance, whereas it takes place in an abrupt way after many fluctuations for the SK instance. Here the WA algorithm really starts to work, as can be seen at the curves for the weighted Hamiltonian, where one can see that WA is again and again forced out of the local minimum (upper peaks in the curve for the TSP instance) and tries to minimize the energy (lower peaks, which occur in both curves more often). But due to the large temperature values, basically some randomness is added to the system, as the weights are allowed to develop rather freely. Therefore, the energy measure according to the original Hamiltonian increases. (c) Decreasing the temperature even further, we find in both cases a more or less wide sigmoidal decrease of H_1 while H_W increases at the same time. Obviously, here the optimization of the system takes place. The altered energy landscape gradually approaches the original landscape, the single weight values the value 1. Still, the GRE finds improvements at (nearly) all temperature steps here, if looking at the original Hamiltonian. However, the part of the altered energy landscape in which the system is trapped in is shifted energetically upwards, as the reweighted Hamiltonian increases. (d) At small temperatures, the system is frozen in some state which is energetically lower than the state at the beginning. The changes in the energy landscape are so small, that the local or global optimum remains to be locally minimal.

After this discussion of the behavior of the algorithm, we have now a look at the quality of the results achieved with the WA technique in comparison to the GRE and SA for various computing times (12–12,000 sweeps per temperature step). Tables 1–5 show results for the five TSP instances for various computing times. The results for our SK instances are shown in Tables 6–8. For each instance, the minimum and maximum value found for some calculation time is given. Furthermore, the mean value with the error bar and the median is shown. If the optimum of an instance is reached, then the fraction of runs which reached the optimum is shown in an additional column at the right side of the table.

We find for long computing times, that SA leads to better results for the SK instances, whereas WA is superior for the TSP instances, while the GRE cannot compete with the two more elaborate techniques SA and WA at all. For very short computing times, SA remains the best algorithm for the SK instances, while the best results for the TSP instances can be either achieved with the GRE or with WA. In this regime, SA cannot compete as it allows for large deteriorations at high temperatures, such that it loses time if compared to the Greedy which only accepts improvements and trivial moves, which do not change the energy. WA, which is based on the GRE, does not lose so much time if compared to the Greedy as it is already quenched down at the beginning and then performs a transition from only slightly randomized configurations, whereas SA has to go through the complete transition.

Table 1
Results for the BEER127 instance achieved with the GRE, SA, and WA, respectively, for various computing times (sweeps): the minimum and the maximum result achieved are shown. Furthermore, the median lies close to the mean value. The Δ-value is the error bar. The right column of the table shows the fractions of runs which reached the known global optimum

Method	Sweeps	Minimum	Maximum	Median	Mean $\pm \Delta$	Opt (%)
GRE	12,000	118,986	130,706	123,123	$123,416 \pm 70$	0
	1200	118,986	130,706	123,123	$123,416 \pm 70$	0
	120	118,986	130,706	123,126	$123,430 \pm 70$	0
	12	119,367	134,971	125,272	$125,501 \pm 81$	0
SA	12,000	118,294	120,642	118,572	$118,812 \pm 19$	26.1
	1200	118,294	121,946	119,665	$119,527 \pm 23$	2.7
	120	118,294	126,702	120,514	$120,560 \pm 32$	0.1
	12	119,698	134,566	125,436	$125,737 \pm 76$	0
WA	12,000	118,294	120,883	118,740	$118,846 \pm 12$	0.4
	1200	118,294	121,758	118,818	$118,966 \pm 16$	1.8
	120	118,294	124,632	120,084	$120,155 \pm 33$	0.2
	12	119,023	132,330	124,511	$124,763 \pm 71$	0

Table 2
Results for the LIN318 instance (presented as in Table 1)

Method	Sweeps	Minimum	Maximum	Median	Mean $\pm \Delta$	Opt (%)
GRE	12,000	42,787.0	45,979.7	44,127.8	$44,151.8 \pm 16$	0
	1200	42,787.0	45,979.7	44,127.8	$44,151.8 \pm 16$	0
	120	43,300.4	46,428.6	44,640.6	$44,658.4 \pm 18$	0
	12	45,623.0	52,347.2	49,302.0	$49,268.7 \pm 30$	0
SA	12,000	42,042.5	43,188.9	42,518.3	$42,518.3 \pm 5.6$	0.2
	1200	42,275.9	43,683.4	42,824.5	$42,849.6 \pm 7.5$	0
	120	42,867.3	45,810.9	44,084.7	$44,106.8 \pm 15$	0
	12	50,048.2	57,154.1	53,129.7	$53,162.0 \pm 37$	0
WA	12,000	42,062.5	43,282.5	42,527.5	$42,516.4 \pm 5.7$	0
	1200	42,102.4	43,433.1	42,731.2	$42,747.7 \pm 7.3$	0
	120	42,863.9	45,231.5	43,805.2	$43,843.7 \pm 14$	0
	12	47,328.6	54,034.7	50,498.0	$50,549.3 \pm 34$	0

5. Relations between weight annealing and other algorithms

Of course, we are fully aware of the fact that there are also other optimization algorithms which are at least related or even rather similar to our algorithm. As

Table 3
Results for the PCB442 instance (presented as in Table 1)

Method	Sweeps	Minimum	Maximum	Median	Mean $\pm \Delta$
GRE	12,000	51,607.2	55,456.3	53,422.4	$53,436.5 \pm 19$
	1200	51,607.2	55,589.5	53,483.5	$53,501.5 \pm 19$
	120	52,827.2	57,310.7	54,765.7	$54,806.9 \pm 23$
	12	60,230.1	67,613.9	63,725.4	$63,722.4 \pm 34$
SA	120,00	50,801.8	52,233.6	51,358.5	$51,361.7 \pm 6.5$
	1200	50,949.8	53,368.2	51,914.5	$51,934.1 \pm 12$
	120	52,364.0	56,510.9	54,115.8	$54,113.3 \pm 20$
	12	60,803.3	67,211.3	64,071.1	$64,050.0 \pm 35$
WA	12,000	50,827.8	52,102.1	51,232.5	$51,248.8 \pm 5.9$
	1200	50,969.6	52,875.1	51,679.6	$51,706.4 \pm 9.4$
	120	52,446.0	56,173.3	54,185.6	$54,177.3 \pm 21$
	12	64,201.9	72,019.6	67,983.8	$68,032.9 \pm 38$

Table 4
Results for the ATT532 instance (presented as in Table 1)

Method	Sweeps	Minimum	Maximum	Median	Mean $\pm \Delta$
GRE	12,000	28,345	29,947	29,120	$29,129.6 \pm 8.0$
	1200	28,363	29,973	29,160	$29,169.2 \pm 8.1$
	120	29,122	31,051	29,997	$30,003.7 \pm 11$
	12	33,744	37,425	35,457	$35,451.3 \pm 20$
SA	12,000	27,737	28,440	28,033	$28,026.3 \pm 4.1$
	1200	27,967	29,070	28,398	$28,401.5 \pm 5.5$
	120	28,721	30,560	29,578	$29,578.6 \pm 9.2$
	12	32,909	36,927	34,905	$34,911.8 \pm 19$
WA	12,000	27,743	28,413	28,005	$28,011.8 \pm 3.7$
	1200	27,896	28,771	28,315	$28,315.4 \pm 4.9$
	120	28,656	30,528	29,429	$29,438.3 \pm 8.5$
	12	32,110	35,517	33,812	$33,795.0 \pm 17$

already mentioned, there is a technique called Search Space Smoothing which tries to get to the global optimum by removing barriers in the energy landscape [2–4]. There a smoothness control parameter α is introduced. With decreasing α, the energy landscape which is flat at the very beginning is gradually desmoothed until the landscape reaches its original shape. For each value of α, a short Greedy run is performed starting at the final configuration of the previous Greedy run. Thus, the algorithm shall "guide" the Monte Carlo process from the global optimum in the

Table 5
Results for the NRW1379 instance (presented as in Table 1)

Method	Sweeps	Minimum	Maximum	Median	Mean $\pm \Delta$
GRE	12,000	59,314.2	61,191.4	60,234.0	$60,238.2 \pm 9.4$
	1200	60,737.2	62,904.9	61,688.6	$61,686.3 \pm 12$
	120	62,674.1	65,565.1	64,029.8	$64,045.3 \pm 14$
	12	93,059.3	100,097	96,313.4	$96,327.9 \pm 36$
SA	12,000	57,519.0	58,422.1	57,991.6	$57,992.8 \pm 4.6$
	1200	59,069.2	60,473.2	59,727.5	$59,739.3 \pm 7.7$
	120	62,058.8	65,218.4	63,321.6	$63,342.8 \pm 13$
	12	88,605.8	95,196.7	92,026.3	$92,001.2 \pm 31$
WA	12,000	57,348.1	58,128.0	57,732.8	$57,739.9 \pm 4.4$
	1200	58,461.5	60,057.0	59,354.1	$59,349.2 \pm 6.8$
	120	61,759.0	64,240.2	62,868.1	$62,881.4 \pm 12$
	12	80,468.3	85,636.0	82,993.3	$82,987.2 \pm 28$

Table 6
Results for the SK100 instance (presented as in Table 1)

Method	Sweeps	Minimum	Maximum	Median	Mean $\pm \Delta$	Opt (%)
GRE	12,000	−0.720852	−0.528869	−0.665552	$-0.660706 \pm 1.1\mathrm{E}-3$	0.3
	1200	−0.720852	−0.528869	−0.665552	$-0.660706 \pm 1.1\mathrm{E}-3$	0.3
	120	−0.720852	−0.528869	−0.665552	$-0.660706 \pm 1.1\mathrm{E}-3$	0.3
	12	−0.720852	−0.528869	−0.665552	$-0.660706 \pm 1.1\mathrm{E}-3$	0.3
SA	12,000	−0.720852	−0.711584	−0.720852	$-0.720055 \pm 4.1\mathrm{E}-5$	63.3
	1200	−0.720852	−0.710520	−0.719135	$-0.719550 \pm 5.5\mathrm{E}-5$	48.9
	120	−0.720852	−0.704728	−0.719135	$-0.718358 \pm 9.1\mathrm{E}-5$	33.4
	12	−0.720852	−0.690925	−0.715525	$-0.715333 \pm 1.5\mathrm{E}-4$	17.3
WA	12,000	−0.720852	−0.687413	−0.714539	$-0.714618 \pm 1.7\mathrm{E}-4$	18.5
	1200	−0.720852	−0.686119	−0.714691	$-0.714448 \pm 1.8\mathrm{E}-4$	17.1
	120	−0.720852	−0.686119	−0.715525	$-0.714798 \pm 1.7\mathrm{E}-4$	19.6
	12	−0.720852	−0.686119	−0.714259	$-0.714372 \pm 1.7\mathrm{E}-4$	16.6

smooth surface to an at least very good solution of the original problem. If applying this algorithm to the TSP, the values of the distances are changed, thus changing indirectly the shape of the energy landscape.

The algorithm which is probably most related to our approach is the Noising or Permutation approach [5–7]: here one starts out with the thought that similar instances should exhibit similar ground states. In order to overcome barriers in the energy landscape, a series of surrogate instances is solved with the GRE, which starts

Table 7
Results for the SK200 instance (presented as in Table 1)

Method	Sweeps	Minimum	Maximum	Median	Mean ± Δ	Opt (%)
GRE	12,000	−0.728967	−0.590085	−0.666175	−0.666622 ± 7.8E − 4	0
	1200	−0.728967	−0.590085	−0.666175	−0.666622 ± 7.8E − 4	0
	120	−0.728967	−0.590085	−0.666175	−0.666622 ± 7.8E − 4	0
	12	−0.728967	−0.590085	−0.666175	−0.666622 ± 7.8E − 4	0
SA	12,000	−0.733633	−0.725348	−0.733633	−0.733110 ± 5.8E − 5	91.5
	1200	−0.733633	−0.715716	−0.733633	−0.732171 ± 9.5E − 5	78.7
	120	−0.733633	−0.715261	−0.733633	−0.729931 ± 1.4E − 4	54.2
	12	−0.733633	−0.699896	−0.725348	−0.725026 ± 2.0E − 4	21.1
WA	12,000	−0.733633	−0.687242	−0.722297	−0.720926 ± 2.4E − 4	6.9
	1200	−0.733633	−0.686185	−0.721993	−0.720892 ± 2.4E − 4	6.2
	120	−0.733633	−0.678278	−0.723195	−0.721609 ± 2.4E − 4	7.3
	12	−0.733633	−0.690352	−0.722773	−0.721200 ± 2.4E − 4	7.0

Table 8
Results for the SK400 instance (presented as in Table 1)

Method	Sweeps	Minimum	Maximum	Median	Mean ± Δ	Opt (%)
GRE	12,000	−0.725058	−0.603360	−0.671471	−0.670502 ± 5.8E − 4	0
	1200	−0.725058	−0.603360	−0.671471	−0.670502 ± 5.8E − 4	0
	120	−0.725058	−0.603360	−0.671471	−0.670502 ± 5.8E − 4	0
	12	−0.725058	−0.603360	−0.671471	−0.670502 ± 5.8E − 4	0
SA	12,000	−0.740229	−0.731108	−0.740229	−0.739917 ± 3.8E − 5	78.0
	1200	−0.740229	−0.727740	−0.740229	−0.738873 ± 8.2E − 5	60.9
	120	−0.740229	−0.722357	−0.737274	−0.736259 ± 1.2E − 4	25.8
	12	−0.740229	−0.711722	−0.730519	−0.730834 ± 1.7E − 4	3.4
WA	12,000	−0.740229	−0.693402	−0.727807	−0.727116 ± 2.5E − 4	3.8
	1200	−0.740229	−0.696959	−0.727880	−0.726952 ± 2.5E − 4	2.8
	120	−0.740229	−0.698436	−0.728222	−0.727369 ± 2.5E − 4	3.3
	12	−0.740229	−0.691855	−0.727534	−0.726588 ± 2.5E − 4	3.3

with the final configuration of the previous run, which is mapped on the corresponding configuration of the new instance.

If using a more elaborate version of WA which really considers how well the system is solved locally, then one could even go so far to say that the algorithm uses some kind of adaptive memory and therefore belongs to the class of Tabu Search algorithms if defining this class rather broadly [20].

Another algorithm which uses a related type of an adaptive memory is Guided Local Search [21,22]. This algorithm which can also be counted to the class of Tabu Search algorithms assigns in successive iterations penalties to the worst solved parts of the system, such that they shall be better solved in future iterations. This approach is also rather related to ours.

There are of course many more algorithms changing the energy landscape and making use of the history of the optimization run. For example, there are ant colony optimization algorithms which try to find the best roundtrip by making use of pheromones [23]. The Ant Lion Heuristics also changes the energy landscape in order to encourage the search process to visit good configurations and avoid bad ones [24].

Summarizing, our algorithm is related both to the class of Tabu Search algorithms and to the class of algorithms changing the energy landscape. Additionally, one has to state that these two classes are again strongly related to each other. But generally, any algorithm is related to either the Simulated Annealing algorithm or is a Genetic Algorithm or a Tabu Search algorithm or a hybrid algorithm combining these algorithms.

6. Summary and outlook

In this paper, we have applied a new approach called WA, which can be easily applied to any problem wherever a Greedy search is possible, to the TSP and to the SK-model: for problems, whose Hamiltonian can be written as a sum of Subhamiltonians of the local parts, we assign weights to the local parts, with which the Hamiltonian is reweighted. These weights can be, e.g., random variables, such that the importance of the single parts of the system for the optimization algorithm is altered in a random way. For this approach, we demonstrated that the WA technique mostly leads to better results for the TSP instances than the classic Simulated Annealing algorithm, for the short computing times we used. On the other hand, in this implementation of the WA algorithm, we have been unable to outperform Simulated Annealing for the SK instances. Of course, the results for both SA and WA can be further improved if spending even more calculation time.

More elaborate implementations of the WA algorithm, which actually take into account how well the system is solved locally, might lead to better results. We have already implemented an approach in which the local qualities of the solution are related to lower bounds achieved from the corresponding unfrustrated system, as defined in Ref. [9]. In the case of the TSP, this is the sum of the distances to the two nearest neighbors of a node. First runs showed that this technique leads to worse results than the random approach. We guess that this is due to the fact that a complex system like the TSP has sometimes to be solved in a locally non-optimum way in order to get to an overall very good or even the globally optimum solution, while the adversary reweighting approach puts more emphasis on parts which are non-optimally solved, thus ending up at an overall worse configuration. However, we believe that this approach will work significantly better in other domains. We will continue our research on this topic.

References

[1] S. Kirkpatrick, C.D. Gelatt Jr., M.P. Vecchi, Optimization by simulated annealing, Science 220 (1983) 671–680.
[2] J. Gu, X. Huang, Efficient local search space smoothing: a case study of the traveling salesman problem (TSP), IEEE Trans. Syst. Man Cybernet. 24 (1994) 728–735.
[3] J. Schneider, M. Dankesreiter, W. Fettes, I. Morgenstern, M. Schmid, J.M. Singer, Search space smoothing for combinatorial optimization problems, Physica A 243 (1997) 77–112.
[4] S.P. Coy, B.L. Golden, E.A. Wasil, A computational study of smoothing heuristics for the traveling salesman problem, Eur. J. Oper. Res. 124 (2000) 15–27.
[5] R.H. Storer, S.D. Wu, R. Vaccari, New search spaces for sequencing problems with application to job shop scheduling, Manage. Sci. 38 (1992) 1495–1509.
[6] I. Charon, O. Hudry, The noising method: a new method for combinatorial optimization, Oper. Res. Lett. 14 (1993) 133–137.
[7] B. Codenotti, G. Manzini, L. Margara, G. Resta, Perturbation: an efficient technique for the solution of very large instances of the Euclidean TSP, INFORMS J. Comput. 8 (1996) 125–133.
[8] K.H. Fischer, J.A. Hertz, Spin Glasses, Cambridge University Press, Cambridge, 1991.
[9] S. Kobe, T. Klotz, Frustration—how it can be measured, Phys. Rev. E 52 (1995) 5660–5663.
[10] M. Held, R.M. Karp, The traveling-salesman problem and minimum spanning trees, Oper. Res. 18 (1970) 1138–1162.
[11] M. Held, R.M. Karp, The traveling-salesman problem and minimum spanning trees: Part II, Math. Programming 1 (1971) 6–25.
[12] G. Elidan, M. Ninio, N. Friedman, D. Schuurmans, Data perturbation for escaping local maxima in learning, AAAI-02/IAAI-02, 2002, pp. 132–139.
[13] S. Kullback, R.A. Leibler, On information and sufficiency, Ann. Math. Stat. 22 (1951) 79–86.
[14] E.L. Lawler, J.K. Lenstra, A.H.G. Rinnoy Kan, D.B. Shmoys, The Traveling Salesman Problem, Wiley, New York, 1985.
[15] G. Reinelt, The Traveling Salesman, Springer, Berlin, Germany, 1994.
[16] S. Lin, Computer solutions of the traveling salesman problem, Bell Syst. Tech. J. 44 (1965) 2245–2269.
[17] S. Lin, B.W. Kernighan, An effective heuristic algorithm for the traveling salesman problem, Oper. Res. 21 (1973) 498–516.
[18] D. Sherrington, S. Kirkpatrick, Solvable model of a spin glass, Phys. Rev. Lett. 35 (1975) 1792.
[19] http://www.informatik.uni-heidelberg.de/groups/comopt/software/TSPLIB95.
[20] F. Glover, M. Laguna, Tabu Search, Kluwer Academic Publishers, Dordrecht, 1997.
[21] Ch. Voudouris, E. Tsang, Technical Report CSM-247, University of Essex, UK, 1995.
[22] Ch. Voudouris, E. Tsang, Guided local search and its application to the traveling salesman problem, Eur. J. Oper. Res. 113 (1999) 469–499.
[23] A. Colorni, M. Dorigo, V. Maniezzo, Distributed optimization by ant colonies, Proceedings of ECAL91—European Conference on Artificial Life, Paris, 1991, pp. 134–142.
[24] F.H. Stillinger, T.A. Weber, Nonlinear optimization simplified by hypersurface deformation, J. Stat. Phys. 52 (1988) 1429–1445.