13 Changing the Energy Landscape

13.1 Search Space Smoothing

Simulated annealing (SA) and related optimization algorithms use a temperaturelike control parameter by which the system to be optimized is led from an unordered high-energy regime to an ordered low-energy configuration. All these control parameters have in common that they allow not only for improvements but also for some deteriorations during the optimization runs, either with a certain probability or to a certain extent. Therefore, they give the Monte Carlo walker the possibility to climb over barriers in the energy landscape and by that means to escape bad local minima. One can imagine that the Monte Carlo walker is fed with sufficient energy to climb such that he can climb over the barriers and pull himself/herself out of a deep hole like Baron Münchhausen; then the additional energy is removed from him/her gradually.

A different way not to get trapped in bad local minima would be to introduce additional moves and thereby to create a larger neighborhood for each configuration such that more ways exist to walk around an energy barrier. This means in the picture of the lower-dimensional neighborhood that there are ways through mountains such that the Monte Carlo walker is able to tunnel through barriers. But increasing the number of ways through the energy landscape also increases the possibility of missing the global optimum if a fixed number of move trials is used.

But there is also a third way one could imagine, namely, to smooth the energy landscape in such a way that the Monte Carlo walker can easily jump over barriers or, better, to remove the energy barriers completely. An ideal way of smoothing would mean that the number of minima is reduced to one, such that only the global optimum is left. There are generally four ways to smooth the energy landscape:

- The cost function $\mathcal{H}$ can be changed for every state σ and can be smoothed by some function f such that each state gets a new objective value $f(\mathcal{H}(\sigma))$.
- On the other hand, one could apply the smoothing function f to the energy differences $\Delta\mathcal{H} = \mathcal{H}(\tau) - \mathcal{H}(\sigma)$ between pairs of neighboring configurations (σ, τ), so that the smoothed energy difference between these configurations is given as $f(\Delta\mathcal{H})$.

- If the Hamiltonian of the system is more complex and is given by, e. g., $\mathcal{H}(\sigma) = \sum_i \mathcal{H}_i(\sigma)$ (i. e., if the whole Hamiltonian $\mathcal{H}$ consists of various terms, e. g., penalty or correlation terms), then it could be necessary to smooth the different addends to the Hamiltonian with different smoothing functions f_i. This could lead to the Hamiltonian $\sum_i f_i(\mathcal{H}_i(\sigma))$ for the smoothed system.
- Similarly, the energy difference $\Delta\mathcal{H}$ can also be replaced by, e. g., $\sum_i f_i$ $(\Delta\mathcal{H}_i)$ if the Hamiltonian is composed of various terms.

Although this approach will smooth the energy landscape, it has been common to use the (strictly speaking) false term search space smoothing (SSS). However, the situation is not as easy as Fig. 13.1 suggests: if one were able to smooth the energy landscape in this way, one would already know it perfectly. If one knew it perfectly, one could spare the optimization run, as one would

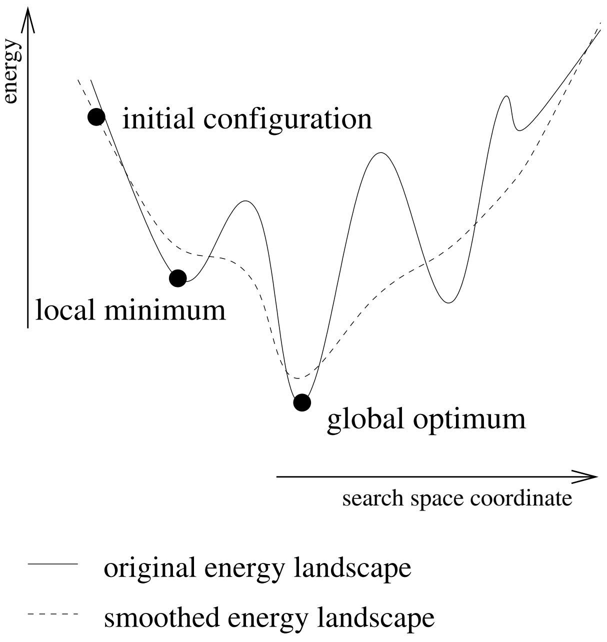

Fig. 13.1. Motivation for SSS: the graphics shows a schematic cut through the energy landscape of a simple problem. If the greedy was used as a local search algorithm in the original, i. e., unsmoothed, energy landscape, the Monte Carlo walker would get stuck in a local minimum near the starting point of his walk. Even if using more elaborate optimization algorithms like SA, TA, and GDA, one cannot be quite sure to end up at the global optimum at the end of the optimization run. However, if the energy landscape is smoothed in a way that only one minimum, identical to the global optimum, remains, the global optimum is always reached, independently of the initial configuration

know the global optimum already. In general, however, one has to deal with problems for which one does not know the energy landscape and therefore one does not know a priori where the energy barriers lie. The only way out of this dilemma is to try to smooth the energy landscape in an indirect way. This smoothing has to be done by a nonlinear formula; a linear formula f would preserve the general appearance of the energy landscape, and so it would only resize it in its height, but microscopic energy barriers are also a problem for a Monte Carlo walker who can change the height of his/her position only microscopically. An example of a nonmonotonous smoothing function, which would decrease very high energy barriers rather nicely, is the logarithm. However, applying $f(x) = \ln(x)$ to all objective values of the individual configurations would only decrease the heights of the barriers, but it would not remove them.

Therefore, one must split the Hamiltonian into many parts and apply a smoothing function f_i to each of these parts i. In the most simple way, the smoothing function is the same for all parts i. However, when smoothing the individual addends of the Hamiltonian separately by a nonlinear function like the logarithm, a further problem occurs, as the following example shows. Let σ be a local minimum and τ a neighboring configuration of σ. If these neighboring configurations only differ in one part, such that their cost function only differs in one addend, everything is fine as long as the smoothing function f is strictly monotonously increasing with the addend: let a_σ be the addend for the configuration σ and a_τ be the addend for τ. As $a_\sigma < a_\tau$, therefore $f(a_\sigma) < f(a_\tau)$. However, a difficulty occurs if neighboring configurations differ at least in two addends from the Hamiltonian: let us consider an example in which the configurations σ and τ differ in exactly two addends. Let us denote the values of these addends as a_σ and b_σ for the configuration σ and as a_τ and b_τ for the configuration τ. Of course, $a_\sigma + b_\sigma < a_\tau + b_\tau$. However, it is not necessarily the case that $f(a_\sigma) + f(b_\sigma) < f(a_\tau) + f(b_\tau)$. As an example, let us again use the logarithm as a smoothing function and choose $a_\sigma = \mathrm{e}^3$, $b_\sigma = \mathrm{e}^3$, $a_\tau = \mathrm{e}^4$, and $b_\tau = \mathrm{e}$. It is $\mathrm{e}^3 + \mathrm{e}^3 < \mathrm{e}^4 + \mathrm{e}^1$, but it is $3 + 3 > 4 + 1$. Therefore, in the smoothed landscape σ is no longer a local minimum. This type of transfer of local minima to other configurations can also happen for every other nonlinear smoothing formula. Even the global optimum might be displaced.

Therefore, one cannot simply perform an optimization run in the smoothed energy landscape, as the system will usually end up in a configuration that is a local minimum or even the global optimum in the smoothed landscape but not a minimum in the original energy landscape. The way to overcome this problem is to introduce a smoothness-control parameter α [76] by which the smoothness of the changed energy landscape can be governed: as Fig. 13.2 shows, the optimization run starts at a, for example, very large value of the smoothness-control parameter, at which the energy landscape hopefully only contains one local minimum, which is therefore the global optimum. At this

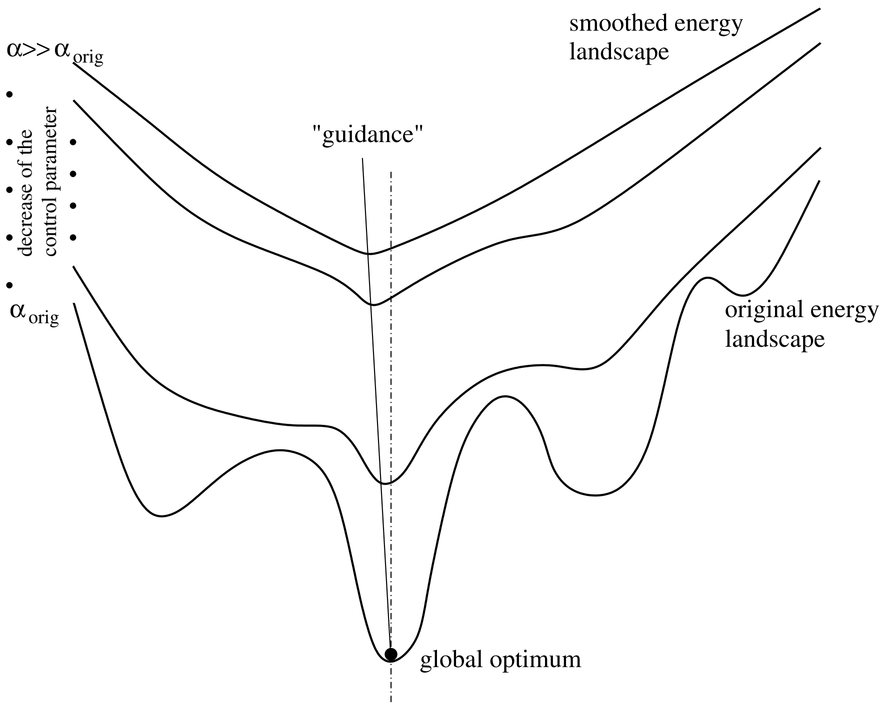
$\alpha >> \alpha_{orig}$
decrease of the control parameter
α_{orig}
smoothed energy landscape
"guidance"
original energy landscape
global optimum

Fig. 13.2. Introduction of a smoothness control parameter: the smoothness of the energy landscape is reduced step by step by reducing some smoothness control parameter α until the original energy landscape is retrieved at the end for $\alpha = \alpha_{\text{orig}}$. After the optimum solution has been found for the smoothed landscape, this solution must be adapted in each desmoothing step such that the optimum, or at least a rather good solution, is achieved at the end (graphics constructed analogously to graphics in [76])

large value of α, a greedy optimization run is performed in order to let the system jump into the global optimum of this smoothed energy landscape. After that, the smoothness-control parameter is reduced by some small amount such that the energy landscape is slightly desmoothed. This might lead to some displacement of the minimum of the energy landscape. Therefore, a new greedy optimization run must be performed in order to get from the former minimum in the energy landscape to the minimum of the new energy landscape. Then α is decreased again and a further greedy optimization run is performed in the again slightly desmoothed energy landscape. This approach is iterated until some final value of $\alpha = \alpha_{\text{orig}}$ is reached that restores the original energy landscape.

Of course, the desmoothing steps must be very small so that one may trust in a guidance effect by which the optimum solution for the smoothed landscape is transferred step by step to the optimum or at least a quasioptimum configuration in the original landscape. By changing α too drastically, one might be moved far away from the former valley and therefore be caught in some bad local minimum. There is some analogy here between SA, in which the temperature has to be decreased slowly, and this guidance effect, which occurs only if the smoothness-control parameter is decreased slightly. Otherwise, one also finds here some quenching effect.

This analogy has been stressed further: both types of optimization heuristics, SA and relatives and SSS, rely heavily on a control parameter; furthermore, a greedy run is performed on the original energy landscape at the end of the optimization run. Therefore, the question naturally arises as to whether this SSS approach could be combined with SA or other algorithms related to it. Of course it makes no sense at all performing a complete optimization run with, e. g., SA at each α step because at large values of T, the system jumps out of the local valley in which it was already caught at small α and performs a quasi-random walk (RW) such that the guidance effect of SSS is lost. However, using the greedy, i. e., the $T = 0$, case of SA and its relatives must be considered the extreme on the other side.

Probably the most natural way of combining SSS with a more elaborate acceptance function is to use the great deluge algorithm (GDA): at each step of α, the water level $\mathcal{T}$ is set to the new value of the initial configuration of the new α step [186]. Now the Monte Carlo walker is restricted to the valley where he/she is already caught, but can search for the best local minimum inside this valley as there might be more than one possible local minimum between which new energy barriers might have arisen. Therefore, at each α step a GDA optimization run is performed, with a starting value equal to the energy of the initial configuration.

The SSS approach of removing energy barriers between neighboring local optima strongly resembles a chemical catalytic process by which reactions are enabled that would not happen in the absence of the catalyst. These reactions also take place at very low temperatures. Therefore, the approach combining SA and related algorithms with SSS would be to work with a very small and constant value of the control parameter T at which the system would be (nearly) frozen in the original energy landscape. However, due to the smoothing process, the energy differences between neighboring configurations can be much smaller in the smoothed energy landscape than in the original energy landscape. Therefore, the optimization process might start in a (quasi)-RW mode at large α. For each α step, a single optimization run shall be performed. The moves shall be accepted with the acceptance probability using the small value of T. With decreasing α, the energy differences get larger such that both a SSS and a SA/TA transition are performed by reducing α. However, when working, e. g., with threshold accepting, it is not necessary to choose $\mathcal{Th} < \min\{\Delta\mathcal{H} | \Delta\mathcal{H} > 0\}$, i. e., smaller than the smallest possible energy difference in the original system. Although the system does not freeze completely, one gets roughly the same process in the simulation. The original SSS is by the way a special case, that is, the $T = 0$ case of this combined method.

13.2 Ant Lion Heuristics and Activation Relaxation Technique

The ant lion (Fig. 13.3) is an insect with an interesting behavior: in order to obtain food, which mainly consists of ants, it alters the landscape by making funnels. If an ant or another small insect comes by, it often gets trapped in one of these funnels. Additionally, the ant lion throws sand to the insect in order to increase the probability that the insect will sink into its funnel. Figure 13.4 shows such a landscape of ant lion funnels.

In summary, ant lions alter the landscape by introducing appropriate local minima in order to maximize their food. This approach is also used as an optimization algorithm, the so-called ant lion algorithm [205], which is mostly used for problems in which an objective function can only partially be written down, either because it is too difficult to quantify all constraints or because it is simply impossible. Therefore, one starts several independent optimization runs based on a restricted Hamiltonian $\mathcal{H}$, each of them leading to some solution. Then the individual solutions are analyzed. If one likes a specific solution due to fulfilling some constraints that are not represented in the Hamiltonian, then one would like to make the corresponding funnel

Fig. 13.3. An ant lion [92]

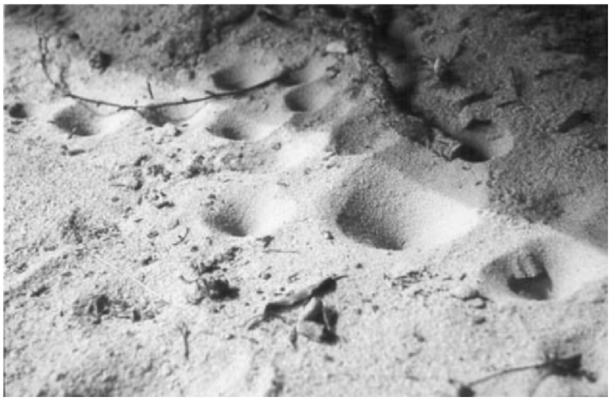
Fig. 13.4. Landscape with holes formed by ant lions [92]

in which this local minimum solution sits deeper, just like an ant lion. Similarly, if one does not like a given solution, one would like to make the funnel less deep. Thus it is not a good idea to simply change the overall objective value of a local minimum, as then the surrounding funnel would stay unchanged.

There are several ways to solve this problem: by checking all neighboring configurations of a "good" local minimum, one might be able to find out what makes this local minimum good, so that the addend to the Hamiltonian of a certain part of the configuration is changed. A second choice consists of comparing several good local minima; in this way one might find that some of them exhibit common parts. Again the values for these common parts are changed. On the other hand, when comparing "bad" solutions and their neighborhood, one will also find some structures that make these solutions bad. The corresponding addends are then increased. If now some further optimization runs are performed in this changed landscape, one will hopefully no longer end up in one of the former "bad" solutions and one might be led to "good" solutions that contain the parts one likes. However, the problem usually arises that not only are some local valleys filled up or turned into a hill, such that the system is less likely to get stuck there, and other local valleys deepened, but additional valley-hill structures occur leading to a rougher landscape than before. However, these new local minima might exhibit several nice structures and might therefore be a superposition of the individual changed addends for the "good" properties. However, it could also happen that the combination of several good properties does not lead to any good solution. This is strongly system dependent.

Another prominent example of algorithms that change the energy landscape in a constructive way is the activation relaxation technique (ART) [15, 152, 153, 208], which performs moves between local minima in the energy landscape of a continuous problem using the conjugate gradient method. These moves are accepted according to the Metropolis criterion. This method is intended to overcome the difficulty of SA to climb over barriers in the energy landscape at low temperatures.

Thus, this algorithm starts at a randomly created configuration and moves to a local minimum nearby with the Conjugate Gradient method. Other methods searching for local minima could be used instead. The idea is now to leave this local minimum and get to another local minimum by following a path of minimum energy. Therefore, the highest possible point in this path must be a saddle point of minimum order, usually of order 1. The construction of this path leading from one local minimum to another consists of three parts:

- Leaving the so-called harmonic area around the current local minimum,
- Propagating the system to a saddle point nearby, and
- Relaxing the system in a new local minimum in the energy landscape.

For the initial local minimum, a local search method is performed using the force field $\boldsymbol{F} = -\boldsymbol{\nabla}\mathcal{H}$. In order to escape the current local minimum, first a small move is performed at random. Then one continues with the force field

$$\tilde{\boldsymbol{F}} = \boldsymbol{F} - \boldsymbol{G} \tag{13.1}$$

with

$$\boldsymbol{G} = (1+\alpha)(\boldsymbol{F} \cdot \boldsymbol{\Delta r})\boldsymbol{\Delta r} \tag{13.2}$$

with

$$\boldsymbol{\Delta r} = \boldsymbol{r}(\sigma_{\text{current}}) - \boldsymbol{r}(\sigma_{\text{last local minimum}}) \, . \tag{13.3}$$

Thus, that part of the force field leading to the last local minimum is removed. The control parameter α determines how fast the system propagates toward the saddle point. There are several ways to develop adaptive formulas for α, e. g.,

$$\alpha = -\frac{\alpha_0}{1 + \Delta r} \, , \tag{13.4}$$

with $\alpha_0 = 0.15$ [15, 208]. Thus, the more the system has already moved from the optimum, the faster it can move. One performs conjugate gradient steps iteratively with the force field $\tilde{\boldsymbol{F}}$, which changes after each step until $\boldsymbol{F}$ decreases again. Due to the finite step size of the moves, one will not touch the saddle point accurately. But one can check whether or not the local valley of the last local minimum has already been left: if $\boldsymbol{F} \cdot \boldsymbol{\Delta r} < 0$, then one is still in the local valley and thus in the attractor region of the former local minimum. Thus, one verifies with the condition $\boldsymbol{F} \cdot \boldsymbol{\Delta r} > 0$ that the saddle point has already been trespassed. In order to ensure that the system does not fall back in the previous local minimum, one adds some conjugate gradient steps with the modified force field and then relaxes the system in a new local minimum by applying the conjugate gradient method based on the original force field $\boldsymbol{F}$. The new local minimum is then accepted with the Metropolis criterion

$$p(\sigma_{\text{old local minimum}} \to \sigma_{\text{new local minimum}}) = \min\{1, \exp(-\Delta\mathcal{H}/(k_{\text{B}}T))\} \, , \tag{13.5}$$

with $\Delta\mathcal{H}$ being the energy difference between these two local minima. If this move is accepted, one performs a new move trial starting from the new local minimum. Otherwise, one tries a new move starting from the previous local minimum.

Please note that individual local minima are not found in a process with moves chosen at random but with an elaborate construction heuristic. Just as with ruin & recreate, the individual states do not occur according to their Boltzmann weights. Thus, although the Metropolis criterion is applied, no thermal euqilibrium is reached. Furthermore, when investigating the saddle points more closely, one finds that many of them are not first order but of a higher order. Up to half of the assumed saddle points are in reality local minima.

There are various ways to apply this method to combinatorial optimization problems with a discrete energy landscape. Starting at a random initial configuration, one would first perform a greedy run or the steepest descent method in order to reach a local minimum. Then one performs a few moves in the RW mode in order to leave the local minimum and the area around it. Then one switches back to the greedy mode and only accepts moves that do not increase the energy and do not increase the overlap to the previous local minimum, in order not to end up at the old local minimum again. This restriction is then removed again after a few moves so that the greedy search can lead to a true local minimum. Of course, one will do this in accordance with how one defines a Hamiltonian

$$\tilde{\mathcal{H}}(\sigma) = \mathcal{H}(\sigma) + \lambda \mathcal{O}(\sigma, \sigma_{\text{old local minimum}}) \,, \qquad (13.6)$$

with $\mathcal{O}(\sigma, \sigma_{\text{old local minimum}})$ being the overlap between the current configuration σ and the old local minimum. If the greedy method then leads too often to the previous local minimum, then the number of moves in the RW mode must be enlarged.

This ART algorithm can be extended for both continuous and discrete problems in such a way that several former local minima get part of the force field or of the Hamiltonian in order to avoid them all.

13.3 Noising or Permutation of System Parts

Noising or perturbation heuristics also belong to the class of energy-landscape-changing algorithms but are derived from a completely different approach [206, 34, 39]: usually, one expects that if one changes an instance of a given optimization problem slightly, then the optimum solution for this changed instance should be roughly the same as for the original instance. Analogously, local minima in the energy landscape will either be the same or only slightly displaced. The extent of these changes depending on the size of the changes in the input parameters can be measured by observables like the sensitivity that are used in the analysis of computational problems in order to find out how stable a solution is if some input parameters of the problem instance are changed slightly.

From this point of view the idea arises to move from the given instance to a slightly changed instance of the proposed problem. The outline is rather similar to the SSS techniques but with a different starting point: here one starts with the original problem instance, and therefore in the original energy landscape, and performs an optimization run leading to a local optimum solution. Then the problem instance is slightly perturbed such that one now transfers the local optimum solution for the original problem to a configuration of the perturbed problem instance. There, this configuration is not necessarily a local optimum solution, so that an optimization run is performed with the greedy algorithm in order to reach a local optimum of the perturbed problem instance. Now the various approaches differ:

- One can either return to the original problem instance by transferring the solution of the perturbed instance to a configuration of the original instance. There, the current configuration might again not be locally optimal, so that a greedy optimization run is performed. After that, one either returns to the former perturbed instance or creates another perturbed instance. This scheme is iterated several times.
- On the other hand, one can also jump directly to another instance that was also created by slightly perturbing the original instance. Therefore, the local optimum solution of the first perturbed instance is transferred to a configuration of the second perturbed instance. This configuration serves as a starting point for a greedy optimization run, which is performed on this second performed instance. Also, at this point the scheme is iterated several times.

In the end, one usually returns to the original instance and performs a final greedy optimization run there. Various implementations of these algorithms furthermore differ in the amount of noise added to the original instance. Usually, this amount is decreased in some way in order only to consider instances more similar to the original instance. Note that this amount serves therefore as a control parameter like the smoothness-control parameter α of SSS or the temperature T of SA. Note again the close relation between these control parameters.

There are usually several ways of perturbing a problem instance, depending on the underlying problem: one might think of, e. g., displacing some parts of the problem, changing the size of the parts, and removing some parts of the instance or, on the other hand, introducing further parts in the instance.

13.4 Weight Annealing

Closely related to these permutation heuristics is another approach called weight annealing [156]: here one assigns weights w_i to the different parts of the problem. These weights are introduced into the cost function $\mathcal{H}$ of the problem, leading to a cost function $\mathcal{H}_{\boldsymbol{w}}$. In this way, the impact of the individual parts on the cost function is reweighted by the weight vector $\boldsymbol{w}$. There are various reasons for introducing such weights:

- One wants to represent the importance of some parts for the whole system. According to some subjective perception or objective a priori considerations, it might be that some parts have to be solved in a better way than other parts of the problem.
- After having already performed several optimization runs on the specific problem instance, one finds out that some parts are not solved or not solved very well. Therefore, one wants to give some further weight to these parts so that they are considered by the optimization process in a better way. These approaches can be summarized in the term a posteriori approaches.

- The easiest way, however, is a random approach, which simply assigns random weights to the individual parts of the system. This approach is essentially identical to the perturbation method of the last section.

By adding or multiplying this further weight, one wants to favor configurations in which the corresponding part of the problem is solved (rather) well. As the optimization process looks for the optimum in the energy landscape, it is then more probable that the final configuration will contain a better-solved part.

However, from this picture we see already the dangers of this approach: if a weight for a certain part is rather large compared to other weights, then the optimization process might concentrate too much on solving this part correctly, thus leading to an overall worse solution. The energy landscape might in this case look like a rather flat landscape, but containing a canyon system. Therefore, one must always check the final configuration to see whether it is good for the original cost function $\mathcal{H}$.

Instead of assigning weights only once, performing an optimization run based on the weighted Hamiltonian $\mathcal{H}_{\boldsymbol{w}}$, and printing the result, one could also think of changing the weights in various steps: one might either start at weights according to one's own a priori considerations or with each weight $w_i = 1$. After the first optimization run, one checks which parts of the system were solved rather badly. According to the importance of the individual parts and of the difference between the desired local solution and the current one, one changes the weights, so that those parts that are to be solved in a better way get a relatively larger weight. This approach is iterated several times or until the weights no longer change. According to the underlying optimization algorithm, one starts either with the final configuration of the previous iteration or with a random solution in the current reweighting iteration.

This approach can be used in different ways:

- It can be used like the perturbation approach of the last section. Therefore, the weights should converge to 1 for all parts.
- One can use it to solve some parts of a problem better than they are solved when using the original cost function.
- One can use it to find out which parts of a system are more or less important than other parts. Therefore, one is interested also in the final values of the weights in order to find a good cost function for the considered problem.