Simulated annealing

Matthias Springer
Hasso-Plattner-Institut an der Universität Potsdam
matthias.springer@student.hpi.uni-potsdam.de

December 6, 2011

Abstract

Metaheuristics are general problem solving algorithms which abstract from the actual problem description. Therefore they can be easily applied to many optimization problems. Simulated annealing is a simple and fast metaheuristic with an analogy to metal processing. As metal particles generate a solid and regular structure when cooling slowing simulated annealing seeks a low-energy solution avoiding local optima.

1 $\mathcal{NP}$-complete problems in computer science

Since Stephen Cook's paper *The complexity of theorem proving procedures* from 1971 computer scientists know about a special class of problems which are very difficult to solve. These problems were later called $\mathcal{NP}$-complete problems. Richard Karp showed in 1972 that there are some very important and often needed problems among them, such as the *0-1 integer programming* problem. Many problems in the area of business studies and operations research can be reduced to a combinatorial optimization problem and then be solved by an integer programming algorithm.

Capacity planning is such a problem. Imagine a big car company which has several production sites all over the world. A car typically consists of about 10000 components. The difficulty is to decide which component to produce at which site. It might be cheap to produce components in countries which are rich in raw materials or countries having low unit labor costs but some production processes might only be feasible in industrialized nations. Furthermore the finished cars need to be transported to the countries where they can be sold. By minimizing expenses and maximizing profits good solutions can be found. Such a problem can be expressed mathematically by a set of integer inequalities. This mathematical representation, also called a constraint optimization problem, is a combinatorial optimization problem and can be solved by linear integer programming, which is a generalization of 0-1 integer programming. Thus the capacity planning problem is $\mathcal{NP}$-complete.

Unfortunately $\mathcal{NP}$-complete problems can't be solved efficiently so far and they will most likely never be solved efficiently. In this context *efficiently* denotes polynomial computation complexity.

2 Heuristics

Although an optimal solution to an $\mathcal{NP}$-complete problem can't be computed efficiently an approximation of the solution can often be found quickly. Algorithms that solve a problem only approximately are called heuristics.

Consider the travelling salesman problem (TSP), which is an $\mathcal{NP}$-complete problem. Given a list of cities (vertices) and distances (edges with costs) between the cities the task is to find the shortest route such that every city is visited and the final city is the origin city. The TSP can be approximated using a minimal spanning tree, where only those edges between the cities are considered that minimize the

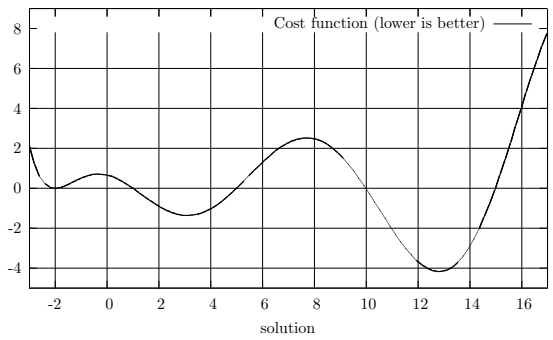

Figure 1: A cost function with local optima

total costs of the graph. The graph must be still connected, of course, which means that every vertex must be reachable from any other vertex. By traversing the minimal spanning tree a solution for the TSP can be found which is never worse than twice the total cost of the optimal TSP route.

3 Metaheuristics for optimization problems

While it can be very difficult to find good heuristics metaheuristics are general problem solving techniques which abstract from the definition of the actual problem. Metaheuristics can only be applied if the search-space is discrete, which is true for combinatorial optimization problems because the number of possible solutions is finite.

Most metaheuristics generate new solutions on the basis of a given solution with some randomness. The new solution is evaluated by a cost function and then either accepted or rejected. At first glance this might look like a greedy algorithm.

A greedy algorithm tries to improve a given solution by making elementary changes to it. After comparing the old solution with the new solution using the cost function the new solution is accepted if and only if the new solution has lower costs than the old solution.

Most problems have a cost function that could look like figure 1. The graph contains local optima at $x_1 = -2$, $x_2 = 3$ and a global optimum at $x_3 = 13$. Let's suppose we're given a solution at $x = 0$. By making elementary changes we can generate solutions which are at close distance to the left or to the right (neighbors). In this situation we can randomly decide which direction to take because in both directions the solutions have a lower cost function value. So let's decide to turn right. After that we'll always turn right since the solution would be worse when turning left. This process continues until we reach the local optimum x_2. Now we would neither turn left nor turn right since the solutions degrade in both directions. The greedy algorithmus would stop at the local optimum. A metaheuristic can avoid this problem by accepting worse solutions with a given probability.

4 Simulated annealing

4.1 Metal processing

Simulated annealing is a metaheuristic derived from metal processing. For modifying the state of a meterial the temperature can be changed (figure 2). In metal processing steel is heated before further processing. The particles (atoms) inside the material are now in a disordered state with high energy and can move around more or less freely (viscous state). In the end the metal is cooled slowly. If the temperature decreases too fast (quenching) the material is brittle, unstable and of low quality because the particles generate an amorphous solid state. They simply don't have enough time to build a stable low-energy state. The amorphous solid state corresponds to a local optimum because a better state could be reached when cooling slowly.

4.2 Adaption to computer science

The process of simulated annealing can be applied to optimiziation problems in computer science. A solution is considered poor if the cost function has a high value, in terms of simulated annealing called a solution with high energy. The following pseudocode describes the principle of simulated annealing.

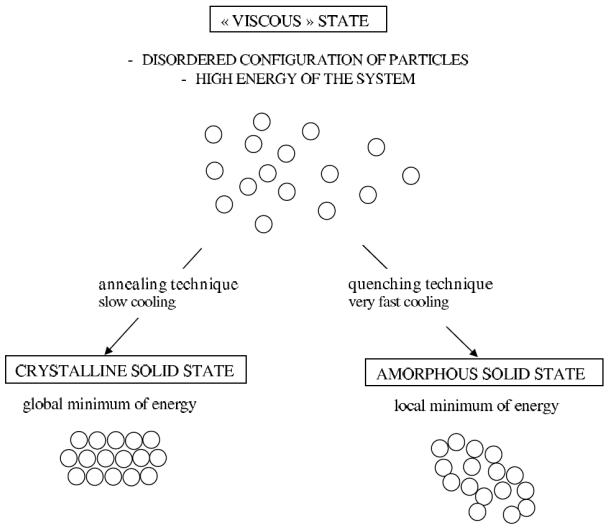

Figure 2: Simulated annealing in metal processing (Source: [1])

$S \leftarrow$ InitialSolution()
$E \leftarrow$ Costs(S)
$T \leftarrow$ InitialTemperature(S)

while $T \geq 0$ **do**
  for $i = 1$ **to** Iterations(S) **do**
    $S_{old} \leftarrow S$
    $S \leftarrow$ Neighbor(S)
    $E_{new} \leftarrow Costs(S)$
    $\Delta E \leftarrow E_{new} - E$

    if $\Delta E > 0$ **then**
      $r \leftarrow$ Random(0, 1)
      if $r < e^{\frac{-\Delta E}{T}}$ **then**
        $S \leftarrow S_{old}$
      end if
    end if
  end for
  $T \leftarrow T - 1$
end while
return S

The variable S contains the current solution and needs to be initialized before the algorithm can begin. In most cases a random but valid solution is generated but it's also possible to start with a good solution generated by another algorithm. That solution will be improved piece by piece.

E contains the costs of the current function and T contains the current temperature. At the beginning the temperature is initialized to a high value which can depend on the size of the problem. Big problems sometimes require a higher starting temperature, which leads to a higher number of computation steps.

At the each cycle of the while-loop the temperate is decreased by one unit until the temperature is zero. Within each cycle a specific number of iterations take place, this number also depends on the size of the problem. Big problems require a higher number of iterations since there are more changes needed to generate a good solution out of an initial random solution.

At first the current solution S is saved and a neighbor solution is generated. This can be done by making elementary modifications to the existing solution. An elementary change to a TSP solution could be the permutation of two cities. It's important that an elementary change doesn't generate invalid solutions since an invalid solution's costs aren't defined. Alternatively they could be considered infinite. For instance a TSP tour which doesn't visit at least one city would be invalid.

Subsequently the costs of the new solution are computed. ΔE contains the difference between the costs of the new and the old solution. If the new solution is worse than the original solution ΔE will be greater than zero. Otherwise ΔE will be lower or equal to zero. The algorithm now accepts or rejects the new solution with some randomness. If the solution was improved it's accepted definitely. Otherwise the solution is only accepted with a probability of $e^{\frac{-\Delta E}{T}}$.

Figure 3 show the acceptance function $e^{\frac{-\Delta E}{T}}$ at different temperature levels. The vertical axis indicates the probability of accepting a solution with a given value of ΔE. A solution with a probablity higher than 1 is always accepted. At a high temperature almost every solution is accepted, even if the solution is much worse than the solution before. The lower the temperature is the more unlikely is it that a worse

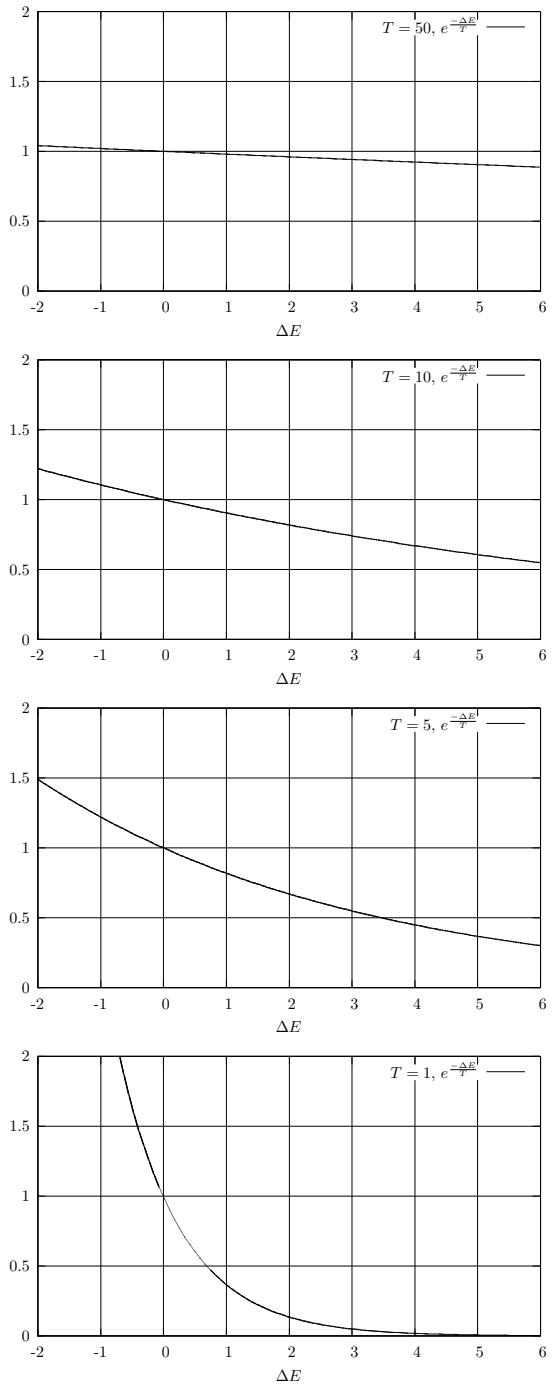

Figure 3: Acceptance function at different temperature levels

solution is accepted. At a very low temperature simulated annealing degenerates into a greedy algorithm, which only accepts better solutions.

By accepting worse solutions simulated annealing can overcome local optima and probably even find a global optimum. In figure 1 simulated annealing might accept the right neighbor of $x_2 = 3$ at a high temperature and overcome the hill at $x = 6.8$ until the global optimum at $x_3 = 13$ is found.

4.3 Adavantages and disadvantages

Simulated annealing is a metaheuristic and as with most metaheuristics the final solution can be arbitrarily poor. Furthermore it is nondeterministic, with the consequence that it can generate different solutions when running it multiple times. It was, however, shown that simulated annealing in theory always finds the global optimum if the temperature is lowered infinitely slow. Thus the algorithm's performance can be increased by making the steps of temperature reduction slower.

In contrast to algorithms which always find the optimal solution simulated annealing has a polynomial computational complexity. The runtime complexity is $\mathcal{O}(T \cdot I \cdot (C + G))$ where T is the initial temperature and I is the number of iterations per temperature step. I usually grows linearly with the problem size. C is the number of computation steps required for evaluating the cost function. Together with G, the number of computation steps required for generating a new neighbor solution, these two steps are the most performance critical steps. If new solutions can be generated and evaluated very quickly the algorithm can test a lot of possible solution in reasonable time.

4.4 Variations

There is a big number of possible variations of the simulated annealing algorithm. In this chapter two basic variations will be discussed.

Simulated annealing performs elementary changes to generate a neighbor solution. The algorithm can be changed such that bigger changes are made at a

higher temperature. When solving the TSP the algorithm could for instance permutate more than only two cities at a high temperature. This variation is based on the fact that particles with a high temperature, and thus with a high energy level, can move more quickly inside the material and overcome a bigger distance.

Simulated annealing as described in this paper always terminates after a specific number of elementary changes since every temperature step consists of a fixed number of iterations. The problem is that in most cases it's unknown if a better solution exists. Therefore the number of iterations per temperature step can be made variable. The algorithm could be changed such that the temperature is only decreased if no better solution was found after a fixed number of elementary changes in the current temperature step. The algorithm is still guaranteed to terminate because eventually the algorithm either doesn't find a better solution after a fixed number of steps or the algorithm reaches a global optimum which can't be improved anymore. However, this can take a very long time. In the worst case each neighbor improves the current solution only by one unit[1]. Hence the computational complexity shown before doesn't apply anymore.

5 Other metaheuristics

There are many other metaheuristics besides simulated annealing. Genetic algorithms try to improve a set of solutions by combining individual solutions (crossover) and performing small random changes (mutation) afterwards. As simulated annealing has an analogy in metal processing, genetic algorithms rely on Darwin's *survival of the fittest*. Genetic algorithms often generate better solutions than simulated annealing but it takes a longer time until the first suitable solution is calculated. Another interesting metaheuristic is ant colony optimization, which is often used to find paths in graphs. Ant colony optization algorithms simulate the behavior of ants which spread pheromones on their way when searching for food. Other ants follow paths with a high pheromone amount more likely. There are very efficient implementations of ant colony optimization algorithms for solving the TSP. However, it's difficult to choose the parameters wisely since there are much more parameters than in simulated annealing.

[1] There's only a finite number of cost function values since the number of possible solutions is finite in combinatorial optimization.

References

[1] Dréo, Pétrowski, Siarry, Taillard. *Metaheristics for Hard Optimization*. Springer, 2005.

[2] Vöcking, Alt, Dietzfelbinger, Reischuk, Scheideler, Vollmer, Wagner. *Taschenbuch der Algorithmen*. eXamen.press, Springer, 2008.

[3] A. K. Dewdney. *The New Turing Omnibus*. 66 excursions in computer science. Henry Holt, 2001.

[4] Cook-Levin theorem — Wikipedia, The Free Encyclopedia, 2011. [Online; accessed November 27, 2011] `http://en.wikipedia.org/w/index.php?title=Cook%E2%80%93Levin_theorem&oldid=455727368`.

[5] Karp's 21 NP-complete problems — Wikipedia, The Free Encyclopedia, 2011. [Online; accessed November 27, 2011] `http://en.wikipedia.org/w/index.php?title=Karp%27s_21_NP-complete_problems&oldid=455379220`.

[6] Survival of the fittest — Wikipedia, The Free Encyclopedia, 2011. [Online; accessed December 04, 2011] `http://en.wikipedia.org/w/index.php?title=Survival_of_the_fittest&oldid=462467346`.