Simulated Annealing Overview

Zak Varty

March 2017

Annealing is a technique initially used in metallurgy, the branch of materials science concerned with metals and their alloys. The technique consists of melting a material and then very slowly cooling it until it solidifies, ensuring that the atomic structure is a regular crystal lattice throughout the material. If the cooling is not done slowly enough then the material will form a glass, where the structure has many imperfections. Further, if the material is cooled very quickly then many separate crystal structures form, leading to an arrangement which is far more disordered. As the atomic structure becomes more ordered, the total energy stored in the material lowers and it becomes more stable.

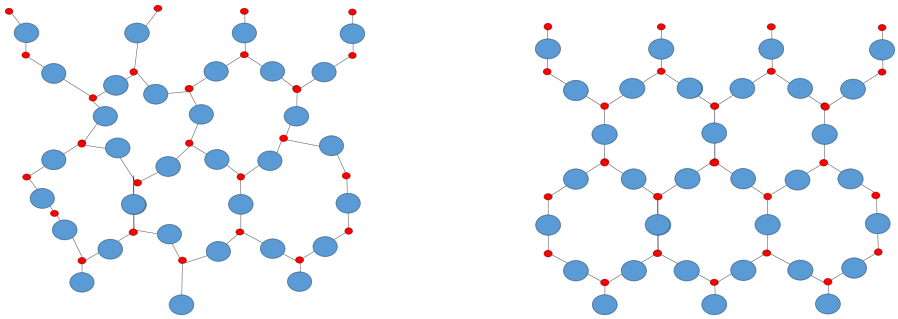

Figure 1: Left: Imperfect molecular structure of a glass. Right: Regular lattice atomic structure of a crystal.

Simulated annealing exploits the idea of annealing to go about finding lowest cost solutions for problems concerning arrangement or ordering of some collection, these are called combinatorial optimisation problems. An example of such a problem would be the travelling salesperson problem. The task in the travelling salesperson problem is to find the shortest route visiting each of a group of towns once and then returning home at the end of the day. In many such problems it is not necessary to spend a huge amount of time find the absolute best solution, but to find one which comes close to being the best in a short time. This is a task ideally suited to simulated annealing, which is not guaranteed to find the best solution in a finite amount of time but does find good solutions quickly in comparison to other methods.

Simulated annealing finds these solutions by starting off at a high 'temperature' and initially accepts all suggested arrangements, but as the temperature decreases becomes more discerning and the proportion of shifts which worsen the solution is reduced. If the cooling is done correctly this finds an arrangement which is equivalent or close to the crystal structure, finding a solution which is either the cheapest or else very close.

The speed and high quality of solutions provided by simulated annealing mean that it is often used in practice. It is used for tasks such as scheduling classes so that students minimise the walk between rooms, or the design of computer chips so that interference between components is minimised.

This report explores how we might go about using simulated annealing in practice for combinatorial type problems. From there, we consider how it might be extended beyond combinatorial problems, to problems where the components of the solution can take on any value. The limitations of simulated annealing and how these might be worked around are explored, particularly focusing the large amounts of calculation required. How this time taken to perform these calculations can be reduced is considered by splitting the technique into components which can be executed separately and simultaneously, using a parallel computing system.

Simulated Annealing.

Zak Varty

March 2017

1 Background and Motivation

Heuristic algorithms locate solutions to a problem that are considered to be 'good enough'. These are solutions that come close to being optimal in some sense of closeness, and which indeed may actually be exactly optimal. However, assuring this optimality is not of concern in a heuristic approach, heuristics trade guaranteed optimality for computational efficiency.

Metaheuristic algorithms extend the idea of heuristics by combining one or more heuristic procedures, employing some strategy to do so. The name *meta*-heuristic arises because the strategy of when to use each heuristic procedure is itself a heuristic. The procedures used within a metaheuristic can vary in complexity, from simple operations to a whole other metaheuristic, but these operations are considered as 'black-box' operations by the metaheuristic employing them. For example if a sorting is required as one of the procedures of a metaheuristic, then this procedure could be anything from a simple bubble sort to a more complicated merge sort, but these would all yield an equivalent metaheuristic because it is only concerned with the inputs and outputs of the procedures [1].

Examples of metaheuristics include; iterated local search algorithms, genetic algorithms, simulated annealing, ant colony optimisation, tabu search and particle swarm optimisation. A concise description, motivation and implementation of each of these metaheuristics is given by Brownlee [1]. Simulated annealing is the third most popular metaheuristic technique (by number of publications with titles including the technique on Google Scholar), behind genetic algorithms and particle swarm optimisation but still ahead of the other well known metaheuristics. The popularity of simulated annealing has been attributed to its wide applicability, ease of understanding to the nonspecialist, and high quality solutions [10] [9].

The simulated annealing metaheuristic is inspired by statistical mechanics and the metallurgy technique of annealing. From statistical mechanics, simulated annealing draws the idea that in the limit as temperature decreases, a substance either remains fluid or solidifies to form a crystalline solid or a glass; which of these it forms depends both on the substance and the rate at which it is cooled. The state of lowest total energy is called the ground state of a substance, which occurs when a crystalline structure is achieved at the limiting temperature. The ground state can only be reached by first melting the substance and then very slowly cooling it, particularly around the freezing point, so that a global crystalline structure is formed around a single initial crystal. This process of melting and then slowly cooling is called annealing. If this annealing is not done slowly enough then defects in the crystal structure form, leading to a higher total energy than the ground state. If the cooling is done rapidly then the substance is said to have been quenched, and will form a glass. Quenching results in no crystalline structure, forming only locally optimal structures and a total energy far greater than the ground state [5]. Annealing to reach the lowest energy state is inspired by and is used in practice during metallurgy. Here annealing ensures that the crystal size in founded metals is maximised, producing the strongest and most durable product.

In the seminal paper on simulated annealing, Kirkpatrick et al. [5] combine these ideas with a Metropolis-type algorithm to create the methodology for simulated annealing. As opposed to a greedy approach only accepting downward moves, a Metropolis approach accepts an upward

move with a given probability, which avoids becoming forever stuck in a local optimum. In simulated annealing this probability is dependent on the current 'temperature' of the problem. By having a high acceptance probability of upward moves at high temperatures, simulated annealing begins by freely jumping between potential solutions. As the temperature is slowly reduced, the acceptance probability of an upward move decreases and the metaheuristic focuses more and more on improving solutions until at freezing point it is left with a low energy state which is at or close to the ground state.

Simulated annealing is predominantly used for combinatorial optimisation, an area which covers problems such as the travelling salesperson problem, timetable creation, and the quadratic assignment problem.

2 Simulated Annealing for Combinatorial Optimisation

A combinatorial optimisation problem is composed of a configuration (solution) space and a cost function. The configuration space is the set of all possible configurations of the system to be optimised, $S = \{x | x = (x_1, x_2, \ldots, x_m)\}$, where the dimension of the problem space, m, is the number of components in the system. The associated cost function $C : S \to \mathbb{R}$ assigns a real number to each configuration and we seek one of potentially many lowest cost configurations, that is we want to find $x^* \in S$ such that $\forall x \in S,\ C(x^*) \leq C(x)$ [9].

In order to use simulated annealing to solve a combinatorial optimisation problem, we require more than just the solution space S and the cost function $C(\cdot)$. We also require a generator of configuration changes, which given the current configuration of the system will suggest another 'neighbouring' configuration to which the system may move. Also required are control parameter T and an annealing schedule detailing how T is to be reduced. In the analogy to statistical mechanics, the configuration is the current state of the substance, the cost is the total energy and the generator is the states to which the substance could move in the next short time period - the 'neighbouring' states. The control parameter is analogous to temperature and the annealing schedule describes the speed and manner of cooling.

The general combinatorial problem is approached by inhomogeneous annealing as follows. Start by generating an initial solution, either at random or by some informed method, and set the initial temperature so that a high proportion of moves are accepted so that the system is 'melted'. Next, propose a change to the configuration and compare the cost of the two configurations. If the proposed cost is lower then accept this move, otherwise accept the move with a probability based on the current temperature. Lower the temperature slightly and return to proposing a new configuration until the system temperature reaches zero and 'freezes'. This procedure is detailed using pseudo-code in Algorithm 1.

Van Laarhoven and Aarts [8] detail the proof of convergence of the simulated annealing algorithm for an inhomogeneous annealing schedule, the case where the temperature is decreased after each change of configuration, as well as for a homogeneous schedule as considered in Section 3.1. This is encouraging in that if a simulated annealing routine were left to run indefinitely then it is guaranteed to find the configuration of lowest cost, despite being a metaheuristic method. It is however disheartening that the computation required to be certain of achieving an optimal configuration might be greater than that required to enumerate the entire configuration space.

To make clear the use and adaptability of simulated annealing its use in a four colouring problem is explored in the following section.

2.1 Four Colouring of a Planar Graph

Every planar graph can be coloured using only four colours such that no two adjacent nodes have the same colouring. This is useful in that it provides a simple combinatorial optimisation problem where the optimal solution is known for demonstration and assessment of the simulated annealing. The cost function for the graph colouring problem is given by the number of adjacent nodes with the same colouring. A random initial colouring is used where the colours are coded

Algorithm 1 Simulated Annealing Pseudocode

procedure SIMANNEAL($m, iter_{max}, T$)
  $x_{curr} \leftarrow InitialConfig(m)$ ▷ Select some $x \in S$
  $x_{best} \leftarrow x_{curr}$
  for $i = 1$ **to** $iter_{max}$ **do**
    $x_{prop} \leftarrow NeighbourConfig(x_{curr})$ ▷ Propose some neighbour configuration
    $temp_{curr} \leftarrow CalcTemp(i, T)$ ▷ Anneal system
    if $Cost(x_{prop}) \leq Cost(x_{current})$ **then**
      $x_{curr} \leftarrow x_{prop}$
      if $Cost(x_{prop}) \leq Cost(x_{best})$ **then**
        $x_{best} \leftarrow x_{prop}$
      end if
    else if $\exp\left\{\frac{Cost(x_{curr}) - Cost(x_{prop})}{temp_{curr}}\right\} > Random(0, 1)$ **then** ▷ Accept upward step?
      $x_{curr} \leftarrow x_{prop}$
    end if
  end for
  return x_{best} , $Cost(x_{best})$ ▷ Best configuration and associated cost
end procedure

1, 2, 3, and 4. The solution space for the problem is therefore $\{1, 2, 3, 4\}^m$ where m is the number of nodes in the graph. The annealing schedule is taken to be linear, decreasing from initial temperature T to 0, over n iterations. To propose new configurations, a node is randomly selected and its colour randomly assigned to one of the three remaining colours.

This four colouring specific metaheuristic was applied to the 49 mainland states of the USA, with states being neighbours if they share a border. An initial temperature of 5.5 was used, leading to an initial acceptance probability of 81.6%. This initial temperature was based on the review by Wright [10], suggesting a high recommendation that the initial acceptance should be greater that 80%, to reduce dependency of the output on the initial state. Figure 2 shows two paths of the metaheuristic, demonstrating both non-optimality and optimality of solution, which can be seen clearly from the enlargements of the last 1000 iterations of each path. The plots of the full paths show that both begin by freely moving around the configuration space and then begin to discern better solutions, the cost lowering with the temperature. Both paths can also be seen to jump out of local minima, for example around 1500 iterations into the first path.

3 Improvements to Simulated Annealing

3.1 Extending to Continuous Optimisation

As well as being used in combinatorial optimisation, the simulated annealing metaheuristic can be extended to the minimisation of continuous functions. Corana et al. [3] extended the metaheuristic to a finite region, finding the global minimum or a nearby point for continuous multivariate functions which are highly multimodal. In such cases, standard optimisation routines fail by becoming stuck in one of the many local minima and taking many initialisations over the search space for these routines fails when the dimensionality of the problem becomes even moderately large.

The continuous extension of simulated annealing by Corana et al. [3] corresponds to a homogeneous annealing schedule, where the temperature is held constant until the system reaches equilibrium and is then reduced slightly. This is in contrast to the inhomogeneous annealing schedule in Section 2 where temperature is reduced after each iteration.

Corana et al. [3] initialise a random initial location and steps are proposed at each iteration in only one of the n dimensions, which will be referred to as the h^{th} dimension. The steps are proposed uniformly at random on an interval of length $2v_{m_h}$ centred at the current location $\boldsymbol{x}_i$,

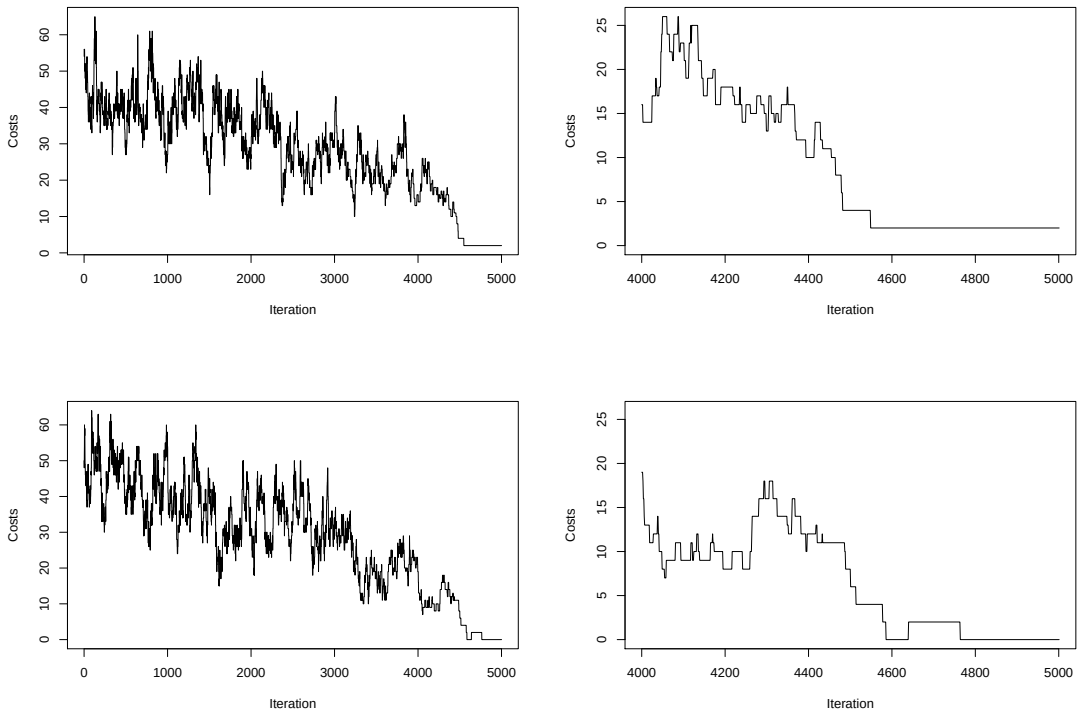

Figure 2: Two simulated annealing paths for a 4 colouring of the states of the USA, path 1 above and path 2 below, with enlargements of final the 1000 iterations in the right-hand plots.

giving a proposed point $\boldsymbol{x}' = \boldsymbol{x} + U v_{m_h} \boldsymbol{e}_h$ where $\boldsymbol{e}_h$ is a unit vector in the h^{th} coordinate direction and $U \sim \text{Unif}\,[-1, 1]$. If the proposed point is outside of the finite optimisation region then steps are proposed until one falls within the region. This step is then accepted if the function evaluated at the proposed point is less than at the current point, or else with a probability proportional to the current temperature as in Algorithm 1. The value of $\boldsymbol{v}_m$, and hence the maximum length of the proposed jumps, is calculated after a fixed number N_s of moves so that acceptance probability is held constant at each temperature and after each N_t updates of $\boldsymbol{v}_m$ the temperature is reduced according to some annealing schedule.

This annealing metaheuristic was compared with the Nelder-Mead Simplex Method and an Adaptive Random Search routine on a range of test functions to be minimised [3]. In all cases the simulated annealing algorithm was successful in identifying the global minimum, which the comparison methods failed to do. Simulated annealing was also more efficient in its search, in terms of number of function evaluations, for all cases apart from the 2 dimensional Rosenbrock function, often used as a test function because of its multimodality, where the simplex method was more efficient. Simulated annealing was the most efficient but remained a computationally costly procedure, with the number of evaluations growing linearly with number of dimensions. This cost is less prohibitive than the other techniques, particularly as the cost is dependent only on the starting temperature meaning that computation time is known in advance, which is a benefit of the metaheuristic. Further methods of extending simulated annealing to continuous optimisation are detailed in Van Laarhoven and Aarts [8].

3.2 Parallelising Simulated Annealing

One of the main issues with simulated annealing for both combinatorial and continuous function optimisation is the high computation time of finding a solution which is close to or exactly optimal. This problem arises from proposing and evaluating function values only to then reject them and can be somewhat mitigated by parallelising the metaheuristic. Many problem specific

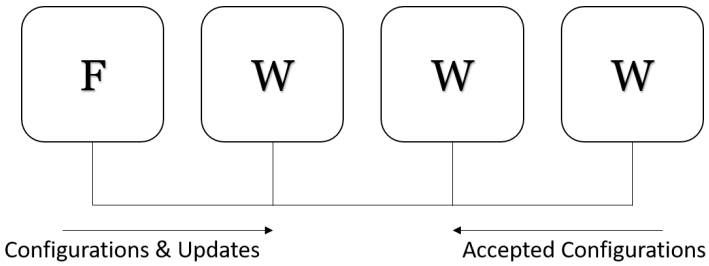

Figure 3: Diagram of farming-type parallelisation where farmer maintains the global configuration while workers check potential improvements to configuration.

attempts of parallelising simulated annealing have been successful, especially for well studied problems such as the travelling salesperson problem [4].

This parallelisation is usually done by splitting the problem into smaller sub-problems, each to be solved by sequential simulated annealing as in Algorithm 1 and then recombining these solutions. Splitting the problem in this way must be done with great care, as high interdependency of the sub-problems can be problematic. If the interdependent sub-problems do not communicate then the recombination will lead to a solution that is potentially far from globally optimal, analogous to the locally optimal structures from quenching the system as a whole. Otherwise, if the sub-problems communicate, because of their high interdependence then communication between processors is required leading to only small efficiency gains as compared to running a sequential annealing.

Diekmann et al. [4] considered a more general setting for parallelisation, though restricting their focus to combinatorial optimisation. Here we will consider two of the parallelisation methods they discuss. They considered two classes of combinatorial problems, K_1 and K_2, which cover the majority of all practical applications of combinatorial optimisation. Problems of class K_1 have the feature that the time taken to generate neighbouring configurations is approximately equal to the time taken to compute the change in cost between the current and proposed configurations. Problems of class K_2 have that the time to generate a neighbouring configuration being much less than that to compute the change in cost. An example of each type of problem would respectively be a travelling salesperson problem and a link assignment problem. A link assignment problem involves clustering a network of components so that the total distance between all clusters is minimised and is of value in computer chip design [5].

Farming parallelisation

In the Farming parallelisation of simulated annealing the computation of the change in cost and the Metropolis step to determine acceptance are parallelised. In this scheme, a 'farmer' processor, using the current configuration of the system generates neighbour configurations, which it sends to one of many 'worker' processors as in Figure 3. Each worker then calculates the change in cost and determines acceptability. If the configuration is accepted then it is sent back to the farmer which then initiates a global update of the current configuration. Otherwise, the configuration is discarded and the worker receives another configuration to assess.

Since there is one global state being maintained by the farmer processor, the convergence behaviour of this parallelisation is the same as that of the sequential annealing. This means that the metaheuristic is guaranteed to eventually converge to the optimal solution, but the time taken to reach a near optimal solution is reduced by simultaneously evaluating potential improvements to the configuration. In the case of K_1 class problems, because the proposal times and calculation times are similar the farmer acts as a bottle neck. At high temperatures or when the number of worker processors is high, accepted configurations arrive faster than the farmer can update the global solution. For K_2 problems however, the farming parallelisation works well, as the farmer is not overwhelmed. Averaged across 50 link assignment problems farming

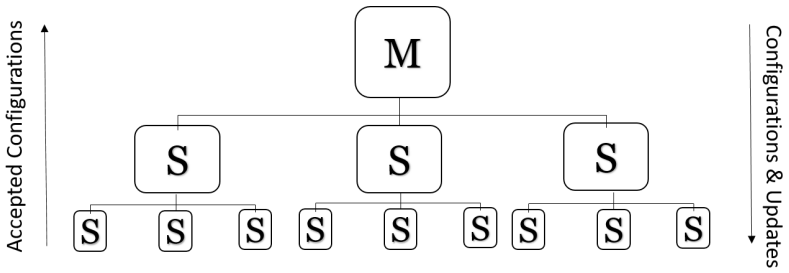

Figure 4: Diagram of One-Chain parallelisation, avoiding the bottleneck of the farming parallelisation. A master processor maintains the global configuration, upper level servant processors manage accepted configurations and propose neighbour configurations to lower level servant processors to compare to the current configuration.

parallelisation lead to a 30% reduction in computation time while maintaining 60% utilisation rate across processors [4].

One-Chain Parallelisation

In order to avoid the bottleneck forming at the farmer processor, the generation of new configurations must also be parallelised. This leads to a one-chain parallelisation, as shown in Figure 4. Here a 'master' processor maintains the global current configuration which is passed down to upper level 'servant' processors that generate neighbouring configurations which are then passed down again to lower level servant processors for cost computation and comparison to the current configuration as in Algorithm 1. If a lower level servant accepts a solution then this is passed to the upper level servant and then on to the master, which performs a global update of the configuration.

At high temperatures we still have many configurations being accepted in quick succession. By synchronising the servant processors, each batch of accepted configurations are available to the master processor at one time. This means that a servant is never interrupted mid-calculation by a global update, and that the master has potentially many acceptable configurations from which to choose. This choice of configuration can be made in several ways.

If the first accepted configuration is always chosen by the master this gives preference to solutions which are fast to calculate, influencing the convergence of the metaheuristic, and so this is not a feasible option. If the configuration with the lowest cost is always chosen then this reduces the metaheuristic to a type of local search and therefore quenching, suggesting some probabilistic rule should be used. The choice of configuration could be weighted by the cost of solution, so that each is selected according to a Boltzmann distribution. However, the gain from doing this over a random selection is not worth the computational cost of calculating these probabilities [4]. Therefore, the best policy for the master's choice between accepted configurations is to select uniformly at random between them.

By selecting only one of the accepted moves proposed by the upper level servants and discarding the others, the master processor loses many good configurations while at high temperature. Instead, moves may be retained by the upper level servants and supplied again to the master for selection until a better solution is found from the servants in the tier below.

Implementing this on 50 runs of 2 travelling salesperson problems (of 442 and 532 nodes respectively) lead to an average of a 45% reduction in computation time while maintaining an 80% utilisation rate across all processors. This resulted in solutions which were consistently within 2% of the known optimal cost [4].

3.2.1 Parallelisation Review

Sequential simulated annealing requires a large computational effort to arrive at close to optimal solutions, which in the sequential case can lead to large computation times. Through the careful

division of the problem into loosely related sub-problems, or through more general parallelisation methods such as farming and one-chain, the time required to achieve these near optimal solutions can be dramatically reduced. Indeed, for some classes of problem the time taken can be reduced by almost fifty percent, combating one of the weakest aspects of the simulated annealing metaheuristic.

4 Conclusion

Simulated annealing provides a clear and simple approach to finding near optimal solutions of difficult combinatorial optimisations where there are many local minima. Although not fast in absolute terms it is still comparatively fast in relation to other approaches to combinatorial optimisation, and this speed can be improved though careful parallelisation.

The wide applicability of simulated annealing, as well as its ease of explanation to non-specialists through the crystallisation metaphor, make it an attractive metaheuristic. This is demonstrated by its use in a range of combinatorial applications including but not limited to timetabling, circuit design and image processing [9]. Simulated annealing has also been shown to be successful when extended to continuous multivariate functions which are highly multimodal, though the practical applications of this extension are limited [3].

One key weaknesses of simulated annealing, both sequential and parallelised is that it can require much tuning of the initial temperature and annealing schedule. Although this is costly in terms of the time of the user, Vidal [9] argues that this process provides valuable insight into the traits of the near optimal configurations. For applications where this insight is not desired, more recent work has been focused on automating this procedure as in Wright [10].

References

[1] Brownlee, J. (2011). *Clever algorithms: nature-inspired programming recipes*. Jason Brownlee.

[2] Brualdi, R. A. (1977). *Introductory combinatorics*. Pearson Education India.

[3] Corana, A., Marchesi, M., Martini, C., and Ridella, S. (1987). Minimizing multimodal functions of continuous variables with the "simulated annealing" algorithm corrigenda for this article is available here. *ACM Transactions on Mathematical Software (TOMS)*, 13(3):262–280.

[4] Diekmann, R., Lüling, R., and Simon, J. (1993). Problem independent distributed simulated annealing and its applications. In *Applied Simulated Annealing*, pages 17–44. Springer.

[5] Kirkpatrick, S., Gelatt, C. D., Vecchi, M. P., et al. (1983). Optimization by simulated annealing. *science*, 220(4598):671–680.

[6] Press, W. H., Teukolsky, S. A., Vetterling, W. T., and Flannery, B. P. (1982). *Numerical recipes in C*, volume 2. Cambridge Univ Press.

[7] Safaei, N., Banjevic, D., and Jardine, A. K. (2012). Multi-threaded simulated annealing for a bi-objective maintenance scheduling problem. *International Journal of Production Research*, 50(1):63–80.

[8] Van Laarhoven, P. J. and Aarts, E. H. (1987). *Simulated annealing*. Springer.

[9] Vidal, R. V. (1993). *Applied simulated annealing*, volume 396. Springer.

[10] Wright, M. (2010). Automating parameter choice for simulated annealing.