Lancaster University Management School
Working Paper
2010/032

Automating Parameter Choice For Simulated Annealing

Mike Wright

The Department of Management Science
Lancaster University Management School
Lancaster LA1 4YX
UK

AUTOMATING PARAMETER CHOICE FOR SIMULATED ANNEALING

Mike Wright
Department of Management Science
Lancaster University
Lancaster
UK
m.wright@lancaster.ac.uk

Abstract

Most metaheuristic techniques, including Simulated Annealing, require the specification of parameters before they can be used. Setting these parameters is not straightforward, and there is no accepted way to determine good values for the parameters other than trial and error, which is uncertain and can be very time-consuming.

Experimental results are presented to help determine how parameters for Simulated Annealing should be set automatically, given only the time available for a solution. These results, for two randomly-generated Euclidean Travelling Salesman Problems, demonstrate, among other features, that the prevailing orthodoxy concerning starting temperatures is significantly suboptimal. The paper ends with a suggested set of steps to be undertaken to determine the parameters for any given instance.

Key words

Simulated Annealing; parameters; automation; Travelling Salesman Problem

Introduction

All successful metaheuristic techniques require various decisions to be made before they can be implemented. For neighbourhood search methods such as Tabu Search (TS) or Simulated Annealing (SA) this includes the creation of an initial solution and the definition of neighbourhood moves; for a Genetic Algorithm (GA) this includes recombination methods and often repair mechanisms; etc.

Over and above this, when these decisions have been taken, various parameters must be set. Thus for TS the list length, candidate list percentage and, in some cases, recency and frequency parameters must be set; for GA there is the population size and the mutation rate; for Particle Swarm Optimisation (PSO) there is the swarm size and the relative weight of local and global best solutions; etc.

Unfortunately there is often little evidence-based consensus as to how the best values for these parameters should be determined. Thus research papers often refer to fine-tuning processes, which are usually little more than trial and error, which are undertaken as a preliminary to running the methods for the problem and data in question.

This situation is unsatisfactory for a number of reasons.

- We generally do not know whether the parameters used really are the best, or even approximately the best – usually the "fine-tuning" is far from exhaustive, and in practice details of it are often omitted from papers. This may make a significant difference when different techniques are compared with one another.

- This fine-tuning may take a lot of time – time which is usually not reported or used when comparing run times of various methods.
- For practical OR implementations it is not normally appropriate to expect the user of a system to carry out this fine-tuning, or even to understand what the parameters mean.

However, there is one factor which is not only reasonable but also almost essential to expect a system user to specify: this is the time elapsed before a solution is reached. In some applications this may be a matter of seconds, while in others it may literally be several days. The timescale is very likely to affect the best parameters to use.

Thus it would appear to be very worthwhile to try to find ways in which these methods can specify their own parameters automatically, given the time available.

Ideally one could take this reasoning two steps back; initial analysis of a problem could determine which approach to use and also make decisions about the best neighbourhood definitions etc. This paper has a less ambitious aim than this: it aims to shed some light as to how parameters should be automatically specified when using SA in a fairly simple and standard way. This can be seen as a necessary first step towards the overall goal.

Simulated Annealing

SA is a technique which has been in widespread use since the 1980s, though it was first proposed by Metropolis et al. (1953) some thirty years earlier, and its use shows no sign of diminishing. Google Scholar™ (accessed on 29th June 2010) gives 453 articles published during 2009 with the phrase “Simulated Annealing” in the title. This makes SA equal third in the popularity league table for metaheuristics, well behind GA (4003) and PSO (1574), about the same as Ant Colony Optimization (429), but ahead of other well-known methods such as TS (204), Iterated Local Search (21) or Hyperheuristics (4).

SA has been the subject of many variations and hybridisations over the years, and thus many of these papers will be concerned with non-standard versions of SA. Dowsland (1995) mentions a variety of strategies including dynamic temperature control, periodic reheating and parallel implementation.

However, this paper will only concern itself with the simplest form of SA. The reasoning is that, until it is understood how best to run simple SA, there is little hope of understanding how best to run the multitude of variations.

Simple SA works as follows.

1. Create an initial solution – denote it the “current solution”
2. Set an initial value (T_0) of T, the “temperature”
3. Randomly perturb the current solution to create a neighbouring solution (this needs a definition of perturbation and thus of neighbourhood)
4. Calculate ΔC, the increase in cost (assuming this is a cost minimisation problem)
5. Accept the new solution (i.e. make it the new current solution):
    a. definitely if $\Delta C \leq 0$
    b. otherwise with probability $e^{-(\Delta C/T)}$
6. Possibly adjust T (see below)

7. Return to 3 above and continue, until some stopping criterion is met, in which case stop.

There are a number of ways of adjusting the temperature (as in Step 6 above), but one of the most common ways is to multiply T by a predetermined factor α after every X iterations, where an iteration is a loop through Steps 3 to 7 above. α is usually just below 1, and X may take a variety of values, but needs to be of a lower order of magnitude compared with the total number of iterations.

The simplest form of stopping criterion is to stop either after a fixed number N of iterations or when the temperature reaches a value T_N. This of course needs to be determined in line with the overall timescale required for presenting a solution.

Thus we have five potential parameters, T_0, T_N, N, α and X. However, these can easily be reduced from five to three by the following reasoning.

First, it is easy to see that $T_N = T_0 * \alpha^{(N/X)}$. Therefore, as soon as any four parameters have been specified, the fifth may be calculated directly.

Moreover, multiplying the temperature by $\alpha^{(1/X)}$ every iteration has almost exactly the same effect on T as multiplying it by α every X iterations, unless X is significantly large compared with N (which it usually is not). Thus we can simplify further by letting X = 1.

So we need only specify three parameters. The simplest course is to specify the starting and ending temperatures (T_0, T_N) and the number of iterations N. Unlike α, which is clearly closely related to the other parameters, these three parameters have intuitively clear meanings and may perhaps work fairly independently of one another (such that, for example, the best value of T_0 may not depend on the value chosen for N, though this will be tested later).

For a given implementation, it can be easily seen that the solution quality may depend on getting sensible values for all three of these parameters. For example:

- If N is too high, the answer is not produced soon enough, which may be no better in practice than not producing any answer at all;
- if N is too low, then we are not making as much use as we could of the time available – and, like all metaheuristics, SA has no natural stopping point, and thus using more time will on average produce better solutions;
- if T_0 is too high, then a significant proportion of the time is being wasted at the start making little if any progress, perhaps being little better than a random walk;
- if T_0 is too low, then there is not enough diversification at the start, and the search will not have the opportunity to deviate significantly away from the area of search space containing the initial solution;
- if T_N is too high, then there is not enough intensification at the end, and the search may not even reach a single local optimum, let alone make a thorough examination of a good area of search space;
- and if T_N is too low, then large amounts of time may be wasted towards the end of the search doing more or less nothing, with the solution changing hardly if at all.

Preliminary analysis

It is not possible to set these parameters without knowing something about the problem being addressed. Firstly, in order to set N, we need to know how long each iteration takes. This will depend upon the type and size of the problem, the efficiency of programming, the operating system, the computer used etc. Preliminary analysis of a few iterations will give us a good approximation for this, since for most applications the time taken for each iteration is usually approximately constant, or even if it is not then an approximate distribution can be achieved with a reasonably small number of preliminary iterations.

However, some prior analysis is also necessary in order to set the starting and ending temperature. This is because the temperature T is used in the formula $e^{-(\Delta C/T)}$, so the best values for T_0 and T_N will depend upon the distribution of ΔC in some way. Until we have some information about this distribution, we cannot even speculate sensibly about good values. Note that this means that we need to undertake this preliminary analysis not just when addressing a new type of problem but for every single instance.

So, for both the above reasons, we need to start by creating an initial solution, undertaking some iterations and analysing the results.

How to set the initial temperature

A number of authors have suggested principles to be observed when setting T_0. Kirkpatrick et al. (1983) suggest that it should be high enough so that almost all perturbations are accepted, and other authors have followed their lead. Aarts et al. (1997, page 114) suggest that it should be set equal to an estimated value for the maximal cost difference between any two neighbouring solutions. Dowsland (1995, page 29) agrees, saying that the initial temperature must be hot enough to allow an almost free exchange of neighbouring solutions, in order that the final solution should be independent of the starting solution. Hoos and Stützle (2005, page 77) have used an initial temperature such that only 3% of perturbations are not accepted. Ben-Ameur (2004) has suggested that this percentage should be 20%, and has devised a technique for quickly determining a temperature that will produce the desired percentage, to which we will return later.

Among more recent papers, Zhang et al. (2010) suggest a rejection rate of around 5% or 10%. Kuo (2010) suggests that the initial temperature should be such that even the worst perturbation should have at least a 50% probability of acceptance.

Other authors are less helpful. For example, many such as Seyed-Alagheband et al. (2010), Sahin et al. (2010) and Mesgarpour et al. (2010) simply state that some experimentation was done in order to obtain a good value, without giving any further details. Some, e.g. Liu and Liu (2010), vaguely describe the initial temperature value as "high". Worse still, very many authors simply do not say how the initial temperature was obtained or why.

However, some experimental work has been reported which shows that the orthodox view may be highly suboptimal. Varanelli and Cohoon (1999) have shown that reducing the initial temperature significantly can result in final solutions of equal quality in a much shorter space of time. Moreover, even in the very early days of SA implementations, Johnson et al. (1989) reported that an initial rejection rate of around 60% worked well for their graph partitioning problems. Given that they were starting from random initial solutions, for which a high proportion of perturbations would improve the cost, this is a very long way from the prevailing orthodoxy.

Indeed, on close examination it appears that the orthodox view as propounded by Aarts et al., Dowsland, etc. is based not on actual evidence but more on mathematical theorems and appeals to what might make sense or seem reasonable. But there is a lot about their arguments that does not appear reasonable at all.

In particular, why should it be important that the final solution is independent of the initial solution? This might be of importance if there were virtually infinite time available and if it were of supreme importance to find the global optimum, but this is rarely (if ever) the case. Generally the objective of the exercise is to find a good solution within the available time. A good solution produced quickly which is not independent of the initial solution will be far preferable to a less good solution produced slowly which is independent, or to no solution at all because time has run out.

Perhaps it is precisely this insistence upon independence which has led to the frequent adoption of very high initial temperatures which, as noted earlier, mean that the search spends a very long time making virtually no gains. It is thus not surprising that there is a widespread view that SA is a very slow and/or ineffective technique.

This is not to say that high initial temperatures are necessarily a bad thing, but merely that there is no experimental evidence that they are a good thing. What is missing, and what this paper aims to provide in a relatively small way, is hard factual evidence about what works best; either a direct way of relating T_0 to the distribution of ΔC or an indirect way, first specifying a suggested value for the Percentage Acceptance of Worsening Solutions (henceforth PAWS), followed by a method such as that suggested by Ben-Ameur (2004) to find a temperature that will produce this PAWS value. Note that only worsening solutions have been included in this measure; non-worsening solutions are always accepted whatever the temperature and thus to include them is inappropriate.

How to set the final temperature

It is even harder to decide how to set a value for T_N, since the distribution of ΔC near the end may be very different from its distribution near the start, and thus cannot be gauged easily at the start of the process. Likewise, the relationship between any given value of PAWS and the ending temperature may be very different from the relationship between PAWS and the starting temperature.

However, as will be seen later, the specification of the ending temperature is even more important than the specification of the starting temperature. There will be further discussion of this later, after the presentation of experimental results, and a way forward is proposed.

Experiments

It is therefore necessary to undertake extensive experimentation to investigate this issue, in order to gain some evidence as to what works best in practice. If we can discover what parameters seem to work best for a specific example, we can then to relate this to PAWS and/or the distribution of ΔC for that specific example.

Two simple Euclidean Travelling Salesman Problems were chosen for the experiments, one with 100 cities and one with 200 cities. For each data set, the position of each city was determined by choosing x and y co-ordinates at random between 1 and 1000. The neighbourhood scheme used for each was the common 2-opt scheme.

Now of course such problems are routinely solved to optimality nowadays – see, for example, Applegate et al. (2006). However, the purpose of the experiments is not to put forward a method for solving simple TSPs, but rather to obtain comparative evidence for various temperature parameters when using the basic version of SA. Simple TSPs were chosen because iterations can be made extremely quickly – the calculation of change in cost takes a small fraction of a millisecond even when run on an ordinary laptop computer – and thus extensive experimentation is feasible.

The problems are not only quick to solve but they are also very simple in concept and may have rather more tractable solution landscapes than many real-life implementations. Thus further experimentation may be required later to see to what extent the results for simple Euclidean TSPs can be generalised. However, such experiments are outside the scope of this paper.

Originally a TSP with only 50 cities was chosen, but it proved too easy to reach the optimal solution for a wide range of parameter values. Thus the results did not discriminate sufficiently between the different combinations of parameters. However, 100 cities proved to be large enough to pose a serious test so that the importance of setting good values for the parameters would be shown up by the experiments. The 200-city example was later run as a check to see if consistent policies could be found which would apply to both data sets.

For each data set, over 100,000 computer runs were made, by making 100 runs for all combinations of the following:

- two strategies for creating the initial solution – random or non-random
- eight values for T_0: 1000, 500, 200, 100, 75, 50, 25 and 10
- thirteen values for T_N: 50, 25, 20, 15, 10, 7.5, 5, 3.5, 2, 1, 0.5, 0.2 and 0.1
- 5 values of N: 25000, 100000, 500000, 2000000 and 5000000

The non-random initial solution procedure used was a technique sometimes known as Successive Inclusion or Insertion. This is a constructive heuristic which considers cities in a fixed order and builds up a tour one city at a time. At each step it inserts the new city into the section of the partial tour which minimises the increase in total cost. This method very quickly produces a reasonably good solution.

To put the values of N in context, the neighbourhood size for a TSP under 2-opt is roughly equal to the square of the number of cities. Thus, for the shortest runs, the number of iterations was only around 2.5 times the size of the neighbourhood even for the 100-city example, and significantly smaller than the neighbourhood size for the 200-city example. This is a very small ratio indeed compared with most implementations of SA, but this small value of N was included in order to detect whether such a small number of iterations would have a noticeable effect on the best values for T_0 and T_N.

The aim, for each initial strategy and each value of N, was to discover which set of temperature parameters worked best. Initially eight values were included for each of T_0 and T_N, but after examining the results a further five values of T_N were added (50, 20, 15, 7.5 and 3.5) so as to give a more complete picture.

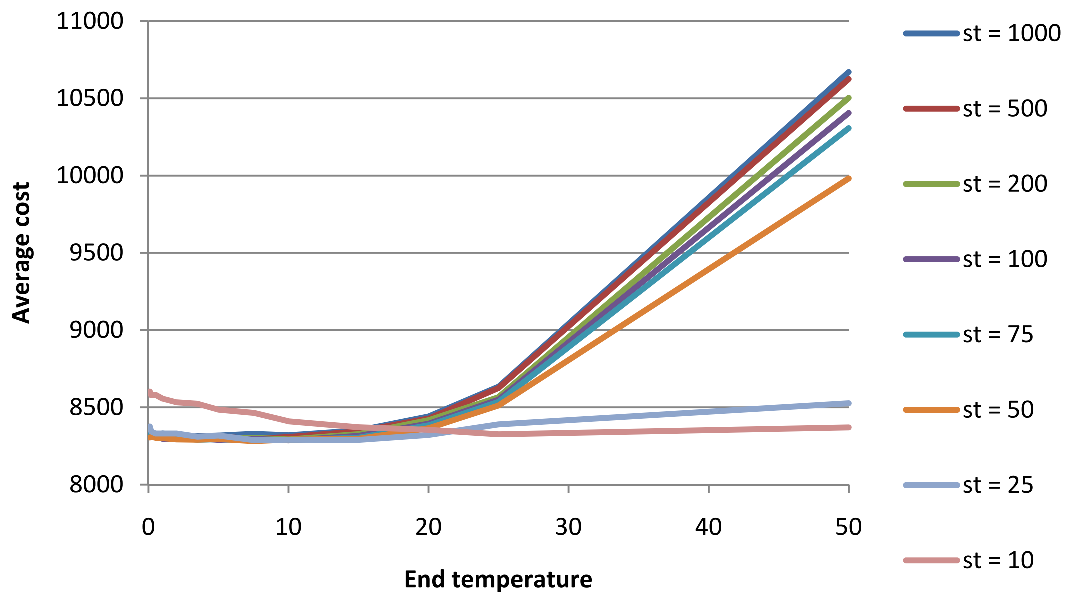
The runs include some where the final temperature exceeded the initial temperature, which is against the spirit of SA as it is usually applied! However, these were included for completeness and do show interesting characteristics.

Results

An example of a summary of results is given in Figure 1, which is for the 100-city example with random initial solutions and N = 5000000. This shows the average of the best cost found over the 100 runs on the y-axis, with the value of T_N on the x-axis. The different lines relate to different values of T_0 ("st" in the figures).

For each run, the solution value recorded was always the best found during the run, which is not necessarily the same as the final solution.

Figure 1: Average solution costs for 100 cities for sets of 100 runs of five million iterations each, with random initial solution

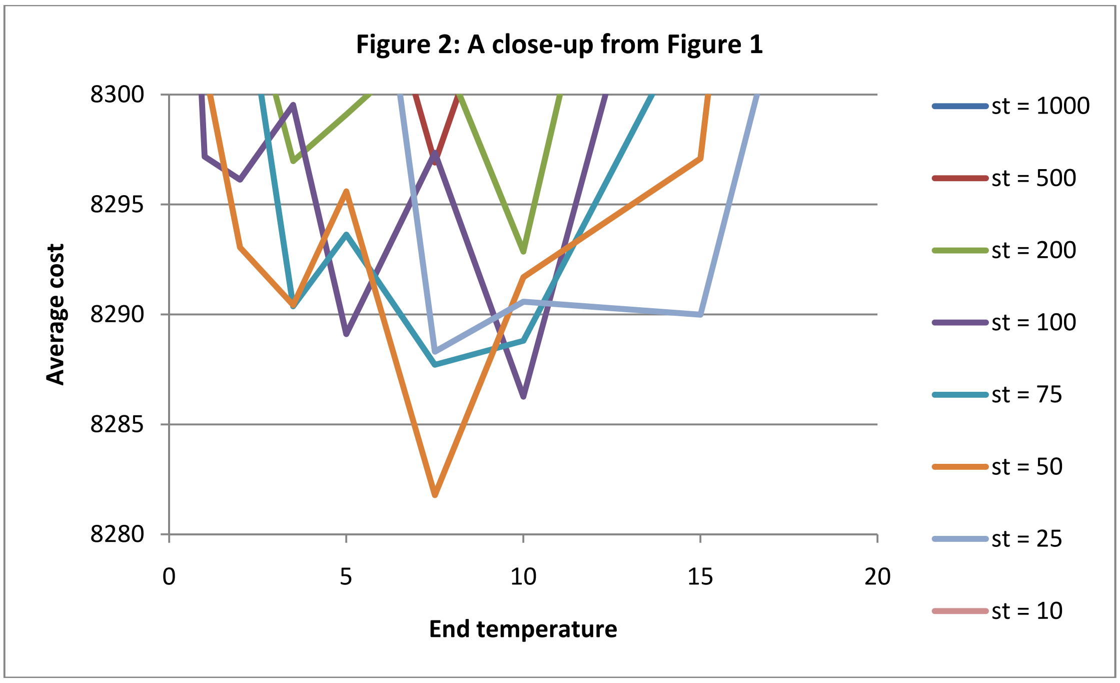


On the right-hand side of this figure, e.g. for when the end temperature is 50, the lines are in logical order, with the lowest representing st = 10, the next lowest st = 25, etc.

Some interesting points emerge from Figure 1:

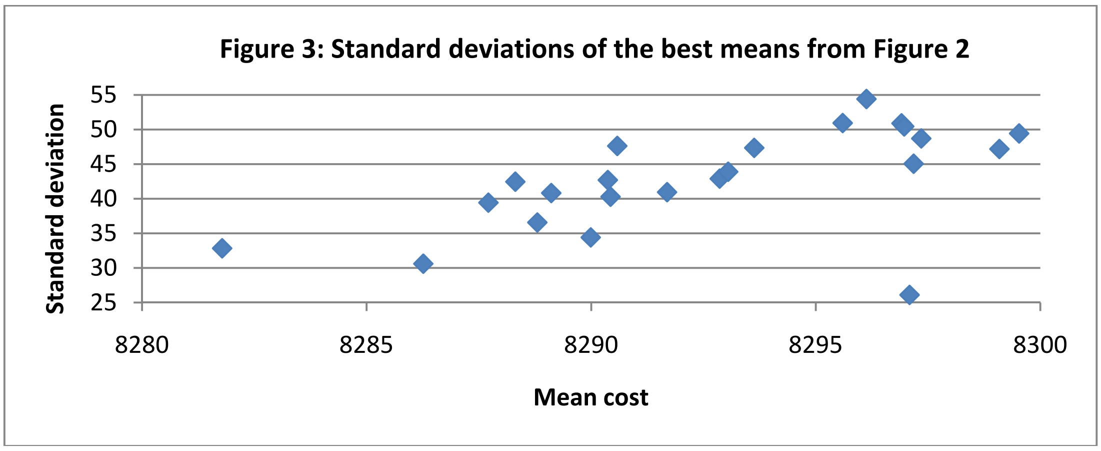
- If the starting temperature is very low (e.g. 10), then the temperature needs to increase during the search ($\alpha>1$). This is because there is enough intensification, but not enough diversification, near the start of the search.
- In other cases, the most important thing is to ensure that the ending temperature is not too high; otherwise the solution quality will be very poor indeed.
- If the ending temperature is too high then the picture looks entirely logical as regards the effect of the starting temperature: for the whole of the right-hand half of the picture a line with a higher starting temperature is consistently higher than a line for a lower starting temperature.

However, this logical ordering does not apply at the most interesting points, i.e. where average costs are at their lowest. Figure 2 therefore gives a close-up of that part of Figure 1 where the solutions are best.



In this picture, the most extreme values of T_0 (1000 and 10) do not feature at all – they never succeeded in producing an average solution of below 8300. The line that comes down the lowest is for st = 50, followed closely by st = 100, st = 75 and st = 25 in that order, with st = 200 and st = 500 only just making it into the top of the diagram.

To distinguish between these results we need to know the standard deviations involved. Figure 3 shows the standard deviation of the best costs graphed against the average best costs.



We can see from Figure 3 that, among the best solutions, there is a general tendency for the runs with the lowest means to have the lowest standard deviations, showing that the best sets of solutions are also the most consistently good. We can also see that the sets of runs with good means have standard deviations below 50, and the best set of all has a standard deviation of about 33. Given that the means are of 100 values, and making reasonable assumptions about approximate normality, we can therefore say that the standard deviations of the best means are between 3 and 5; thus any mean cost more than about 8 above the best is significantly worse in a statistical sense using a 95% criterion.

The sets of runs with a mean cost within 8 of the best are shown in Table 1.

T_0	50	100	75	25	75	100
T_N	7.5	10	7.5	7.5	10	5
Mean cost	8281.8	8286.3	8287.7	8288.3	8288.8	8289.1

Table 1: summary of runs with means within 8 of the best, for 100 cities and 5 million iterations with random initial solutions

Thus we can be fairly confident that, for this data set, using simple SA with 2-opt neighbourhoods, with a random starting solution and 5000000 iterations, the best policy is to set the initial temperature between 25 and 100 and the final temperature between 5 and 10.

Now going through the same procedure for other sets of runs for the 100-city data set, the best ranges for T_0 and T_N are shown in Tables 2 (for random initial solutions) and 3 (for non-random initial solutions).

N	25000	100000	500000	2000000	5000000
T_0 range	50	25 - 75	50 - 75	25 - 200	25 - 100
T_N range	2 - 5	3.5 - 10	5 - 10	3.5 - 15	5 - 10

Table 2: best ranges for T_0 and T_N, with random initial solutions, for all values of N, for the data set with 100 cities

N	25000	100000	500000	2000000	5000000
T_0 range	10 - 50	25 - 50	25 - 50	25 - 100	25 - 100
T_N range	2	5 - 15	10	3.5 - 15	5 - 15

Table 3: best ranges for T_0 and T_N with non-random initial solutions, for all values of N, for the data set with 100 cities

In general, there was a slight tendency for values of T_N at the upper end of the range to work best in combination with values of T_0 at the lower end of its range, and vice versa. In particular, a value of $T_N = 15$ produced good results only when in combination with $T_0 = 25$.

The differences between the best values can be seen not to depend very much on the value of N, except perhaps for the shortest runs, with N = 25000. With such a short run it is imperative that intensification is given greater prominence than diversification, otherwise there is a danger that no solution will be produced which is even reasonably good. However, the effect is not as marked as had been expected.

Not surprisingly, for the two smallest values of N, the best average solution was significantly lower with the Non-Random Initial Solution (NRIS) than for the Random Initial Solution (RIS). This is not surprising, since Successive Inclusion does give quite good solutions. However, for large values of N there was no significant difference between the effects of the initial solution procedures. Table 4 gives the details.

N	25000	100000	500000	2000000	5000000
NRIS	8724.4	8543.0	8366.5	8304.8	8279.1
RIS	9005.6	8571.5	8366.1	8306.6	8281.8

Table 4: Best average final solutions found for the 100-city data set

The best single solution found had a cost of 8254.4. This cost was reached on several occasions. It is thus reasonable to speculate that this may be the cost of the optimal solution or solutions.

Figures 4, 5 and 6 are the same as Figures 1, 2 and 3 except that they apply to the 200-city example.

Figure 4: Average solution costs for 200 cities for sets of 100 runs of five million iterations each, with random initial solutions

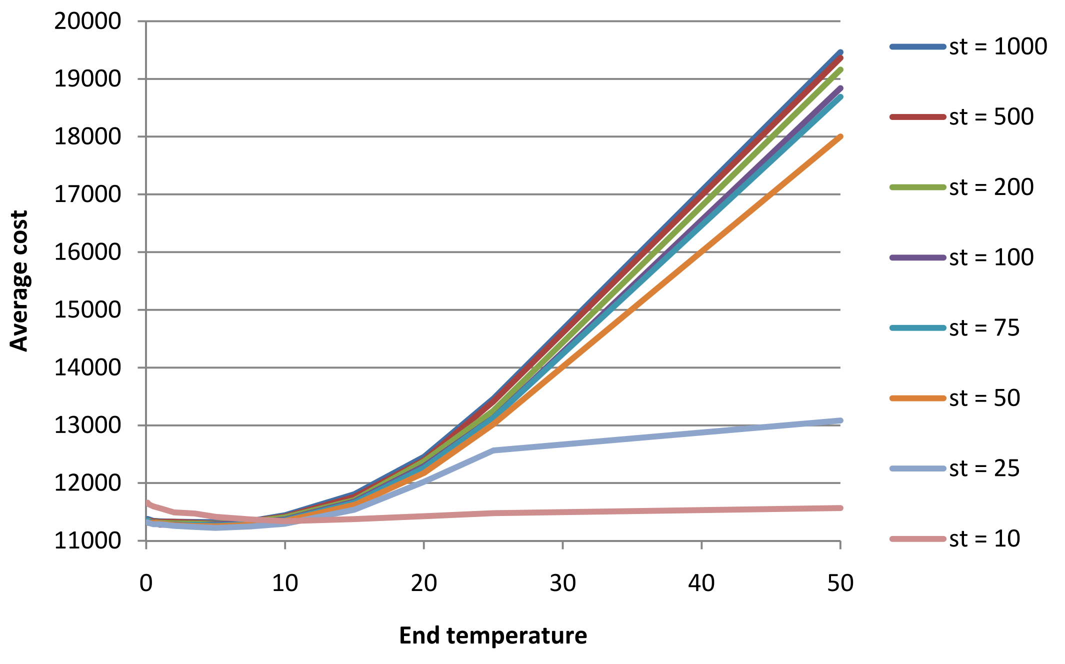
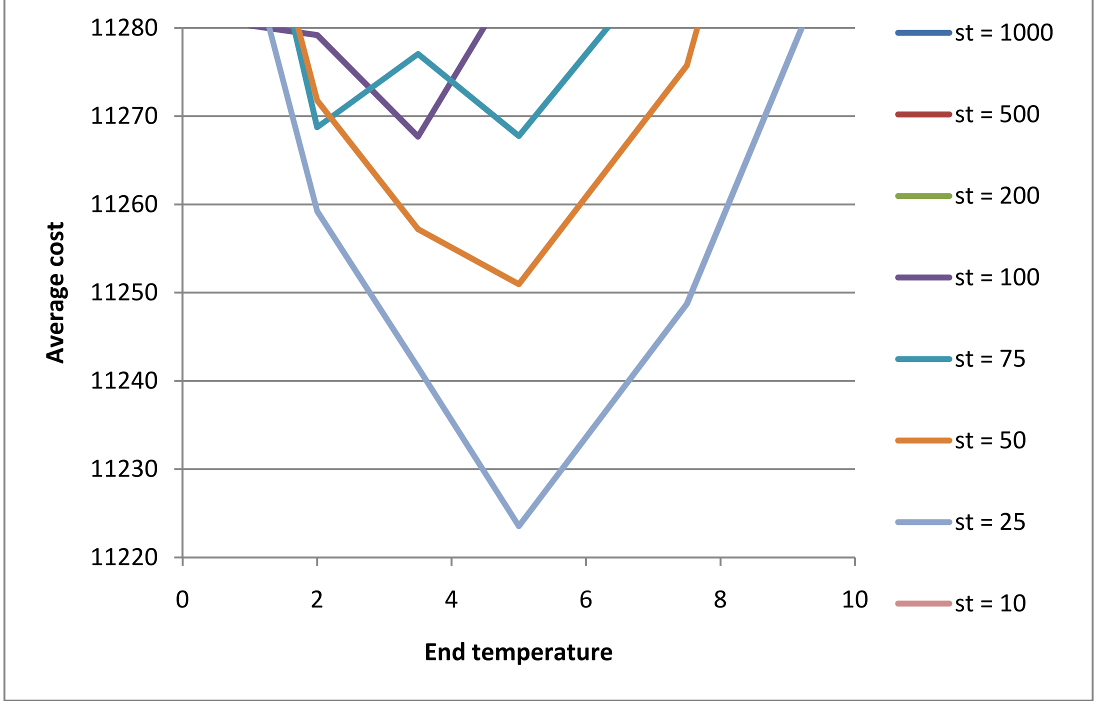
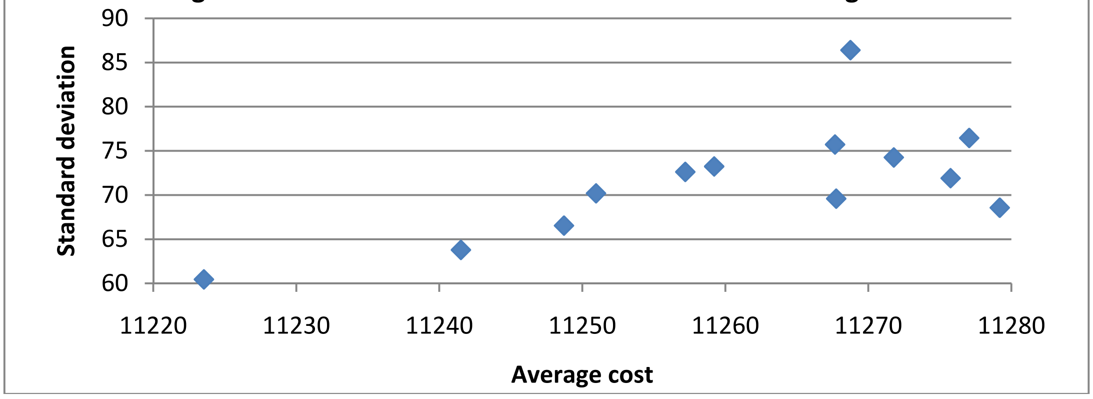


Figure 5: A close-up from Figure 4



Here the line that comes down the lowest is for st = 25, followed by st = 50, then st = 75 and st = 100. Other values do not feature in this Figure.

Figure 6: Standard deviations of the best means from Figure 5



Figures 4 to 6 display the same general features as Tables 1 to 3. The standard deviations of the best means are between 6 and 7.5. However, here the best mean is 25.2 lower than the next best, a difference considerably more than twice these standard deviations. Thus the values T_0 = 25 and T_N = 5 can be confidently declared better than all the other combinations tried for this example.

Tables 5 to 7 are the same as Tables 2 to 4 except that they apply to the 200-city example.

N	25000	100000	500000	2000000	5000000
T_0 range	10 - 50	25 - 50	25 - 50	25	25
T_N range	0.1 - 5	1 - 5	5 - 7.5	5	5

Table 5: best ranges for T_0 and T_N, with random initial solutions, for all values of N, for the data set with 200 cities

N	25000	100000	500000	2000000	5000000
T_0 range	10	10 - 25	25	25 - 75	25 - 100
T_N range	0.5	0.1 - 5	5	3.5 - 7.5	3.5 - 7.5

Table 6: best ranges for T_0 and T_N with non-random initial solutions, for all values of N, for the data set with 200 cities

N	25000	100000	500000	2000000	5000000
NRIS	12014.3	11805.7	11543.5	11335.7	11252.7
RIS	15289.6	12292.0	11558.8	11292.3	11223.5

Table 7: Best average final solutions found for the 200-city data set

Again the differences between the best values do not appear to depend very much on the value of N, except when N is small. This effect is stronger here than for 100 cities, which is not surprising: 25000 iterations is relatively a much smaller number for the 200-city example, with neighbourhood size around 40000, than for the 100-city example, with neighbourhood size about 10000.

For the two smallest values of N, the best average solution was significantly lower with the non-random starting solution than for the random starting solution, which again was not surprising. Perhaps more surprising was the fact that, for larger values of N, the random initial solutions gave significantly better final solutions than initial solutions produced by Successive Inclusion, as shown in the final two columns of Table 7. However, we should be cautious here: the best possible values of T_0 and T_N are probably not represented in these results, since we only tried a small number of discrete options. So it may be, for example, that values of $T_0 = 35$ and $T_N = 4$ could have changed the figure of 11252.7 in Table 7 (obtained with $T_0 = 25$ and $T_N = 5$) to something closer to 11223.5.

Percentage of Acceptance of Worse solutions (PAWS)

During the computer runs with 5 million iterations, a record was kept of the percentage of worsening perturbations accepted near the start (during the first 1000 worsening iterations) and near the end (during the last 5000 worsening iterations). During these periods the temperature was thus very close indeed respectively to T_0 and T_N.

The results are shown in Figures 7 to 10. Figures 7 and 8 relate to the 100-city TSP, and Figures 9 and 10 to the 200-city TSP. As before, "PAWS" is the Percentage Acceptance of Worse Solutions (solutions which are not worse than the current solution are always accepted, whatever the temperature); "RIS" means Random Initial Solution and "NRIS" means Non-Random Initial Solution.

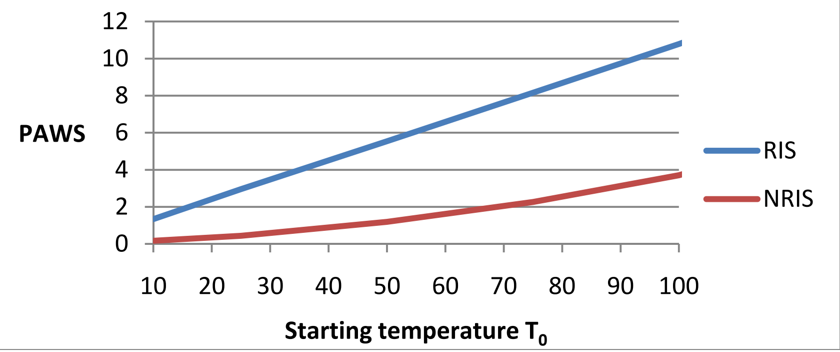


Figure 7: 100-city TSP: PAWS against T_0

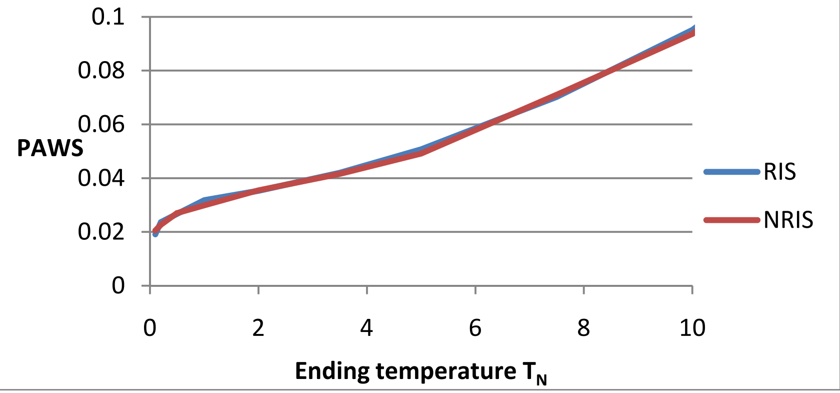


Figure 8: 100-city TSP: PAWS against T_N

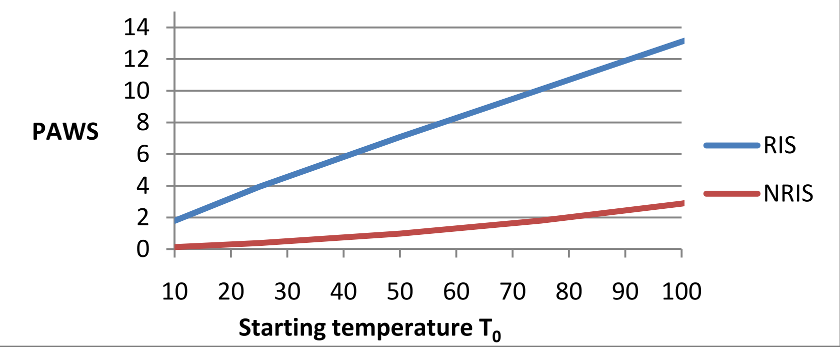
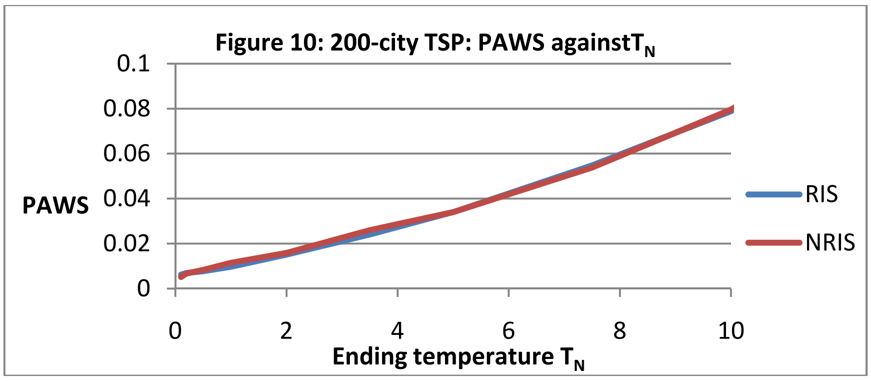


Figure 9: 200-city TSP: PAWS against T_0



In Figures 7 and 9 the RIS line is higher than the NRIS line; for Figures 8 and 10 the lines are almost identical, showing unsurprisingly that the best PAWS value at the end of the search does not depend at all on the nature of the initial solution procedure.

Recall that, for the 100-city example, the best values of T_0 are in the range 50 to 75 approximately. From Figure 7 we can see that this corresponds to a PAWS value between 5 and 8 with a random initial solution, and between 1 and 2 with a non-random initial solution.

Likewise, for the 200-city example, the best values of T_0 are in the range 25 to 50 approximately. From Figure 9 we can see that this corresponds to a PAWS value between 4 and 7 with a random initial solution, and between 0.3 and 1 with a non-random initial solution.

Contrast this with the advice from other authors quoted earlier, who suggest in effect a PAWS value of close to 100 for T_0, assuming a random initial solution. Even a value of T_0 = 1000 here gives a PAWS value of only about 71 for each of the two data sets, and as we have seen from Figures 2 and 5 this value is far too high to give solutions which even come anywhere near the best. Thus these results can be seen to challenge the prevailing orthodoxy to a significant degree.

Recall that, for the 100-city example, the best values of T_N are in the range 5 to 10 approximately. From Figure 8 we can see that this corresponds to a PAWS value between 0.05 and 0.1. Likewise, for the 200-city example, the best values of T_N are in the range 3.5 to 7.5 approximately. From Figure 10 we can see that this corresponds to a PAWS value between 0.02 and 0.05. There is thus a reasonable level of consistency between the best PAWS values for the two data sets, which suggests that it may be a good measure to use.

Ben-Ameur's (2004) method for finding a temperature to correspond with any particular value of PAWS can be used here. It starts by generating worsening perturbations at random from the initial solution and recording, for each, the resultant increase in cost (perturbations for which the cost does not increase are discarded). This is followed by a very quick iterative procedure for which the temperature gets ever closer to the required value.

The accuracy of this technique of course depends upon the number of perturbations generated, the required value of PAWS and the size of the problem (more strictly, the size of the neighbourhoods). If time is short, then it is a moot question as to how many perturbations should be chosen, since a large number carried out now will mean there is less time remaining for a correspondingly smaller number during the search process.

However, this procedure may need to be modified when determining how to relate the final PAWS value to the ending temperature. This can be seen by looking at those temperatures which we have used both as a starting temperature and as an ending temperature, as shown in Tables 8 and 9.

Temperature	10	25	50
Around an RIS	1.33	2.95	5.54
Around an NRIS	0.17	0.44	1.19
Around a final solution	0.09	0.41	1.74

Table 8: Average PAWS values for the 100-city TSP

Temperature	10	25	50
Around an RIS	1.79	3.94	7.08
Around an NRIS	0.13	0.38	0.98
Around a final solution	0.08	0.42	1.74

Table 9: Average PAWS values for the 200-city TSP

Certainly if we start from a random initial solution, we cannot assume that any given PAWS value will lead to the same temperature at the start as it would at the end of the search. The solution landscapes must be very different.

However, it is reasonable to expect that the solution landscape near a good final solution may be similar in nature to that near a good initial solution such as that produced by the successive inclusion procedure, and these figures appear to bear this out to some extent at least. Further runs were therefore made to obtain values of PAWS for other starting temperatures from non-random starting solutions and then to compare the PAWS figures with those obtained when using the same temperature as the ending temperature, with results as shown in Tables 10 and 11.

Temperature	0.1	0.2	0.5	1	2	3.5	5	7.5	10	15	20	25	50
Around an NRIS	0.026	0.025	0.033	0.042	0.059	0.085	0.105	0.135	0.166	0.241	0.334	0.438	1.185
Around a final solution	0.021	0.023	0.027	0.030	0.036	0.042	0.049	0.071	0.094	0.171	0.273	0.412	1.738

Table 10: Average PAWS values for the 100-city TSP

Temperature	0.1	0.2	0.5	1	2	3.5	5	7.5	10	15	20	25	50
Around an NRIS	0.008	0.011	0.018	0.022	0.031	0.054	0.069	0.105	0.128	0.216	0.283	0.380	0.984
Around a final solution	0.005	0.007	0.008	0.012	0.016	0.026	0.034	0.054	0.080	0.152	0.262	0.422	1.740

Table 11: Average PAWS values for the 200-city TSP

The figures look surprising for the higher temperature values, but this can be simply explained, because for high ending temperatures the final solutions are generally worse (far worse in the case of $T_N = 50$) than the initial solutions produced by Successive Inclusion. As can be seen from Figures 1 and 4 above, these values of T_N are highly suboptimal so need not concern us further.

The more interesting and relevant range of ending temperatures is between 2 and 10, as shown in Tables 2, 3, 5 and 6. In this range we see from Tables 10 and 11 that typically the PAWS value around a non-random initial solution is about twice that around a final solution.

Thus an initial analysis that starts from such an initial solution may be helpful towards setting a good final temperature. For example, if we are looking for an ending temperature that gives a 0.05 value for PAWS around the final solution, then we can construct a non-random initial solution and look for a temperature which gives a PAWS value around that solution of about 0.1.

Translating these results into action

Based on the above results, we can put forward a suggested plan of action to use when implementing a standard form of SA for a simple TSP or similar problem, given a fixed limit to the amount of solution time available.

We should aim to find a starting temperature with a PAWS value (around a random initial starting solution) of between 5 and 8 (see Figures 7 and 9 in conjunction with Tables 2 and 5) for medium to long runs, decreasing down to around 1 or 2 for short or very short runs (as an approximate heuristic we can perhaps regard a run as short if the number of iterations is less than about ten times the neighbourhood size and very short if it less than three times the neighbourhood size). Alternatively we could start from a non-random initial solution and use a PAWS value of between 1 and 2, decreasing down to about 0.1 to 0.2 for very short runs.

For our ending temperature, for medium to long runs we require a PAWS value around a final solution of around 0.05, which appears to equate to a PAWS value around a non-random starting solution of around 0.1; and for short or very short runs these figures should perhaps decrease down to around 0.01 and 0.02. However, as can be seen from Figures 1 and 4, it is much better to err on the low side than on the high side when choosing an ending temperature, and so maybe these values should be reduced a little in practice.

Ben-Ameur's (2004) method can then be used. Ben-Ameur, who used a 100-city TSP as one of his test data sets, reported that 2500 perturbations was sometimes but not always sufficient to obtain "a good approximation", without explaining precisely what this means. Given that we may be looking for PAWS values around 0.1 (or 1 in a 1000), a single acceptance out of 2500 would make a significant difference to the estimated PAWS value, and so maybe a larger number (perhaps 10000 or more?) would be required for a reasonable estimate to be reached, as long as the time available is not so short that this takes up a significant proportion.

Summing everything up, the following steps are proposed:

1. Let T be the computer time available – this is specified in advance;
2. Construct a good initial solution by means of SI (or similar fast method);
3. Generate 2500 random perturbations around this solution and calculate the average computer time taken (t) for each;
4. Set $N = (T / t) - 2500$ (or perhaps a little less so as to err on the side of caution) ;
5. If N is very large compared with 2500 (e.g. at least 100000), generate a further 7500 perturbations and reset $N = (T / t) - 10000$;
6. Calculate a factor Z equal to N divided by the neighbourhood size (for a TSP using 2-opt the neighbourhood size is approximately the square of the number of cities – for other implementations it is usually easy to estimate the neighbourhood size very quickly);
7. Use Ben-Ameur's method on these 2500 or 10000 perturbations to calculate a value for T_N corresponding to a PAWS value equal to about 0.1 if $Z > 10$, 0.01 if $Z < 1$, or $0.01 * Z$ otherwise;

8. If Z > 10, go to Step 9, otherwise go to Step 12;
9. Discard the SI solution and create a new random initial solution;
10. Carry out 10000 random perturbations around this and use Ben-Ameur's method to calculate a value for T_0 corresponding to a PAWS value between 5 and 8;
11. Go to Step 12;
12. Use Ben-Ameur's method on the 2500 or 5000 perturbations already created around the SI initial solution to calculate a value for T_0 corresponding to a PAWS value between 1 and 2;
13. Start SA from the initial solution created either at random (in Step 9) or non-randomly (in Step 2, if Step 8 not undertaken), with the values calculated for T_0, T_N and N.

Summary

Experiments have been carried out on two TSPs to produce a suggested plan for automating parameter setting within simple SA so as to produce solutions of high quality, given in advance only the time available.

The results show a number of interesting features. One result of particular interest is that the initial temperature suggested by these results is far lower than the prevailing orthodoxy, corresponding to a Percentage Acceptance rate for Worsening Solutions around a random initial solution of only around 5% to 8%, compared to figures close to 100% suggested by many other authors. If the initial solution is already of reasonable quality then this value should be between 1% and 2%.

Another interesting result is that, unless the runs are very short indeed, there is no sign of the best initial or final temperature values increasing as the number of iterations available increases. While even the longest runs carried out here may be short compared with many real-life implementations, this suggests that the main features of these results may be valid whatever the run length.

Of course, these results have only been demonstrated for randomly-generated Euclidean TSPs, and the conclusions may turn out to be significantly different for other problems. However, this work represents an important first step which needs to be built upon by carrying out similar research for other implementations.

References

Aarts E.H.L., Korst J.H.M and van Laarhoven, P.J.M. (2003) "Simulated Annealing" in *"Local Search in Combinatorial Optimization"*, eds. E.H.L. Aarts and J.K. Lenstra, Princeton University Press, Princeton, USA.

Applegate D. L., Bixby R. E., Chvátal V. and Cook W. J. (2006). *"The Traveling Salesman Problem: a Computational Study"*, Princeton University Press, Princeton, USA.

Ben-Ameur W. (2004) "Computing the Initial Temperature of Simulated Annealing", *Computational Optimization and Applications*, **29** (3), 369-385.

Dowsland K.A. (1995) "Simulated Annealing" in "*Modern Heuristic Techniques for Combinatorial Problems*", ed. C.R. Reeves, McGraw-Hill, London, UK.

Hoos H.H. and Stützle T. (2005) "*Stochastic Local Search*". Morgan Kaufmann, San Francisco, USA.

Johnson D.S., Aragon C.R., McGeoch L.A. and Schevon C. (1989) "Optimization by Simulated Annealing: an Experimental Evaluation; Part I, Graph Partitioning", *Operations Research* 37 (6), 865-892.

Kirkpatrick S., Gellat C.D. and Vecchi M.P. (1983) "Optimization by Simulated Annealing", *Science* **220**, 671–680.

Mesgarpour M., Kirkavak N. and Ozaktas H. (2010) " Bicriteria Scheduling Problem on the Two-Machine Flowshop using Simulated Annealing", *Lecture Notes in Computer Science* **6022**, 166-177.

Metropolis N., Rosenbluth A.W., Rosenbluth M.N., Teller A.H. and Teller E. (1953) "Equation of state calculation by fast computing machines". *J. of Chem. Phys.*, **21**, 1087-1091.

Şahin R., Ertoğral K and Türkbey, O. (2010) "A simulated annealing heuristic for the dynamic layout problem with budget constraint". *Computers and Industrial Engineering*, doi:10.1016/j.cie.2010.04.013.

Seyed-Alagheband, S. A., Fatemi Ghomi, S.M.T. and Zandieh, M. (2010) "A simulated annealing algorithm for balancing the assembly line type II problem with sequence-dependent setup times between tasks", *International Journal of Production Research*, doi:10.1080/00207540903471486.

Varanelli J.M. and Cohoon J.P. (1999) "A fast method for generalized starting temperature determination in homogeneous two-stage simulated annealing systems", *Computers & Operations Research* **26**, 481-503.

Zhang D., Liu Y, M'Hallah R and Leung, S.C.H. (2010) "A simulated annealing with a new neighborhood structure based algorithm for high school timetabling problems", *European Journal of Operational Research* **203(3)**, 550-558.