Simulated Annealing Cooling Schedules for the School Timetabling Problem

David Abramson

School of Computing and Information Technology, Griffith University, Kessels Road, Nathan, Qld 4111

davida@cit.gu.edu.au

Mohan Krishnamoorthy

CSIRO Mathematical and Information Sciences, Private Bag 10, Rosebank MDC, Clayton, VIC 3169.

Mohan.Krishnamoorthy@cmis.csiro.au

Henry Dang

Department of Computer Systems Engineering, Royal Melbourne Institute of Technology,

P.O. Box 2476V, Melbourne, 3001.

First Submitted: 15 July 1996

Revised: 6 November 1997

Abstract

This paper describes the use of simulated annealing (SA) for solving the school timetabling problem and compares the performance of six different SA cooling schedules: the basic geometric cooling schedule, a scheme which uses two cooling rates, and four schemes which allow reheating as well as cooling. The first two of these reheating schemes are variants of the basic geometric cooling schedule. The third and fourth reheating schemes incorporate a method to compute the temperature that is used as the reheating point. Extensive computational results, which are performed and described, show that the fourth reheating scheme produces consistently better quality solutions in less time than the solutions produced by the other schemes.

Keywords: simulated annealing, timetabling problem, cooling schedules

1. Introduction

Simulated Annealing (SA) is a heuristic algorithm. It is based on an analogy with the physical process of annealing, which involves the aggregation of many particles in a physical system as it is cooled. It was first introduced as a technique to solve complex non linear optimisation problems.

SA has general applicability because it performs optimisation without prior knowledge of problem structure or of any particular solution strategy. Traditionally, it has been easy to apply because *transitions* (in the solution space of the problem being solved) are easy to derive. Furthermore, the formulation of a cost function, however complex, is easily obtained in most cases. Transitions (or *moves*) lead to either *uphill* or *downhill* movements in the solution space. In a minimisation problem, moves of the latter type are automatically accepted while moves of the former type are accepted with a certain probability.

Because of its ease of use, SA is an extremely popular method for solving large, practical-sized problems. See Collins, *et al.* [17] and Koulamas, *et al.* [36] for a review of SA applications. For a recent bibliography on SA implementation and applications, see Osman [46].

SA can also be viewed as a probabilistic process in which a control parameter called *temperature* is employed to evaluate the probability of accepting an uphill move (in the case of minimisation). The probability of acceptance of an uphill move of size ΔC at temperature T is denoted by $e^{(\quad)/T}$. Clearly as the temperature decreases, the probability of accepting a large decrease in solution quality decays exponentially toward zero. Therefore, the final solution is optimal when temperature approaches zero, provided we apply a very small decrement in the temperature value between successive iterations.

In general, it is possible to show that, given a schedule which is infinitely long the final solution will be optimal (see, for example, Aarts & Korst [1], Gidas [26], Hajeck [27], Romeo & Sangiovanni-Vincentelli [47] and van Laarhoven & Aarts [51]). However, most of the real world applications of SA to solving large combinatorial problems employ a finite, and fast, decrement of temperature in order to obtain a solution in a reasonable amount of time.

The process of decrementing the temperature is referred to as the *cooling schedule*. A discussion of a number of different schedules can be found in the literature (see, for example, Aarts & Korst [1], Collins, *et al.* [17], Eglese [25] and van Laarhoven & Aarts [51]). In order to improve the performance of polynomially bounded cooling schedules, a number of techniques have been developed around the central idea of extending the neighbourhood of a configuration in the hope of escaping from a local optimum. Some examples of these are the introduction of a complicated neighbourhood structure, the use of sophisticated transition mechanisms and cooling schedules involving re-heating as well as cooling Lundy & Mees [39]. Some proposals (for example Ingber [30] and Ingber & Rosen [31]) have used different acceptance probability criteria in order to attain fast convergence to a good solution.

Abramson & Dang [6] apply simulated annealing to the school timetabling problem. The paper describes a scheme for representing the scheduling problem as one of placing a number of teaching *tuples* into periods of the week so that the number of clashes for resources (e.g. teachers, rooms, classes, *etc.*) is minimised. SA proved to be an effective method of solving a number of difficult test data sets as well as some real world school timetabling problems. The algorithm uses a commonly adopted cooling schedule, called geometric cooling schedule. In this paper, we extend the earlier work by describing a number of different cooling schedules and comparing their performance on the school timetabling problem. Whilst all of these cooling schedules are polynomially bounded in time, their performance and the quality of the final solution varies considerably.

For a better understanding of the several aspects of the working of an SA algorithm, see, for example, Aarts, *et al.* [2],[3], Casolta, *et al.* [14], Cerny [15], Chyan, *et al.* [16], Kang, *et al.* [33], Kirkpatrick, *et al.* [34],[35] and Leong & Liu [37].

2. The School Timetabling Problem

Timetable scheduling is the problem of allocating various combinations of class, teacher, subject and room onto periods of the week. The goal is to schedule these in such a way that there are no clashes. In other words, no teacher, class or room should appear more than once in any one time period. Real world timetabling problems have many other complexities such as: scheduling double periods, spreading the teaching load evenly throughout the week, and so on.

The problem has attracted a great deal of attention over the years, and many different solutions have been proposed. In the past, exact and heuristic solution approaches for the school (and university) timetabling problem have been proposed by, for example, Abramson [4], Akkoyunulu [8], Britton & Forest [13], Costa [20], Dempster [21], De Gans [22], DeWerra ([23], [24]), Hertz ([28], [29]), McDiarmid [40], Tripathy [49], Ulph [50] and White [52]. Because of the size of real problems, almost all effective solutions are heuristic in operation, and thus do not guarantee optimality. For example, in Abramson [4] it was shown that simulated annealing can be used to obtain good solutions to both artificial and real world timetabling problems.

In its simplest form the scheduling task consists of mapping *class*, *teacher* and *room* combinations (which have already been pre-allocated) onto time slots. We define a *tuple* as a particular combination of identifiers such as class, teacher and room, which is supplied as an input to the problem. The problem now becomes one of mapping of tuples onto period slots such that tuples which occupy the same period slot are disjoint (have no identifiers in common). If tuples are assigned arbitrarily to periods, then in anything but the most trivial cases, a number of *clashes* will exist. We can use the number of clashes in a timetable as an objective measure of the quality of the schedule. Thus, we adopt the number of clashes as the cost of any given schedule.

It is simple to measure the cost of a schedule. For each period of the week, we make a count of the number of occurrences of each class, teacher and room identifier. If the count is less than 2, then the cost is zero. Otherwise, it equals the number of occurrences less 1. The cost of the entire timetable is the sum of each of the individual costs. This procedure is discussed in more detail in Abramson [4]. Pictorially, the mapping of tuples to periods is shown in Figure 1.

The SA algorithm operates by moving tuples out of one period and inserting them into another. The change in cost that is caused as a result of such a movement (or, move or transition) is computed. A decrease in cost is accepted, and an increase is only accepted if it satisfies the Boltzmann equation. In order to compute the change in cost, it is not necessary to recompute the cost of the entire schedule, but only the change caused by the tuple which is moved.

The following algorithm makes it possible to compute this change very efficiently. An auxiliary scoreboard of the number of occurrences of each class, teacher and room identifier is maintained for each period. When a tuple is moved, the count is decremented for each identifier in the source period, and incremented for the destination period. If any of the count values in the source period fall below 2, then a change in cost of -1 is counted for each identifier. If the count values in the destination period rise above 1 then a cost of +1 is counted for each identifier. These two cost values are added to give a total change in cost for moving the tuple. Given this scheme, it is possible to compute the change in cost quickly and calculate the probability of acceptance using an exponential function. Many more details are discussed in Abramson [4].

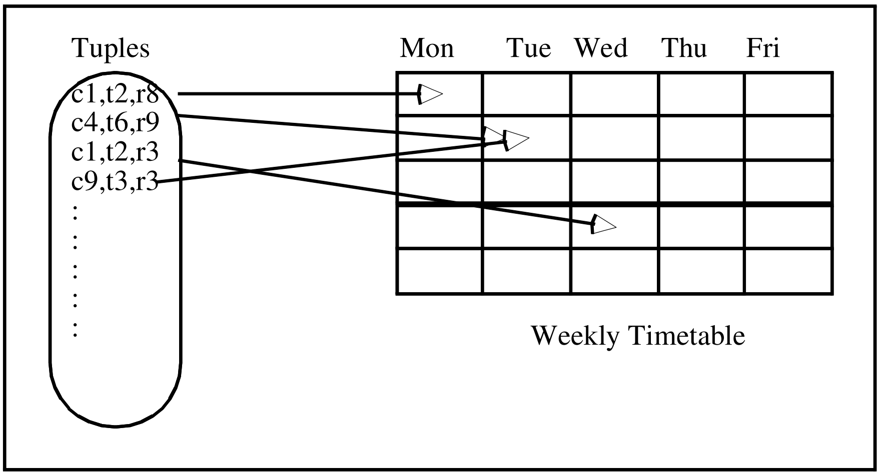


Figure 1 Mapping Tuples onto periods

The algorithm published in Abramson [4] was tested on a variety of simulated and real data and performed reasonably well. However, all of the results were achieved using a simple geometric cooling schedule. Whilst simple to implement, a disadvantage of the geometric scheme is that it is unable to escape from local minima once the temperature has become too low. More recently, a number of alternative, competing, schedules have been suggested in the literature. In this paper we trial the performance of some of these and also introduce a new scheme. All of these schemes are tested on the timetabling problem.

3. Cooling Schedules

In this section we explore six different cooling schedules. First, we introduce the most commonly used scheme, geometric cooling. Second we discuss a modification of this which uses two different cooling rates depending on the temperature value and state of the search space. Third we propose a further modification to geometric cooling, which allows the system to escape from local minima by reheating. Fourth, we propose a further modification which causes reheating to occur by a different trigger mechanism. Fifth, we use a reheating scheme proposed in [45] in which the temperature is raised to either half of its former level (since the immediate previous reheat) or to the temperature at which the best solution was obtained. Finally, we propose a scheme which raises the temperature to a new value based on the quality of the solution at that point in time.

A cooling schedule consists of: (i) *Starting temperature*; (ii) *Cooling factor*; (iii) *Markov chain length*; and (iv) *Process termination condition*.

All of the cooling schedules which are considered in this paper share the same procedures for (i) and (iii) mentioned above. The starting temperature is as defined by White [53], who proposed that the initial temperature should be such that the expected cost at this temperature is just within one standard deviation of the mean cost. For simplicity, we used a constant Markov chain length. This allows us to compare, for each cooling schedule, the solution quality obtained, independent of the Markov chain length. There are two different termination criteria that we had to implement. For the first two cooling schedules it is only necessary to check if there has been no cost improvement for some small number of consecutive Markov chains. However, in the last four schemes reheating occurs when a local minima is detected. Thus, it is not possible to use a simple termination criteria. Consequently, these schemes terminate when some specified time limit is exceeded.

3.1 Geometric cooling schedule

Geometric cooling is described by the temperature-update formula: $T_{k+1} = \alpha T_k$. The *cooling factor,* α, is assumed be a constant and less than one (see Bonomi & Lutton [10],[11],[12], Morgenstern & Shapiro [42], Sechen & Sangiovanni-Vincentelli [48], van Laarhoven & Aarts [51]). This scheme is probably the most commonly used in the SA literature, and acts as a base line for comparison with other more elaborate schemes. Some typical examples of its use can be found in Johnson, *et al.* [32], Sechen & Sangiovanni-Vincentelli [48]. Typical value of α for moderately slow cooling rates are 0.9 through 0.99. It is clear that this simple cooling scheme does not take account of the state of the system in the process of searching the solution space. Thus, in using such a scheme, it would be difficult to adaptively modify the intensity of the search depending on the difficulty of the problem.

3.2 Multiple cooling rates

The simple geometric cooling scheme always applies the same cooling rate regardless of the state of the system. In physical systems, it is known that a substance may undergo a number of phase transitions during cooling, before it reaches its frozen state (see, for example, Kirkpatrick, *et al.* [34],[35]). At each phase transition the physical structure of the substance undergoes a change which alters its behaviour. In combinatorial optimisation problems, phase transitions can be observed when sub-parts of the problem are resolved. At high temperatures the gross structure of the solution is resolved, and at low temperatures the minute details of the solution are resolved. This behaviour of SA has been observed by Lister [38] for the travelling salesman problem (TSP).

The geometric cooling scheme described in the previous section applies the same cooling factor regardless of the phase. However, at high temperatures almost all proposed interchanges are (mostly) accepted, even though many of them could be non-productive. The application of a different cooling rate, that depends on the phase, would allow the algorithm to spend less time in the high temperature phases. Consequently, more time would be spent in the low temperature phases, thus reducing the total amount of time required to solve a particular problem. By the same token, the high temperature phase cannot be omitted because it is responsible for setting up the gross structure of the solution.

In order to compute the temperature at which a phase transition occurs it is necessary to calculate the *specific heat* of the substance. A phase transition occurs when the specific heat is maximal, and this triggers the change in the state ordering. It is possible to compute the specific heat by observing the temperature at which the expression, $\frac{\sigma^2_C(\ )}{T^2}$ is maximal (In Figure 2 this is referred to as the Scaled Variance). In practice, we merely need to calculate the variance of the cost function ($\sigma^2_C(\ )$) over a number of trials at a particular temperature, T. The background theory of derivations are available in van Larrhoven & Arts [51]. Figure 2 shows a plot of the specific heat against temperature for a real scheduling problem. The specific heat occurs at the point where the above expression is maximal. This general approach can be used regardless of the exact nature of the optimisation problem as discussed in van Larrhoven & Arts [51].

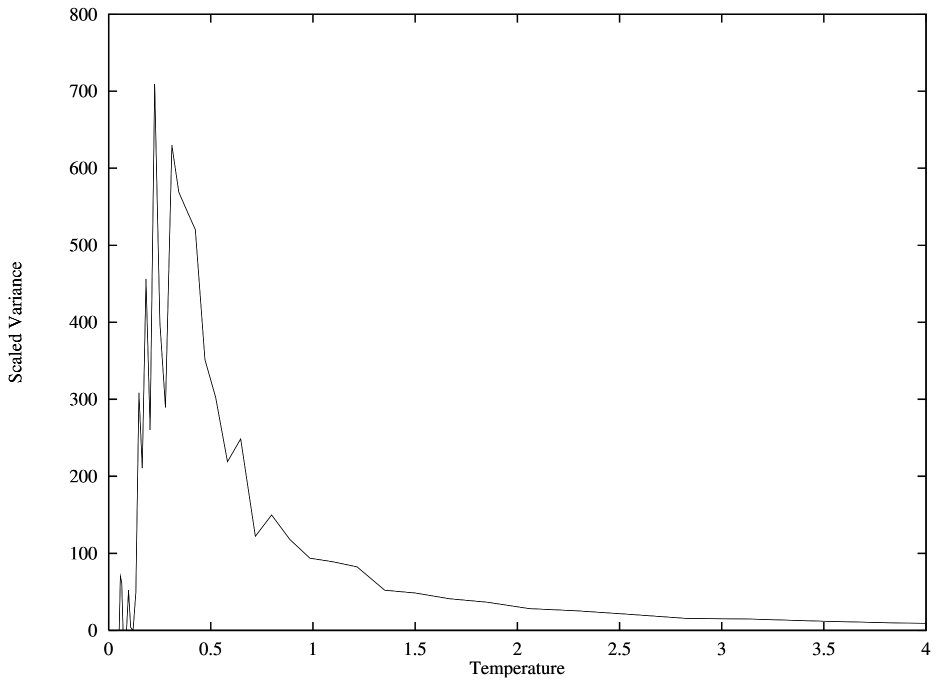


Figure 2 Plot of specific heat against temperature

The new cooling schedule is defined below. The temperature is reduced geometrically in this scheme too. However, two different cooling rates, α and β, are applied depending on whether the system is above or below the temperature at which the specific heat is maximal:

$T_{k+1} = \alpha T_k$ if $T_k > T_{msp}$

$T_{k+1} = \beta T_k$ *if* $T_k \le T_{msp}$

where T_{msp} *is the temperature at which the maximum specific heat occurs;*

α *and* β *are constants* < 1, *and*

$\alpha < \beta$.

As in the simple geometric cooling scheme, typical values of α might be 0.5 for a fast cooling rate, and then values of 0.9 or 0.99 for β, yielding much slower cooling after the transition.

3.3 Geometric reheating

The previous two sections have described cooling schedules in which the temperature is only ever reduced. Once the temperature reaches a particular level, the system will not be able to accept cost increases that are sufficiently large. However, these large increases may be required in order for the SA to proceed to the global optimum. Thus, no matter how slow the cooling rate, from that point on, the algorithms will always deliver a local minima. The only solution to this problem is to detect that a local minima has been reached, and to *reheat* the system, thus allowing it to escape from local minima. In this section we describe a simple reheating scheme for which can be expressed as:

***Cooling**:* $T_{k+1} = \alpha\, T_k$

Heating: $T_{k+1} = T_k/\beta$

where, α *and* β *are constants, less than 1.*

In this scheme, it is necessary to determine when to swap between heating and cooling. Initially we experimented with a simply strategy. Cooling is terminated when no improvement in cost is detected after a small number of temperature decreases. At this point, heating is commenced. Heating is terminated when the cost changes (either increases or decreases). Whilst this scheme is simple, it has a potential to get caught in cycles, in which the system gets frozen in a local minima, the temperature is increased just enough to allow it to escape and then cooling is resumed, at which stage it proceeds immediately to the same local minima. In this scheme α and β might have the same value, typically 0.9 or 0.99, although it is possible to use a lower value of β for faster reheating than cooling. In the next section we propose a variant of this stopping criteria which helps avoid cycling behaviour.

3.4 Enhanced geometric reheating

This scheme is an enhanced version of the geometric reheating method. It is introduced to prevent excessive heating of the system and to overcome cycling which occurs between heating and cooling. As with the previous heating scheme, an arbitrary chosen value of the heating factor, β, might result in either excessive heating or under heating. Excessive heating increases the total amount of time to solve a problem due to the introduction of non productive moves, while under heating could result in cycling within a local minimum. Therefore it is necessary to choose a good value for the heating factor. The method we chose to determine an effective heating factor involves a learning process. This is achieved by assigning a large heating value which is close to 1 every time a local optimum is encountered. This value is then decreased if, after a fixed number of Markov chains, the system is still trapped in the local minima. The following description summarises this process:

$\beta = \beta$*_default*

***Cooling**:* $T_{k+1} = \alpha\, T_k$

Heating: $T_{k+1} = T_k/\beta$

if Count(local_optima) > Fixed_number then $\beta = \beta - \varepsilon$

where, α *is a constant less than 1;*

β*_default is a large constant less than 1;*

$\varepsilon << \beta$*_default*

In this scheme, if β has a value of 0.9, then a suitable value for ε might be 0.1 so that β is adjusted slowly. The heating termination condition is now modified so that a constant change in cost between successive Markov chains will not terminate the heating sequence. This eliminates the problem of local minima cycling.

The following expression summarises the termination condition:

if *($\Delta C_k > 0$)* ***and*** *[($\Delta C_{k+1} = -\Delta C_k$)* ***or*** *($\Delta C_{k+1} = -2\Delta C_k$)]* ***then*** *fluctuation =* ***true***

where $\Delta C_k = C_k - C_{k-1}$

$\Delta C_{k+1} = C_{k+1} - C_k$

C_k *is the cost at the end of the k'th Markov chain*

Although this scheme provides a general way of reheating the system based on the system state, there is still an overhead in slowly increasing the temperature to a point at which a new neighbourhood can be explored. The next section introduces a heating technique which is based on both the solution quality and the phase transition properties of the system.

3.5 Non-monotonic cooling schedule

In the geometric reheating schemes discussed above, the temperature is slowly raised after a local minima has been detected. A potential disadvantage of this approach is that it may take a number of cycles before the temperature is sufficiently high to allow significant changes to occur. In the non-monotonic cooling regime the temperature is reset in one step, thus removing the non productive cycles.

Whenever the occurrence of a local minima is detected, this scheme attempts to escape from it by resetting the temperature to the maximum of either half the previous reset temperature or the temperature at which the best solution was attained.

This scheme starts from an initial temperature, which is decremented after each Markov chain according to a standard temperature decrement rule. When, at an iteration k, a local minima is detected (a cycle of search is made in which no transition is accepted), the temperature at iteration $k+1$, T_{k+1}, is updated according to the following rule: $T_{k+1} = \max[T_{r/2}, T_b]$. Here, T_r is the immediately previous temperature reset value (or, the initial temperature, if the temperature is reset for the first time). T_b is the temperature corresponding to the best solution attained. The algorithm stops when a certain number of resets have been performed.

A version of this scheme has been applied by Osman & Christofides [43] for the capacitated clustering problem, Connolly [18] for the quadratic assignment problem, Connolly [19] for a general purpose SA algorithm, Osman [45] for the vehicle routing problem and Osman [44] for the generalised assignment problem. In his applications, Connolly ([18], [19]) resets the temperature only once (to that of T_b) in the entire run. In his applications, Osman ([43], [44], [45]) uses the temperature reduction scheme described in Lundy & Mees [39]. This scheme assumes a Markov chain length of 1. Moreover, Osman uses a different starting temperature to the one proposed in this paper. We obtain the initial temperature as defined by White [53].

We use a Markov chain length that depends on the problem size and a geometric temperature decrement rule. Moreover, we employ the above reheating scheme in conjunction with random transition mechanism (as opposed to a systematic neighbourhood transition mechanisms).

We have modified the original applications of the non-monotonic scheme because this allows us to compare, the performance of the cooling schedules, independent of the method of obtaining the initial temperature, the Markov chain length and the temperature decrement rule.

3.6 Reheating as a function of cost

In this section we describe another method of reheating which directly sets the temperature to a certain value without the need to approach the desired temperature slowly. This scheme also allows the system to escape local minima. As in the non-monotonic scheme, it avoids the wasted cycles that occur during heating. The temperature value is a function of both the quality of the current best configuration and the temperature at which the maximum specific heat occurs. Thus, as the quality of solution improves, the system is reheated less.

In section 3.2 we stated that the high temperature phase cannot be omitted due to its role in creating the super-structure of the solution. The phase transition is the point at which the system begins to solve the sub problems; when the super-structure of the solution is fixed. Consequently, the phase transition temperature acts as a lower bound on the temperature to which the system should be reheated. If it is any lower than this value, then the system cannot escape from local minima which are caused by poor sub-structures. Thus, unlike the previous reheating schemes which attempt to approach the reheating temperature slowly from below, we propose to reset the temperature in one operation.

However, there may be cases when the temperature needs to be set to an even higher level. If the best solution found to date is poor (has a high cost) then the super-structure may require re-arrangement. This can only be performed by raising the temperature to a level which is even higher than the one at which phase transition occurs. In general, the higher the best cost, the higher the temperature which is required for the algorithm to escape the local minima. From the above discussion, the reset temperature should be a function of both the current best configuration and the temperature at which the system changes state. Although we have experimented with the simple linear relationship given below, other complex forms are also possible:

$$T_{new} = P\ C(x^b) + T_{msp}$$

where, T_{new} is the reset temperature;

T_{msp} is the temperature at which the maximum specific heat occurs;

P is a constant > 0;

$C(x^b) > 0$ is the current best cost.

It is difficult to give any general rules for the values of P since it is so problem specific. However, in most of our work we used very low values so that the temperature did not rise too much above T_{msp}. We calculate T_{msp} as the temperature at which the maximum specific heat is obtained. The method for obtaining this temperature is described in section 3.2.

In this scheme it is important to detect local minima. We use the same technique as described in the geometric reheating scheme, described in section 3.4.

4. Computational Results

It is possible to generate a number of test problems randomly. Tuples are built by choosing random class, teacher and room identifiers, and are then packed into a period which can accommodate them. If the tuple cannot be placed in an available period, then the tuple is discarded and another one generated. This process continues until all class, teacher and room identifiers have been used. The resulting schedule has no spare slots and uses all of the identifiers, and thus is *tightly constrained*. The tuples are then deallocated from the period slots and are reallocated using the simulated annealing algorithm.

For the computational comparisons in this paper we generated a number of test data sets containing from 150 to 390 tuples each with 30 periods, as shown in Table 1. The problems with more than 300 tuples (10 classes, 10 teachers and 10 rooms) are quite difficult to solve as verified by a separate heuristic which we tested in another study Abramson & Dang [6]. The advantage of using randomly generated synthetic data is that it is possible, via suitable parameters, to construct a carefully controlled experiment. Therefore, we have varied the number of classes, teachers and rooms to generate a variety of problems of varying complexity. The interested reader is referred to Abramson [4] for details of the application of SA to a real timetabling problem. All of these results were obtained by an application-specific attached processor which was designed specifically for the timetabling problem. More implementation details can be found in Abramson [5].

Table 1 Problem Characteristics

Problem Number	# Tuples	# Classes	# Teachers	# Rooms
1	150	5	5	5
2	180	6	6	6
3	210	7	7	7
4	240	8	8	8
5	270	9	9	9
6	300	10	10	10
7	330	11	11	11
8	360	12	12	12
9	390	13	13	13

In this section we summarise the results of comparing the 6 different cooling schemes on the timetabling problems described in the previous section. Table 2 compares the results of the 6 schemes. It shows the average (final) solution obtained by the 6 schemes across six runs, and the average time taken to reach that solution. We have reported the average achieved for each scheme regardless of the time taken. For each of the 6 schemes we start at high initial temperatures (as discussed in section 3). Therefore, each of the six runs will start with a different (random) initial configuration. Graph 1 shows these results in a graphical form.

These results allow some comparisons to be made between schemes. It can be seen that the multiple cooling rate scheme tends to either produce lower cost solutions or produce a solution of similar cost to the geometric cooling schedule in less time. The geometric reheating schemes solved more problems than either of the geometric cooling schemes, and where they did not solve the problem, they produced lower cost solutions. The reheating scheme that is based on cost solved even more problems, although it took quite a long time for some of them to complete. The only problem it failed to solve was the largest. This set of results clearly shows that the cost based reheating scheme is capable of finding the global minima much more reliably than any of the other schemes.

In Tables 3 and 4 we show two different ways of comparing the six schemes.

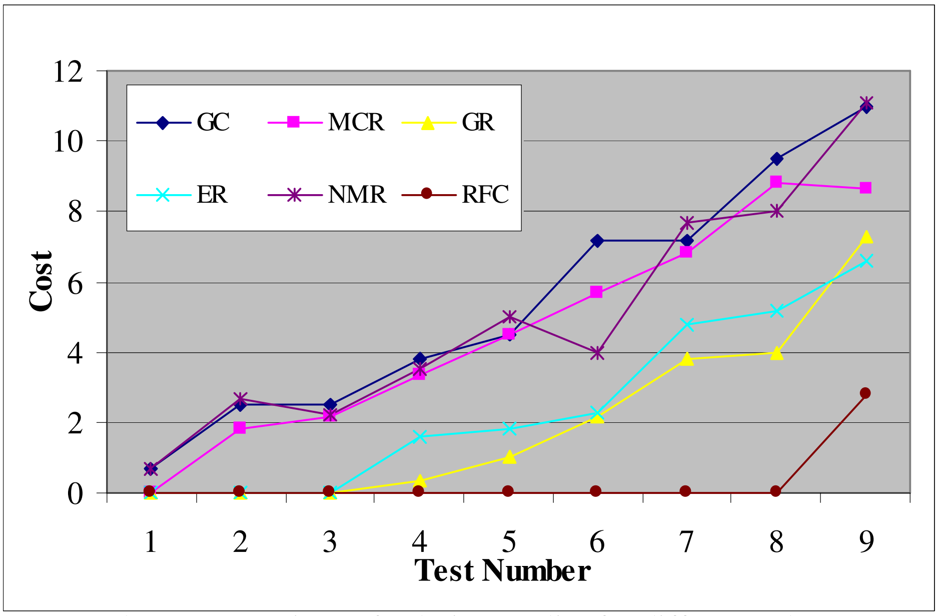
In Table 3 the time is reported, for each schedule, for roughly the same solution quality. At the end of a Markov chain of transitions, we perform a check to see if the solution quality is just below the prescribed threshold. If so, we stop and report the time. We have chosen some arbitrary costs at which to make these comparisons. These results are summarised in Graph 2.

From Table 3 it can be seen that the geometric scheme and the non-monotinic reheating scheme take the longest time to achieve a particular quality of solution, and therefore they have been excluded from Graph 2. An examination of the data in Graph 2 reveals that reheating as a function of cost generally performs best in terms of time to achieve the same quality. The performance of the other schemes is less consistent over the entire data set and seems to be problem-dependent, although enhanced reheating does seem slightly worse than geometric reheating and multiple cooling rates.

Table 2 Comparison of 6 schemes, time and cost varying

	Cooling Schemes				Reheating Schemes							
	Geometric Cooling(GC)		Multiple Cooling Rate (MCR)		Geometric Re-heating (GR)		Enhanced Re-heating (ER)		Non-monotonic Reheating (NMR)		Reheating as function of cost (RFC)	
Test	Ave. Cost	Time (Secs)	Ave. Cost	Time (Secs)	Ave. Cost	Time (Secs)	Ave. Cost	Time (Secs)	Ave. Cost	Time (Secs)	Ave. Cost	Time (Secs)
1	0.67	72.45	0	97.2	0	35.2	0	68	.67	1371	0	9
2	2.5	80.22	1.8	52	0	151.5	0	141	2.7	145	0	32
3	2.5	6635	2.17	46.9	0	205	0	250	2.2	1460	0	63
4	3.8	7859	3.33	1043	0.33	423	1.6	782	3.5	1491	0	112
5	4.5	19167	4.5	111	1	580	1.8	806	5	1662	0	271
6	7.17	1647	5.67	2046	2.17	1603	2.3	1522	4	3739	0	2398
7	7.17	1922	6.83	1191	3.8	1611	4.8	1601	7.7	3196	0	7555
8	9.5	2075	8.8	1342	4	1604	5.17	1602	8	3266	0	18375
9	11	2320	8.67	279	7.3	1604	6.6	1603	11.1	3389	2.8	21224

In Table 4 the solutions are compared, given the same processing time for each of the schemes. Again, the choice of time is arbitrary, but it is relatively short. At the end of a Markov chain of transitions, we perform a check to see if the solution time is just above the prescribed threshold. If so, we stop and report the solution. We have chosen some arbitrary costs at which to make these comparisons. The results are displayed graphically in Graph 3.



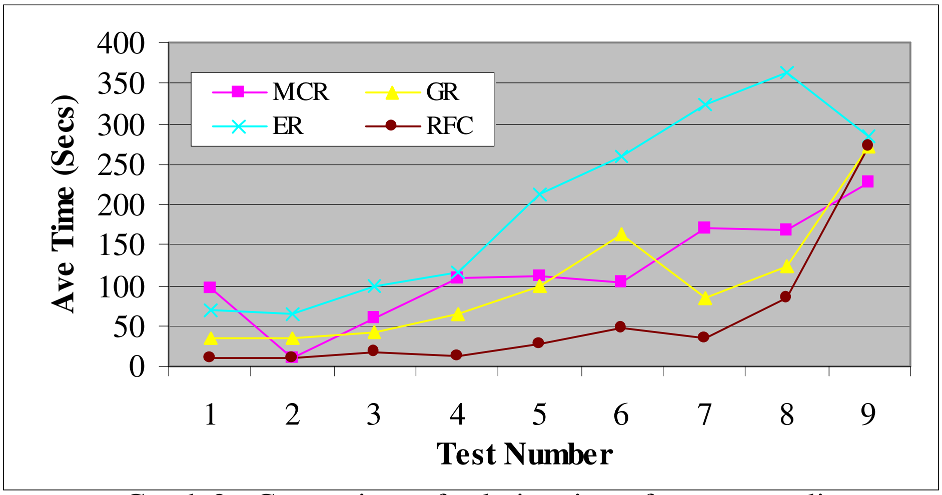
Graph 1 - Comparison of solution quality for different schemes.

Many of the schemes do not produce zero cost solutions in Table 4. It can be seen in Graph 3 that the multiple cooling rate always produces a lower cost solution than the single geometric cooling scheme. The non-monotonic reheating scheme does not perform very differently to the simple geometric scheme. In fact, the non-monotonic scheme produces a poorer solution in most cases. Thus, given the same amount of time, the slower cooling rate is of more value than the reheating provided by the non-monotonic scheme.

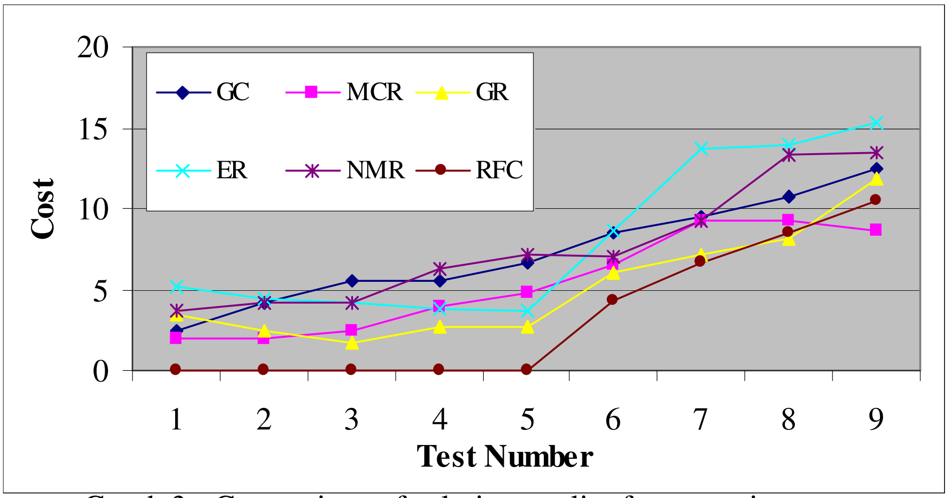
One noticeable feature is that reheating as a function of cost performs exceedingly well, given a set time. It is almost always better than every other scheme and delivers the optimal solution in five out of the nine test problems. No other scheme delivers the optimal solution for this set of five problems.

Table 3 Comparison of six schemes - Time for same quality

	Cooling Schemes				Reheating Schemes							
	Geometric Cooling		Multiple Cooling Rate		Geometric Reheating		Enhanced Reheating		Non-monotonic Reheating		Reheating as function of cost	
Test	Ave. Cost	Time (Secs)	Ave. Cost	Time (Secs)	Ave. Cost	Time (Secs)	Ave. Cost	Time (Secs)	Ave. Cost	Time (Secs)	Ave. Cost	Time (Secs)
1	0.67	731.4	0.0	97.2	0.0	35.2	0.0	68.8	.67	1371	0.0	9.0
2	2.5	80.22	2.83	10.3	2.5	34.9	2.5	63.3	2.67	143	2.67	8.68
3	2.5	6635	2.5	58.6	2.5	42.9	2.83	98.3	2.2	1466	2.33	16.8
4	3.83	7859	4.0	109.4	3.67	64.4	3.83	115.2	3.5	1491	3.83	12.85
5	4.5	19167	4.5	111.7	4.17	99.2	4.5	212.8	5	1662	4.33	28.2
6	7.17	1648	7.33	103.3	7.0	164.0	7.5	258.6	7	376	7.0	45.5
7	9.5	220.8	9.17	171.0	9.33	83.6	9.33	323.0	9	3172	9.33	35.1
8	9.5	2076	9.17	167.3	9.67	123.1	9.5	363.2	9.7	3270	9.5	83.13
9	11.0	2320	10.5	227.83	11.83	271.5	11.17	283.1	11.3	3384	10.5	271.1



Graph 2 - Comparison of solution times for same quality



Graph 3 - Comparison of solution quality for same time

Table 4 Comparison of six schemes - Quality for same time

	Cooling Schemes				Reheating Schemes							
	Geometric Cooling		Multiple Cooling Rate		Geometric Re-heating		Enhanced Re-heating		Non-monotonic Reheating		Reheating as function of cost	
Test	Ave. Cost	Time (Secs)	Ave. Cost	Time (Secs)	Ave. Cost	Time (Secs)	Ave. Cost	Time (Secs)	Ave. Cost	Time (Secs)	Ave. Cost	Time (Secs)
1	2.5	10.23	2	9.38	3.5	10.78	5.17	10.7	3.7	9.5	0	9.0
2	4.17	34.5	2	34.5	2.5	34.9	4.5	34.8	4.2	34	0	32.4
3	5.5	58.3	2.5	58.6	1.67	64.9	4.17	66.9	4.2	56.7	0	63.4
4	5.5	114.5	4.0	109.4	2.67	115.0	3.83	115.2	6.3	110	0	112.4
5	6.67	267.1	4.83	296.2	2.67	255.6	3.67	258.5	7.2	286	0	271.9
6	8.5	221.9	6.5	216.6	6.0	211.0	8.67	211.1	7	210	4.33	211.2
7	9.5	220.8	9.17	171.0	7.17	163.0	13.67	167.0	9.3	177	6.67	163.4
8	10.67	181.0	9.17	167.3	8.17	171.1	14.0	171.4	13.3	162	8.5	171.6
9	12.5	276.5	8.67	279.4	11.83	271.5	15.33	271.4	13.5	279	10.5	279.4

5. Conclusions

In this paper we have compared six cooling schedules for simulated annealing, and have examined their effectiveness experimentally using a difficult scheduling problem. In summary, the use of multiple cooling rates is more effective than one. The point at which cooling rates are switched is the temperature at which the specific heat is maximal to denote a phase change. We experimented with four schemes which raise the temperature in order to allow for the system to escape from local minima. In general, the scheme which uses the phase transition temperature in combination with the best solution quality found to date produced the best results. In this paper we have only tested the schemes with the timetable scheduling problem, and therefore, further research is required to test the effectiveness of the cooling schedules over a wider range of problem classes. We have experimented with the reheating schemes on another problem (set partitioning) and found it quite effective, and these combined results are reported elsewhere (see Abramson, et al [7]).

Acknowledgments

Henry Dang was supported by an Australian Postgraduate Research Award (Industry) and Computer Techniques P/L. Thanks go to Mr. John Shaw of Computer Techniques for his assistance in this work. Thanks also to Andreas Ernst and David Sier of CSIRO for their proof reading of the manuscript.

References

[1] Aarts E.H.L. and Korst. J., *Simulated Annealing and Boltzmann Machines,* John Wiley, 1989.

[2] Aarts E.H.L, de Bont F.M.J., Habers J.H.A. and van Laarhoven P.J.M, "A Parallel Statistical Cooling Algorithm", *Proceedings STACS 86, Springer Lecture Notes in Computer Science*, **210**, 87-97, 1986.

[3] Aarts E.H.L., de Bont F.M.J., Habers J.H.A. and van Laarhoven P.J.M., "Parallel Implementations of the Statistical Cooling Algorithm", *Integration*, **4**, 209-238, 1986.

[4] Abramson D.A., "Constructing School Timetables using Simulated Annealing: Sequential and Parallel Algorithms", *Management Science*, **37**(1), 98-113, 1991.

[5] Abramson D.A., "A Very High Speed Architecture to Support Simulated Annealing", *IEEE Computer*, May 1992.

[6] Abramson D.A. and Dang H. "School Timetables: A Case Study in Simulated Annealing", in *Lecture Notes in Economics and Mathematical Systems: Applied Simulated Annealing*, Rene V.V. Vidal (Ed.), 103-124, 1993.

[7] Abramson D.A., Dang H. and Krishnamoorthy, M., "A Comparison of two Methods for Solving 0-1 Integer Programs using a General Purpose Simulated Annealing Algorithm", *Annals of Operations Research*, **63**, 129-150, 1996.

[8] Akkoyunulu E. A., "A Linear Algorithm for Computing the Optimum University Timetable", *Computer Journal*, **16**(4), 347-350, 1973.

[9] Aust R. J., "An Improvement Algorithm for School Timetabling", C*omputer Journal*, **8**, 135-146, 1976.

[10] Bonomi E. and Lutton J.L., "The N-City Travelling Salesman Problem: Statistical Mechanics and The Metropolis Algorithm", *SIAM Review*, **26**, 551-568, 1984.

[11] Bonomi E. and Lutton J.L., "The Asymptotic Behaviour of Quadratic Sum Assignment Problems: A Statistical Mechanics Approach", *European Journal of Operations Research,* 1984.

[12] Bonomi E. and Lutton J.L., "Simulated Annealing Algorithm for The Minimum Weighted Perfect Euclidean Matching Problem", *R.A.I.R.O Recherche Operationelle*, **20**, 177-182.

[13] Britton J. N. G. and Forest F. J. M., "College Timetable Construction by Computer", *Computer Journal*, **14**, 361-365, 1971.

[14] Casotta A., Romeo F. and Sangiovanni-Vincentelli A.L., "A Parallel Simulated Annealing Algorithm for the Replacement of Macro-Cells", *Proceedings IEEE International Conference on Computer Aided Design*, Santa-Clara, 30-33, November 1986.

[15] Cerny V.,"Multiprocessor Systems as a Statistical Ensemble: A Way Towards General Purpose Parallel Processing and MIMD Computers?", Comenius University, Bratislava, unpublished manuscript, 1983.

[16] Chyan D.J. and Breuer M.A., "A Placement Algorithm for Array Processors", *Proceedings 20th Design Automation Conference*, Miami Beach, 182-188, June 1983.

[17] Collins N.E., Eglese R.W. and Golden B.L., "Simulated Annealing: An Annotated Bibliography", *American Journal of Mathematical and Management Sciences*, **8**(3-4), 209-307, 1988.

[18] Connolly D., "An Improved Annealing Scheme for the Quadratic Assignment Problem", *European Journal of Operations Research*, **46**, 93-100, 1990.

[19] Connolly D., "General Purpose Simulated Annealing", *Journal of the Operational Research Society*, **43**, 494-505, 1992.

[20] Costa D., "A Tabu Search Algorithm for Computing an Operational Timetable", *European Journal of Operational Research*,**76**, 89-110, 1994.

[21] Dempster M. A. H., "Two Algorithms for the Timetabling Problem", in *Combinatorial Mathematics and Its Applications*, D. J. A. Welsh (ed.), Academic Press, New York, 63-85, 1971.

[22] De Gans O. B., "A Computer Timetabling System for Secondary Schools in the Netherlands", E*uropean Journal of Operations Research*, **7**, 175-182, 1981.

[23] DeWerra D., "Construction of School Timetables by Flow Method", *INFOR*, **9**, 12-22, 1971.

[24] DeWerra D., "Notes - Some Comments on a Note about Timetabling", *INFOR*, **16**, 90-92, 1978.

[25] Eglese R.W., "Simulated Annealing: A Tool for Operational Research", *European Journal of Operational Research*, **46**, 271-281, 1990.

[26] Gidas B., "Nonstationary Markov Chains and Convergence of the Annealing Algorithm", J*ournal of Statistical Physics*, **39**, 73-131, 1988.

[27] Hajeck B., "Cooling Schedule for Optimal Annealing", *Mathematics of Operations Res*earch, **13**, 311-329, 1988.

[28] Hertz A., "TAbu Search for Large Scale Timetabling Problems", *European Journal of Operational Research*, **54**, 39-47, 1991.

[29] Hertz A., "Finding a Feasible Course Schedule with Tabu Search", *Discrete Applied Mathematics*, **35**, 255-270, 1992.

[30] Ingber L., "Very fast Simulated Re-Annealing", *Mathematical &Computer Modelling*, **15**, 77-78, 1991.

[31] Ingber L. and Rosen B., "Genetic Algorithms and Very Fast Simulated Reannealing: A Comparison", *Mathematical & Computer Modelling*, **16**(11), 87-100, 1992.

[32] Johnson D.S., Aragon C.R., McGeoch L.A. and Schevon C., "Optimization by Simulated Annealing: an Experimental Evaluation", List of Abstracts, *Workshop on Statistical Physics in Engineering and Biology*, Yorktown Heights, 1986.

[33] Kang Le, Belanger P., White G. and Con Schoenberg G., "Computerized Scheduling", P*roceedings of the 36th Annual College and University Users Conference*, Louisville, Kentucky, April 1991.

[34] Kirkpatrick S., Gelatt Jr. C.D. and Vecchi M.P., "Optimization by Simulated Annealing", *IBM Research Report RC 9355*.

[35] Kirkpatrick S., Gelatt Jr. C.D. and Vecchi M.P., "Optimization by Simulated Annealing", *Science*, **220** 671-680, 1993.

[36] Koulamas C., Antony S. R. and Jaen R., "A Survey of Simulated Annealing Applications to Operations Research Problems", *Omega*, **22**(1), 41-56, 1994.

[37] Leong H.W. and Liu C.L., "Permutation Channel Routing", *Proceedings IEEE International Conference on Computer Design*, Port Chester, 579-584, 1985.

[38] Lister R., "Annealing Networks and Fractal Landscapes", *Proceedings IEEE International Conference on Neural Networks: Volume 1*, San Franciso, 257-262, March 1993.

[39] Lundy M. and Mees A., "Convergence of An Annealing Algorithm", *Mathematical Programming*, **34**, 111-124, 1986.

[40] McDiarmid C. J. H., "The Solution of a Timetabling Problem", *J. Inst. Math. Appl.*, **9**, 23-34, 1972.

[41] Metropolis N., Rosenbluth A., Rosenbluth M., Teller A. and Teller E., "Equation of State Calculations by Fast Computing Machines", *Journal of Chemical Physics*, **21**, 1087-1092, 1953.

[42] Morgenstern C.A. and Shapiro H.D., "Chromatic Number Approximation Using Simulated Annealing", *Technical Report No. CS86-1*, Dept. of Computer Science, University of Mexico, Alberquerque, 1986.

[43] Osman I. H. and Christofides N., "Capacitated Clustering Problems by Hybrid Simulated Annealing and Tabu Search", *International Transactions in Operational Research*, **1(3)**, 317-326, 1994.

[44] Osman, I. H., "Heuristics for the Generalized Assignment Problem: Simulated Annealing and Tabu Search Approaches", *OR Spektrum*, **17**, 211-225, 1995.

[45] Osman, I. H., "Metastrategy Simulated Annealing and Tabu Search Algorithms for the Vehicle Routing Problem", *Annals of Operations Research*, **41**, 421-451, 1993.

[46] Osman I.H. and Laporte G., "Metaheuristics: A bibliography", *Annals of Operations Research*, **63**, 513-623, 1996.

[47] Romeo F. and Sangiovanni-Vencentelli A.L., "A Theoretical Framework for Simulated Annealing", *Algorithmica*, **6**, 346-366, 1991.

[48] Sechen C. and Sangiovanni-Vicentelli A.L., "The Timber Wolf Placement and Routing Package", *IEEE Journal on Solid State Circuits*, **SC-20**, 510-522, 1985.

[49] Tripathy A., "School Timetabling - A Case in Large Binary Integer Linear Programming", *Management Science,* **30**(12), 1473-1489, 1984.

[50] Ulph A., "Notes - An extension of a Result in School Timetabling", *INFOR*, **15**, 255-257, 1977.

[51] van Laarhoven P.J.M and Aarts E.H.L., *Simulated Annealing: Theory and Applications*, Reidel, The Netherlands, 1987.

[52] White D. J., "A Note on Faculty Timetabling", *Operations Research Quarterly*, **26**, 875-878, 1976.

[53] White S.R., "Concepts of Scale in Simulated Annealing", *Proceedings IEEE International Conference on Computer Design*, Port Chester, 646-651, 1984.