Cooling Schedules for Simulated Annealing Based Scheduling Algorithms

Henry Dang
Department of Computer Systems Engineering
Royal Melbourne Institute of Technology
P.O. Box 2476V,
Melbourne, 3001.
henry@mel.dit.csiro.au

David Abramson
School of Computing and Information Technology
Griffith University
Kessels Rd, Nathan, Qld 4111
davida@cit.gu.edu.au

Abstract

This paper describes the use of simulated annealing for solving combinatorial optimisation problems and then compares 5 different cooling schedules. These are the most commonly used scheme, the geometric cooling schedule, a scheme which uses two cooling rates in combination with a method for detecting when a phase transition occurrs, plus three schemes which allow reheating as well as cooling. The reheating scheme are variants of the geometric cooling schedule plus one which computes a temperature to use as the reheating point. In summary, the third reheating scheme produces better quality solutions in less time than the others. The schemes are evaluated using a complex scheduling problem found in timetable computation.

1. Introduction

Simulated Annealing (SA) is a heuristic algorithm which is based on the analogy of 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 any particular solution strategy. Traditionally, it has been easy to apply because a transition mechanism between configurations is easy to derive and the cost function formulation is usually easily obtained.

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) or decreasing the solution quality. The probability of acceptance an uphill move of size ΔC is denoted by $\exp(-\Delta C/T)$. Clearly as the temperature decreases, the acceptance probability of a large decrease in quality decays exponentially toward zero. Therefore, the final solution is optimal when temperature approaches zero, provided a very small decrement in the temperature value is applied. In general, it is possible to show that given a schedule which is infinitely long it is possible to prove that the final solution will be optimal [1, 23].

However, most of the real world applications of SA to the solving of 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, and a discussion of a number of different schedules can be found in [1, 23]. Because many of the problems of interest are NP-hard, a cooling schedule which is not exponentially large compared to the problem size will produce a sub-optimal solution. For such problems, a polynomial bound on the computational time has been shown to return very poor solutions [1, 6] in hard problems.

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 [20]. Some proposals have derived their own acceptance probability and criteria in order to attain a fast convergence to a reasonable quality solution [13, 14].

This paper describes and compares a number of different cooling schedules. Whilst they are all polynomially bounded in time, their performance and the quality of the final solution varies. The timetabling problem, which has been studies widely with simulated annealing [4, 5, 6], is used as a test problem for these techniques. The new cooling schedules are compared to the standard geometric one [6] and performance results are presented. The paper begins by a discussion of simulated annealing, followed by the different cooling schemes.

2. The SA algorithm

SA can be considered as an enhanced version of a local search algorithm. It performs the local search with an addition of up-hill moves to escape from local minima. In this section we will introduce the algorithm.

SA is based on the analogy of the annealing of a solid. From a physical perspective, annealing denotes the process of cooling a solid from its melting state until crystallisation formed. Atoms are freely to arrange themselves in the liquid phase. As the temperature is lowered, interactive bonding forces between atoms gradually increase their strength which restrict the random motion of atoms. Finally, as temperature approaches zero, the ground state (the lowest energy state) of the solid is obtained, provided melting temperature is sufficient high and the temperature decrease in a fine step to allow thermodynamic equilibrium established.

Starting off at some maximum initial temperature, the solid is cooled. At each temperature, thermodynamic equilibrium is established. Thermal equilibrium is characterised by the probability of the solid (denoted by the stochastic variable **E**) being in the state i with energy E_i is given by the Boltzmann distribution.

$$\mathbf{Pr} = \{ \mathbf{E} = i \} = \frac{1}{Z(T)} \times e^{\frac{-E_i}{kT}} \qquad (1)$$

where k = Boltzmann constant
T = Temperature
$e^{\frac{-E_i}{kT}}$ = Boltzmann factor

Z(T) = Normalisation function

$$= \sum_j e^{\frac{-E_j}{kT}} \quad \forall j$$

Metropolis (1953) proposed a *Monte Carlo method* which simulates the evolution of solid at a fixed temperature to thermal equilibrium [23]. This is achieved by introducing a small random perturbation to transform the current state i of the solid to the next state j. If there is a decrease in state energy, the current state will take the form of state j, otherwise the accepted probability is defined by

$$A_{ij} = e^{(E_i - E_j)/kT}$$

where A_{ij} = acceptance probability of the transition from state i to state j
E_i = energy level corresponding to state i.
E_j = energy level corresponding to state j.

From the equation, it is clear that as the temperature decreases, the Boltzmann distribution concentrates on the lowest energy level. The converse of the annealing is the process called *Quenching* in which temperature decrease rapidly. This results in the freezing of defects, which leads to an imperfect lattice structure of the solid structure with high internal energy. In the next section we discuss the application of annealing to combinatorial optimisation problems.

3. Applying SA to Combinatorial Optimisation

From an optimisation perspective, a search space can be viewed as a physical system in which elements or variables correspond to particles. The atomic bonding force is equivalent to the set of constraints imposed on the problem. The energy of the system is modelled as the penalty cost, so that when the system reaches ground state, the process arrives at the global optimum configuration. In order to keep the analogy with the physical phenomenon, the word *temperature* is still employed instead of *control parameter.*

The Boltzmann distribution and the acceptance probability can now be rewritten as

$$\mathbf{Pr} = \{ \mathbf{X} = i \} = \frac{1}{Q(T)} \times e^{\frac{-C_i}{T}} \qquad (2)$$

$$A_{ij} = e^{(C_i - C_j)/kT} \qquad (3)$$

where X is the solution configuration

A_{ij} is the acceptance probability of the transition from configuration i with the cost C_i to configuration j with cost C_j where $C_j > C_i$

Q(T) is the normalisation constant which depends on T

The Metropolis algorithm can now be performed on a given neighbourhood structure of the problem. The current configuration is randomly transformed into one in its neighbourhood. This mechanism is mathematically described as a Markovian transformation where the probability of the current configuration only depends on the previous one [23].

Initially a starting temperature is computed so that all configurations generated are accepted. At each temperature, a sequence of length M, called Markov chain length, of perturbations is generated by randomly choosing a configuration in the neighbourhood structure of the current configuration. If the change in cost resulted from the transition of the current configuration to the

new one is negative, the new configuration is accepted and the current configuration is updated accordingly. With a positive change in cost, the acceptance probability is computed according to (3). The process will continue with the new configuration if this probability is greater than a random number chosen from a uniform distribution interval [0, 1). Otherwise the original configuration is retained. The temperature is then decreased using a cooling function cool(T). The process continues until the system is frozen or there is no improvement after some small number of iterations. The following pseudo code summarises the SA algorithm.

```
Generate an initial configuration called x
Compute cost of initial configuration C(x)
Compute initial temperature T0
Set T = T0
While (not termination) do begin
        Repeat Markov chain length (M) times
        begin
                Randomly choosing a
                configuration called x' ⊂ Sx
                ΔC = C( x ) - C( x' )
                If (ΔC ≤ 0) or (e^(-ΔC/T) >
                        unif_rand(0,1)) then
                Set  x = x'
        end
        T = cool(T)
end
```

where S_x is the neighbourhood of configuration x
unif_rand(a,b) returns a random uniformly distributed integer between a and b
cool(T) returns a new value for the temperature after applying a cooling operation to T

There are four important operations performed in the above algorithm. They are the mechanism to generate the starting temperature, the choice of Markov chain length for the inner loop, the function which controls the decrease in temperature and finally the termination criteria or the outer loop exit. SA is guaranteed to converge to a global optimum if one of two conditions are met. First, with a fast cooling rate, at each temperature an infinite transition has to be generated,. Second, with a finite length of Markov chain, the decrement rate of temperature T_k must not faster than O (1/log(k)) [1, 23]. In any other implementation, SA will not guarantee to converge to optimality. Therefore, the quality of the final solution will depend on the cooling schedule. In the next section we will discuss a number of different cooling schedules.

4. Cooling schedules

In this section we explore 5 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. 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) The starting temperature.

(ii) The cooling factor

(iii) The length of the Markov chain

(iv) The process termination condition.

The variants of cooling schedules which are considered here all share the same procedures in steps (i), (iii) and (iv). The starting temperature (step (i)) is defined by White [1984], who proposed that the starting 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 (step (iii)) for all the problems we tested our SA on, so that the dependency on the Markov chain length of the final solution quality can be easily observed. The process termination (step (iv)) is based on a number of repetition of cost at the end of number of consecutive Markov chains and the time window of which the latter is only applied to the last three reheating schemes.

4.1 Geometric cooling schedule

Geometric cooling is described in the formula below. The cooling factor (α) should be constant and less than one [7, 8, 9, 21, 22, 23].

$$T_{k+1} = \alpha \times T_k$$

This scheme is probably the most common schedule, and acts as a base line for comparison of other more elaborate schemes. Some typical examples of its use can be found in Johnson, et al [1986], Sechen and Sangiovanni-Vincentelli [1985]. It is clear that this simple cooling scheme does not take account of the state of the system in any way, and thus cannot adapt the intensity of the search depending on the difficulty of the problem.

4.2 Multiple cooling rates

The simple geometric cooling scheme always applies the same cooling rate regardless of the state of the system. By observing physical systems, it is known that a substance may undergo a number of phase transitions during cooling, before it reaches its frozen state [17, 18]. 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 as 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 solved. This process has been observed by Lister in the travelling salesman problem [25]. The geometric cooling scheme described in the previous section applies the same cooling factor regardless of the phase. However, at high temperatures almost all interchanges are accepted, many of them being non-productive moves. Applying a different cooling rate depending on the phase would allow the algorithm to spend less time in the high temperature phases and spend more time in the low temperature phases, thus reducing the total amount of time required to solve a problem given. By the same token, the high temperature phase cannot be omitted because it is responsible for setting up the super-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 equation (8) is maximal. Equations (4) through (7) are required, and are derived in van Larrhoven & Arrts [23].

Assume equilibrium state

a. The expected cost

$$< C(T) > = \sum_{i \in R} C(T) Pr_i(T) \qquad (4)$$

b. The expected square cost

$$< C^2(T) > = \sum_{i \in R} C^2(i) Pr_i(T) \qquad (5)$$

c. The variance

$$\sigma^2 = < [C(T) - < C(T) >]^2 > = < C^2(T) > - < C(T) >^2 \qquad (6)$$

d. The entropy

$$S(T) = - \sum_{i \in R} Pr_i(T) \ln Pr_i(T) \qquad (7)$$

e. The specific heat

$$\frac{\partial}{\partial T} < C(T) > = T \bullet \frac{\partial}{\partial T} S(T) = \frac{\sigma^2(T)}{T^2} \qquad (8)$$

Where $Pr_i(T)$ is given by (2)

The new cooling schedule is defined below. Here, the temperature is reduced geometrically, 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.

$$T_{k+1} = \alpha \times T_k \quad \text{if } T_k > T_{msp}$$
$$T_{k+1} = \beta \times T_k \quad \text{if } T_k \leq T_{msp}$$

where T_{msp} is the temperature at which the maximum specific heat occurs
α and β are constant and are less than 1
$\alpha < \beta$

In practice it is possible to compute the specific heat by calculating the standard deviation of cost for each markov chain at each temperature using a geometric cooling function with a fast cooling rate. When $\frac{\partial}{\partial T} < C(T) >$ is plotted against temperature, the maximal value can be determined, and used as the switching value.

4.3 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 sufficiently large to move to the global optimum. Thus, no matter how slow the cooling from that point, 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 to allow it to escape. In this section we describe a simple scheme for reheating. Mathematically, it is expressed as

case operator

cooling : $T_{k+1} = \alpha \times T_k$

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

where α and β are constant and are 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 started. Heating is terminated when the cost changes, either increasing or decreasing. 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 the next section we propose a variant of this stopping criteria which helps avoid these symptoms.

4.4 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 results in cycling of local minima. Therefore it is necessary to choose an optimal 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 algorithm summarises this process:

β = β_default
case operator
cooling : $T_{k+1} = \alpha \times T_k$
heating : begin
$T_{k+1} = T_k / \beta$
if Count (local optima) > Fixed_number then
$\beta = \beta - C$
end

where α is constant and are less than 1
β_default is large constant which is close to 1
C is a very small constant within the interval (0,1)

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 algorithm summarises the termination condition:

if $(\Delta C_k > 0)$ **and**
$[(\Delta C_{k+1} = -\Delta C_k)$ **or** $(\Delta C_{k+1} = -2 \times (\Delta C_k)]$
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 kth 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.

4.5 Reheating as a function of cost

The previous two schemes use a geometric function for both heating and cooling. Heating is terminated when the system begins to allow some cost increases again, in the hope that this will allow it to escape the local minima. However, time is wasted in heating up to the required level. In this section we describe a method of reheating which directly sets the temperature to a value which allows the system to escape the local minima, without the need to approach the temperature slowly. Consequently, 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 increased the system is reheated less.

In section 4.2 we stated that the high temperature phase cannot be omitted due to its responsibility in creating the super-structure of the solution. The phase transition is the point at which the system begins to solve the sub problems and the super structure 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 that this value, then the system cannot escape from local minima which are caused by poor sub-structure. 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 (i.e. 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 higher than the phase transition temperature. In general, the higher the best cost, the higher the temperature which is required 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. We have experimented with a simple linear relationship, as shown below, however, other more complex forms are possible:

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

Where T_{new} is the reset temperature
T_{msp} is the temperature at which the maximum specific heat occurs
P is a constant
$C(x^b)$ is the current best cost.

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 4.4.

5. The Timetabling Problem

We trialed the new cooling schedules on a the school timetabling problem. This problem is NP-Hard, and for highly constrained data sets it can be very difficult to solve. Since all of the cooling schedules we have used are bounded polynomially in the problem size (we used a constant markov chain in our experiments) it is possible to observe the effect of the cooling schedule on the quality of the solution.

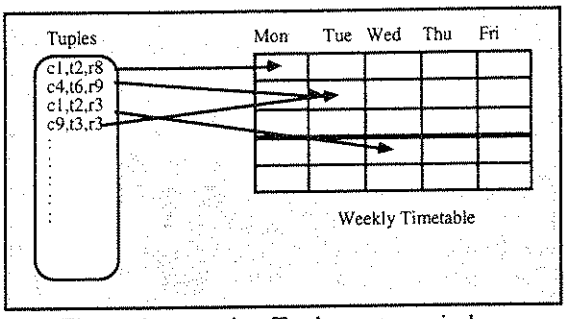

Figure 1 - mapping Tuples onto periods

Timetable scheduling is the problem of allocating various combinations of class, teacher, subject and room onto periods [4, 5, 6] as shown in Figure 1. Let a tuple be a particular combination of identifiers such as class, teacher and room. 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 common identifiers in common). The SA algorithm [2, 3, 10, 11, 12, 16, 17, 18, 19] operates by moving tuples out of one period and inserting them into another. The change in cost caused by this movements is computed. A decrease in cost is accepted, and an increase is only accepted if it satisfies the Boltzmann equation. The cost of the system is computed by counting the number of clashes in each period. This cost, which is clearly non-linear, increases only once the number of occurrences of an identifier in a period is greater than one.

It is possible to generate a number of test problems which are totally constrained using a random generation program. Tuples are built by choosing random class, teacher and room identifiers, and are then packed into a period which can accommodate them. If there is no 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 totally constrained. The tuples are then deallocated from the period slots and are reallocated using the simulated annealing algorithm. In this paper we generated a number of test data sets containing from 150 to 390 tuples each with 30 periods. 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 trailed in another study [6].

6. Results

In this section we summarise the results of comparing the 5 different cooling schemes on the timetabling problems described in the previous section. Table 1 shows the characteristics of each problem together with the best solution obtained by the standard geometric cooling schedule. It can be observed that the larger problems have not been solved even though zero cost solutions are known to exist.

Tables 2 compares the results of the 5 schemes. It shows the average solution obtained by the 5 schemes across 6 runs, and the average time taken to reach that solution. We have reported the best average achieved for each scheme regardless of the time taken. These results allow some rough 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 both solved more problems than either of the geometric cooling schemes, and where they didn't solve the problem they produced lower cost solutions. The reheating scheme based on cost solved even more problems, although it took quite a long time for some of them. The only problem it failed to solve was the largest.

In Tables 3 and 4 we show a more useful comparison of the 5 schemes. In Table 2 the time is reported for each schedule for roughly the same solution quality. From these results it can be seen that the reheating schemes all produce an equivalent solution to the simple geometric cooling scheme in a shorter time. Also, the multiple cooling rate offers a faster solution than the single cooling rate, and in some cases, is even faster than the reheating schemes.

In Table 4 the solutions are compared given the same processing time. It can be seen that the multiple cooling rate always produces a lower cost solution than the single geometric cooling scheme. The reheating which is based on cost solves more problems than any other scheme and always produces a lower cost in the cases that it doesn't solve.

Test	Number of Tuples	Number of Classes	Number of Teachers	Number of Rooms	Cooling Rate	Average Final Cost	Best Cost	Average Times (Secs)
1	150	5	5	5	0.8	7.5	5	3.42
					0.9	6.83	5	3.83
					0.99	2.5	0	10.23
					0.999	0.67	0	72.45
					0.9999	0.67	0	731.43
2	180	6	6	6	0.8	11.33	6	3.68
					0.9	9.0	5	4.08
					0.99	5.5	3	8.67
					0.999	2.5	0	80.22
3	210	7	7	7	0.8	12.17	8	3.92
					0.9	11.83	7	4.08
					0.99	5.67	4	10.45
					0.999	3.83	2	107.05
					0.9999	2.5	2	6635
4	240	8	8	8	0.8	14.5	10	4.23
					0.9	13.83	9	4.77
					0.99	9.33	7	16.85
					0.999	5.5	4	125.83
					0.9999	3.83	2	7859
5	270	9	9	9	0.8	16.17	13	3.97
					0.9	16.0	14	5.02
					0.99	9.67	8	13.63
					0.999	6.17	5	122.45
					0.9999	5.17	4	8106
					0.99995	4.5	2	19167
6	300	10	10	10	0.8	19.5	17	4.12
					0.9	20.0	17	5.28
					0.99	12.8	12	22.17
					0.999	8.17	6	154.42
					0.9999	7.17	7	1647.60
7	330	11	11	11	0.8	23.67	18	4.22
					0.9	19.8	17	4.98
					0.99	15.33	13	19.08
					0.999	11.17	10	153.58
					0.9999	7.17	6	1922.90
8	360	12	12	12	0.8	24.17	21	5.17
					0.9	24.67	21	6.30
					0.99	18.83	15	21.78
					0.999	10.67	9	181.00
					0.9999	9.5	7	2075.97
9	390	13	13	13	0.8	25.83	21	4.78
					0.9	27.5	26	6.03
					0.99	21.5	20	24.00
					0.999	15.5	13	189.58
					0.9999	11.0	9	2320.07

Table 1 - geometric cooling schedule results

	Geometric Cooling		Multiple Cooling Rate		Geometric Reheating		Enhanced 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)
1	0.67	72.45	0	97.2	0	35.2	0	68	0	9
2	2.5	80.22	1.8	52	0	151.5	0	141	0	32
3	2.5	6635	2.17	46.9	0	205	0	250	0	63
4	3.8	7859	3.33	1043	0.33	423	1.6	782	0	112
5	4.5	19167	4.5	111	1	580	1.8	806	0	271
6	7.17	1647	5.67	2046	2.17	1603	2.3	1522	0	2398
7	7.17	1922	6.83	1191	3.8	1611	4.8	1601	0	7555
8	9.5	2075	8.8	1342	4	1604	5.17	1602	0	18375
9	11	2320	8.67	279	7.3	1604	6.6	1603	2.8	21224

Table 2 - comparison of 5 schemes, time and cost varying

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

Table 3 - Comparison of five schemes - Time for same quality

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

Table 4 - Comparison of five schemes - Quality for same time

7. Conclusions

In this paper we have compared 5 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 three 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. We have experimented with the reheating schemes on another problem (set

partitioning) and found it quite effective, and these combined results will be reported at a later date.

Acknowledgments

The work described in this paper was performed at part of a joint project between Griffith University and the Royal Melbourne Institute of Technology. Henry Dang is 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.

References

[1] Aarts, E and Korst, J., (1989), **Simulated Annealing and Boltzmann Machines**, John Wiley, Chichester.

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

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

[4] Abramson, D.A., **Constructing School Timetables using Simulated Annealing: Sequential and Parallel Algorithms**", Management Science, Jan 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**, Lecture Notes in Economics and Mathematical Systems, Rene V.V. Vidal (Ed.), Applied Simulated Annealing, (1993), 103-124.

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

[8] Bonomi, E. and Lutton, J.L., **The Asymptotic Behaviour of Quadratic Sum Assignment Problems: A Statistical Mechanics Approach**, Euro. J. of Oper. Res., 1984b.

[9] 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, pp177-1282.

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

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

[12] Chyan, D.J. and Breuer, M.A., **A Placement Algorithm for Array Processors**, Proceedings 20th Design Automation Conference, Miami Beach, June1983, pp 182-188.

[13] Ingber, L., **Very Fast Simulated Re-Annealing**, Mathl. Comput. Modelling, 1991, 15, 77-78.

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

[15] 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.

[16] Kang, Le, Belanger, P., White, G. and Con Schoenberg, G., **Computerized Scheduling**, Proceedings of the 36th Annual College and University Users Conference, Louisville, Kentucky, April 28th - May 1st, 1991.

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

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

[19] Leong, H.W. and Liu, C.L., **Permutation Channel Routing**, Proc. IEEE Int., Conference on Computer Design, Port Chester, 1985, pp 579-584.

[20] Lundy, M. and Mees, A., **Convergence of An Annealing Algorithm**, Math. Prog., 1986, 34, 111-124.

[21] Morgenster, C.A. and Shapiro, H.D., **Chromatic Number Approximation Using Simulated Annealing** , Department of Computer Science, The University of Mexico, Albuquer, Technical Report No. CS86-1, 1986.

[22] Sechen, C. and Sangiovanni-Vincentelli, A.L., **The Timber Wolf Placement and**

Routing Package", IEEE J. Solid State Circuits, 1985, SC-20, pp 510-522.

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

[24] White, S.R., **Concepts of Scale in Simulated Annealing**, Proc. IEEE Int. , Conference on Computer Design, Port Chester, 1984, pp 646-651.

[25] Lister, R. **Annealing Networks and Fractal Landscapes**, Proc. IEEE Int. Conf. on Neural Networks, San Francisor, March 1993, Vol I, pp 257 - 262.