Staff Workload Scheduling in Large Engineering Schools

M. L. Ng, H. B. Gooi, and C. Lu

Abstract—**This paper uses the simulated annealing algorithm to solve a unique staff workload scheduling problem present in a school that has about 250 academic staff and 4000 students. This paper also examines and compares various types of cooling schedules. Five types of cooling schedules, namely, geometric ratio, hybrid geometric function with reheating, quadratic equation, Huang's equation and improved quadratic equation are tested for the scheduling problem. For each cooling schedule, ten test runs are conducted. The results show that the hybrid geometric function with reheating, whose value is directly proportional to the cost function, yields best results to the problem.**

Index Terms—**Simulated annealing, cooling schedules, scheduling, staff workload.**

I. Introduction

The process of scheduling has become an important school function. Within a school, there are different types of scheduling that can be found for various purposes. For example, there is scheduling of curriculum timetables that show all subjects the school is offering. It is not only a time scheduling, but also a resource scheduling whereby the location and time of the teaching slots are reflected. In a school with various divisions, each division has a division timetable that shows only the teaching slots that the division is undertaking. All division timetables are a subset of the school master timetables. Once teaching slots are known, lecturers will be scheduled for teaching duties. When a particular subject involves two or more divisions, extra coordination will have to take place. Besides teaching scheduling, there would also be examination scheduling which normally takes place at the end of a semester. Examination scheduling, like curriculum scheduling, is also a time and resource scheduling process that involves coordination and careful planning. All these scheduling tasks would have to be properly planned before a school can function in a timely manner.

In the School of Electrical and Electronics Engineering (EEE) of Nanyang Technological University (NTU), the process of scheduling used to be done manually. It involved weeks of planning and corrections before one master timetable could be used. The planner had to balance among manpower, time and availability of physical resources such as lecture theatres, tutorial rooms and laboratories. The types of scheduling that the School of EEE did were 1) curriculum scheduling, 2) staff workload scheduling, 3) student timetable scheduling and 4) examination scheduling.

Any of these scheduling processes could easily take half a month if they were to be done manually. It did not take long before the error-prone manual scheduling became intolerable. This is also a consequence when more students were matriculated. Thus, automated scheduling is not only desirable but also a must in solving the increasingly difficult timetabling problem.

With the advancement of information technology, commercial software is readily available to solve many scheduling problems. Commercial software makes use of state-of-the-art algorithms such as genetic algorithms and simulated annealing. While commercial software packages are able to solve general scheduling problems, they are unable to adapt to unique curriculum changes and IT innovations in the future. To constantly engage software suppliers to upgrade the software package due to curriculum changes and new IT innovations would be too costly for the school to bear. Thus, there is a need to develop an automated scheduling system that meets EEE timetabling requirements so that it can be upgraded internally as and when the need arises.

This paper examines how the simulated annealing algorithm is applied to automate the scheduling of EEE staff workload. The focus is on various cooling schedules.

II. Staff Workload Scheduling

There are six divisions and one institute, Information Communications Institute of Singapore (ICIS) in EEE. A Head of Division (HOD) or director is overall responsible for staff workload assignment in each division/institute. Staff workload scheduling is the planning of the teaching timetable for each lecturer in the division/institute. Its objective is to assign the teaching hours to each staff member according to the average teaching hours as agreed by all HODs and the ICIS director in the meeting chaired by the Dean of EEE. The HOD or director may increase/decrease the staff teaching hours after considering the staff's research and/or administrative duties. While the variation in teaching hours can occur among staff, the number of teaching hours assigned to the division/institute should remain the same as agreed. The teaching schedules generated must be implementable. They must minimize unnecessary inconveniences to the staff member concerned and at the same time must ensure that all teaching hours within the division/institute are assigned.

Staff workload scheduling takes place after the curriculum timetable has been finalized. It is the process whereby the teaching staff members are assigned to

Manuscript received April 4, 2003; revised December 4, 2003. This work was supported in part by the NTU research scholarship, which enabled Mr. Ng to pursue his M.Eng. degree at NTU.

M. L. Ng is with ST Training & Simulation Pte. Ltd., Singapore (e-mail: ngml@pacific.net.sg).

H. B. Gooi and C. Lu are with the School of Electrical & Electronic Engineering, Nanyang Techological University, Singapore 639798 (e-Email: ehbgooi@ntu.edu.sg).

Publisher Item Identifier S 1682-0053(04)0208

curriculum time slots that the division/institute is responsible for. The HOD or director assigns the crucial teaching loads, such as lecture and Final Year Design. The timetable coordinator then assigns the remaining loads. Teaching loads assigned by HOD become hard constraints since they cannot be changed by the timetable coordinator.

There are five types of classes, namely lecture, tutorial, project, design and laboratory that can be assigned to staff. The duration of each semester is thirteen weeks. Lecture is conducted on the basis of one hour per session. For each subject, there could be one to three lecture sessions per week to be conducted and its workload may be distributed among two to four lecturers. Tutorial is also conducted on the basis of one hour per session per week for a total of 12 weeks starting from the second week of the semester. There could be one or more tutors involved to teach the 12-week tutorials. Project and Design are conducted on the basis of three hours per session per week. To complete a project module, it will take 3 weeks. Design modules generally take 2-3 weeks to complete. Each student will complete four modules spread over 12 weeks. Laboratory is conducted on the basis of 3 hours per session per week. Each laboratory session takes only one week to complete. Each student will complete 12 weeks of laboratory.

There are many constraints that the timetable coordinator needs to overcome. They can be generally classified into hard, medium and soft constraints. Hard constraints must be respected. The generated schedule cannot be executed if they are violated. Medium constraints, if violated, will cause discomfort to staff. Soft constraints, if violated, will cause the least discomfort to staff. Each of these constraints carries a cost value if the constraint is violated each time. Thus the total cost would be given by

$$C_T = aC_H + bC_M + cC_S \qquad (1)$$

where C_H is the cost of a hard constraint violation; C_M is the cost of a medium constraint violation; C_S is the cost of soft constraint violation; a, b and c are the number of hard, medium and soft constraint violations respectively. The above equation is also known as the *cost function*. In practice, $C_H >> C_M >> C_S$. In the actual implementation, $C_S = 1$; $C_M = 100$; $C_H = 1000$ to ensure that there is a wide cost variation among each type of constraints. One advantage for such assignment is that the status of the schedule could be identified easily. For example, if the cost is equal to 4578, then, it is likely that the schedule consists of a total of 4 hard constraint violations, 5 medium constraint violations and 78 soft constraint violations, assuming that the total number of medium and soft constraint violations does not reach 10 and 100 respectively.

Thus the objective of the timetable coordinator is to schedule the staff workload so that its C_T is as low as possible. Examples of the three types of constraints are:

1. *Hard Constraints*
   - A staff member cannot be scheduled to appear at two different locations at the same timeslot.
   - A staff member cannot be scheduled to teach in a timeslot if it is within a no-workload period.
   - A staff member cannot be scheduled to teach a subject that he/she did not opt for.
2. *Medium Constraints*
   - The number of consecutive hours assigned to staff should not exceed a pre-specified maximum hours.
   - If the request to teach a particular subject was approved by HOD/director, the staff member should be assigned a certain minimum number of subject groupings so that the number of preparations is kept to a minimum.
   - Certain laboratory modules should be taught by the same staff assigned to teach the associated class.
3. *Soft Constraints*
   - The total number of hours scheduled to a staff member should not be different from what was requested.
   - Staff members giving lectures should be given at least one-hour free time prior to the lecture.
   - The total teaching hours of a staff member in any day should not be more than a pre-specified daily limit.

The generated schedule must meet at least the hard constraints specified before it can be classified as a feasible schedule. Like any other timetabling problems, staff workload scheduling is also categorized as a non-polynomial (NP) problem. Non-polynomial means that the problem cannot be formulated by using any polynomial function, and thus the solution to the problem cannot be verified in polynomial time. The practicality of the final schedule generated will depend on how well the search algorithm is designed and how the optimization algorithm looks for a solution which leads to a minimum C_T.

III. Optimization Algorithms

Many forms of algorithms have been formulated in the past few decades in an attempt to solve NP problems. Some are powerful but too complicated to be implemented on small problems, while others are easy to implement but cannot produce satisfactory results on larger problems. The choice of algorithms depends on numerous factors related to the nature of the problem. Some factors considered are 1) population size of the problem, 2) number of constraints imposed, 3) time of execution to produce a feasible scheduling result and 4) time of implementation.

The population size in the staff workload scheduling problem refers to the number of teaching slots to be scheduled. Typically for a division/institute, there are about 3500-4000 teaching slots to be scheduled. Each teaching slot is one to three hours long. The number of constraints would be directly proportional to the number of staff members involved. The time to produce a feasible schedule varies proportionally to the number of teaching slots. In addition, it would depend on the algorithm chosen.

There are many readily available algorithms to solve NP problems. The more popular ones are genetic algorithms, simulated annealing, graph coloring and constraint logic programming. Each of these has been implemented to solve timetabling problems by many researchers. Rich [1] has designed a software solution based on genetic algorithms for creating a university class timetable. Elmohamed *et al*. [2] have tackled the NP problem of academic class scheduling at the university level based on simulated annealing. Sheung *et al*. [3] used both genetic algorithms

and simulated annealing to solve a prototypical timetabling problem in a tertiary institute.

Performance comparison of these algorithms was done in various types of problems. Hasan *et al*. [4] compared three algorithms, simulated annealing, genetic algorithms and tabu search for the unconstrained Quadratic Pseudo-Boolean (QP) function. The difference between the average final solutions is not significant for all the three algorithms. Youssef *et al*. [5] compared simulated annealing, tabu search and genetic algorithm on the floor planning problem. Of the three algorithms, tabu search produces the best results with respect to the quality of the solution. With respect to the complexity of implementation and tuning of parameters, genetic algorithms require the most effort. Sheung's [3] results reflect that simulated annealing converges faster and its final cost is lower than that of genetic algorithms. This is more so when the complexity of the algorithm grows.

Ng [6] attempted to solve the traveling salesman problem (TSP) using Hopfield nets, simulated annealing, Boltzmann machines and genetic algorithms. Simulations were run for many cities. As the number of cities increases, simulated annealing has consistently produced optimal or near-optimal solutions and is faster compared to the other algorithms.

No single algorithm behaves the best in all types of problem domains, but one algorithm could be more suitable for a specific problem. Also, the success of the algorithm largely depends on the understanding of the programmer of both the algorithm and the problem specifics. Simulated annealing is chosen to solve the staff workload scheduling problem due to its simplicity and ease of implementation.

The most important criterion in choosing an algorithm is to determine whether the algorithm would converge to produce a feasible solution. It can be shown that simulated annealing will converge under appropriate conditions. Such conditions were derived by Laarhoven *et al*. [7]. These conditions will be examined later. Simulated annealing is robust, which makes it easier to solve general and specific problems. Many researchers have implemented simulated annealing not only to solve scheduling problems, but also in VLSI PCB routing [8] and automatic generation of one-line diagrams [9].

Simulated annealing is derived from thermodynamics, specifically imitating the way that metals cool and anneal. This is a technique patterned after the physical process of annealing of a solid, in the branch of engineering, known as metallurgy. A minimum cost function corresponds to the minimum energy (ground state) of a substance. The simulated annealing process lowers the temperature slowly until the system freezes and no further changes occur. At each temperature step, the length of the Markov chain must be long enough for the system to reach an equilibrium. The essence of the process is slow cooling, allowing ample time for redistribution of the atoms as they lose mobility. It is essential to ensure that a low energy state will be achieved.

IV. Implementing Simulated Annealing

An overview of the algorithm is shown in Fig. 1. The algorithm begins by generating an initial schedule and the cost of the schedule is also evaluated by the cost function in (1). The costing shows the number of violated constraints. Thus, the lower the cost, the better the schedule is. An initial temperature is subsequently selected. The iterative process starts by constructing new schedules and imposing a random displacement from the solution space. If the cost of the new schedule generated is smaller than the cost of the current schedule, then the new schedule is accepted as a replacement to the current schedule. If the cost is worsened by a ΔC amount, then the schedule will only be accepted by a probability given by (2),

$$r < \exp(-\Delta C / T) \tag{2}$$

where r is a randomly generated number between 0 to 1 and T is the temperature that changes as the process progresses. This equation is also known as Metropolis criterion. The manner on how the temperature varies is known as the cooling schedule.

The iterative process will halt when the stopping criterion is satisfied. In this case, the program stops when it has reached 1000 iterations or when the cost is below 50. The value of 50 is chosen, as it is reasonably small. Based on the cost of the various types of constraints defined in Section II, the chosen value ensures that the final converged schedule may contain a few soft constraint violations of lesser importance while keeping free of any hard and medium constraint violations.

To achieve good quality schedules, the cooling schedule is of critical importance. The temperature during the scheduling process will affect whether the solution is trapped in a local minimum or not. In the following section, five different types of cooling schedules are examined.

V. Cooling Schedules

The result of simulated annealing is largely dependent on the selection of the cooling schedule. There have been some efforts to derive a universal set of rules for defining the cooling schedule but there is limited achievement. In this section, some popular cooling schedules such as geometric cooling schedule, Huang's *et al*. [10] equation and quadratic equation are discussed. For each cooling schedule, ten test runs are conducted. All test runs use the same initial schedule.

The results of each cooling schedule are shown in Tables I and II. Table I shows the number of iterations before all hard constraints are resolved. When all the hard constraints are resolved, the schedule is feasible and can be implemented. Table II shows the final cost after 1000 iterations. The quality of the solution results is reflected in the final cost. The higher the final cost, the poorer the quality of the solution is.

A. Geometric Cooling Schedule

The geometric cooling schedule is probably the most well-known and simplest in the simulated annealing arena. The geometric cooling schedule is represented by,

$$T_{i+1} = \alpha T_i \tag{3}$$

where α is the geometric ratio and T is the temperature. It is easy to understand and can be implemented without much effort. The value of α is determined experimentally via trial and error. It is usually in the range of 0.6 ~ 0.99. The value of T will change every M iterations. In the

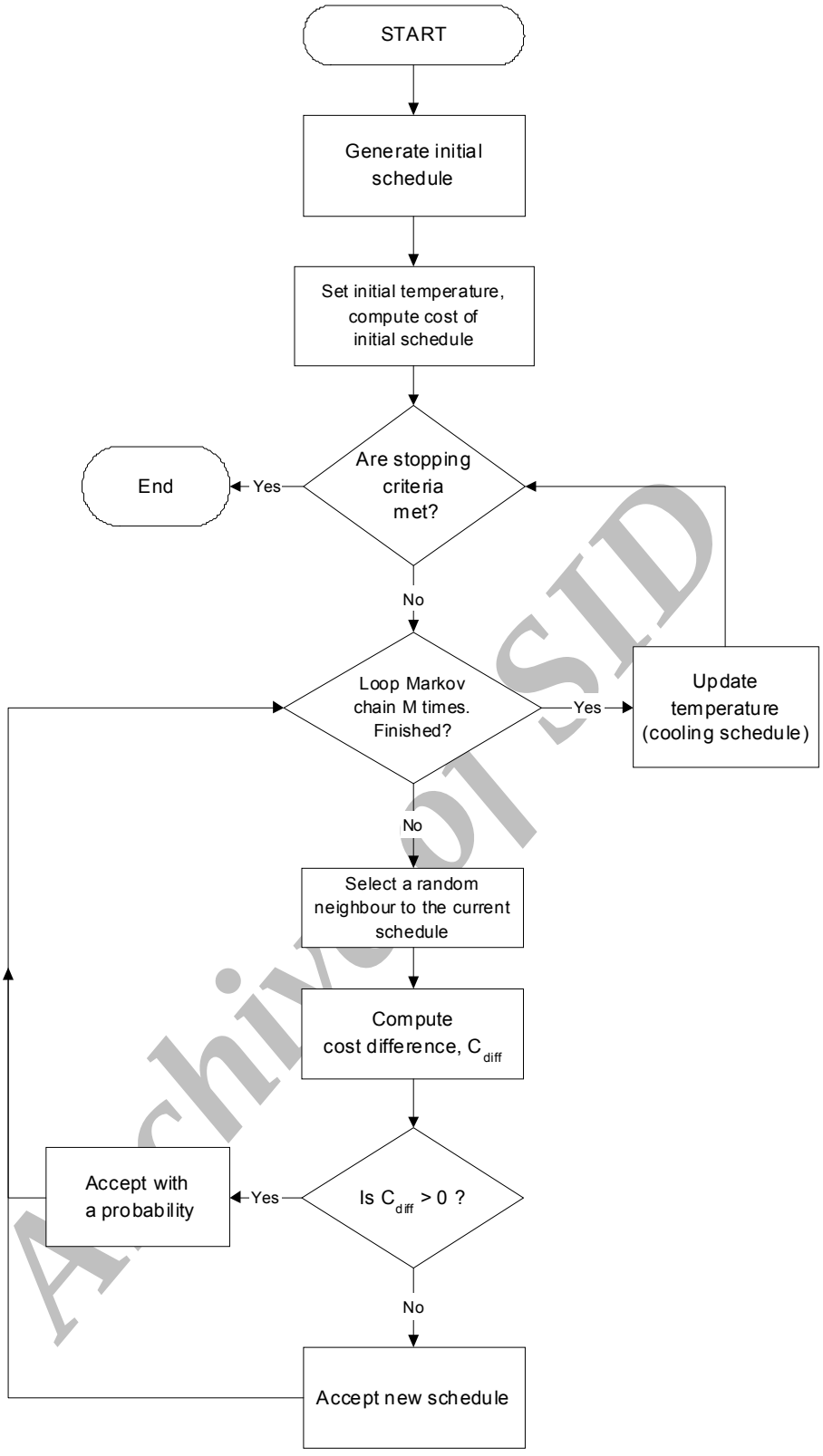

Fig. 1. Flowchart of simulated annealing algorithm.

test, M is chosen to be 10. The geometric cooling schedule, though simple, is often not able to get satisfactory results.

For the results of $\alpha = 0.8$, it can be seen from Tables I and II that no test run is able to resolve all the hard constraints completely. The main reason behind this that the temperature decreases too fast as shown in Fig. 2 and the schedule is often trapped in a local minimum.

For the results of $\alpha = 0.95$ from Tables I and II, some improvement is seen in the results over $\alpha = 0.8$. 40% of the test runs were able to resolve all the hard constraints successfully. However, the final cost of these test runs is still high. The temperature curve is shown in Fig. 3.

Based on the results of these two tests, it can be observed that the higher the geometric ratio, the better the results are. This is because the higher the geometric ratio, the slower the temperature decrements, thereby, giving enough time for the schedule to converge to a global minimum.

TABLE I
NO OF ITERATIONS TO ACHIEVE A FEASIBLE SCHEDULE ($C_H = 0$)

Trial Numbers	Geometric $\alpha = 0.80$	Geometric $\alpha = 0.95$	Hybrid Geometric Function with Reheating	Huang's Equation	Quadratic Equation	Improved Quadratic Equation
Test Run 1	****	****	****	****	****	****
Test Run 2	****	****	634	605	****	832
Test Run 3	****	319	395	518	****	****
Test Run 4	****	282	548	****	****	336
Test Run 5	****	****	823	****	908	274
Test Run 6	****	145	412	****	****	****
Test Run 7	****	302	658	****	****	266
Test Run 8	****	****	314	543	****	326
Test Run 9	****	****	62	689	****	386
Test Run 10	****	****	719	834	837	****

**** means $C_H \neq 0$ at iteration = 1000.

Test Run x, where $x = 1,2 \ldots$ or 10 in Tables I and II, refers to the same run.

TABLE II
FINAL COST AT 1000TH ITERATION

Trial Numbers	Geometric $\alpha = 0.80$	Geometric $\alpha = 0.95$	Hybrid Geometric Function with Reheating	Huang's Equation	Quadratic Equation	Improved Quadratic Equation
Test Run 1	26254	2082	2284	7218	3802	1586
Test Run 2	7320	17330	132	230	4766	36
Test Run 3	49132	114	106	420	1756	1322
Test Run 4	30126	112	14	18480	9742	218
Test Run 5	3112	2254	48	4030	526	2
Test Run 6	35036	206	24	4020	380	2366
Test Run 7	11410	230	232	12162	13174	212
Test Run 8	18398	3102	208	424	4906	206
Test Run 9	29356	5182	42	254	4526	12
Test Run 10	37068	2340	218	265	700	2126

Test Run x, where $x = 1,2 \ldots$ or 10 in Tables I and II, refers to the same run.

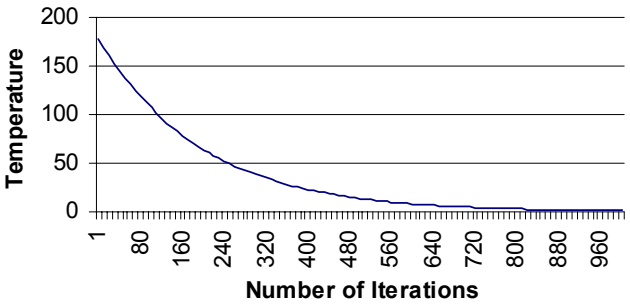

Fig. 2. Temperature curve for $\alpha = 0.8$.

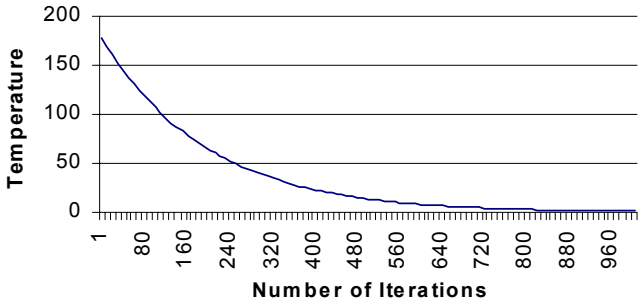

Fig. 3. Temperature curve for $\alpha = 0.95$.

B. Hybrid Geometric Function with Reheating

In geometric cooling schedules, the temperature is decremented without considering the cost values. A more adaptive cooling schedule would be a function of the cost value and the gradient of the cost function. Tying the temperature to the cost ensures that the cooling schedule can be dynamically affected by the cost curve behavior.

The lower the cost value means that fewer constraints are violated and hence the temperature should be kept low. Likewise, the higher the cost value means that more constraints are violated and hence reheating may be needed so that the solution process may chart a new and better convergence path. The implementation of the hybrid function is:

If (Hard Constraints still exists) *Then*
  If Gradient < 0 *Then*
    $T_{i+1} = \sqrt{C_i}$ (*Temperature equal to square root of average cost*)
  Else
    $T_{i+1} = 1.3\sqrt{C_i}$ (*Reheating occurs*)
  End If
Else
  $T_{i+1} = 0.9T_i$ (*Geometric cooling schedule*)
End If

It can be seen that this cooling schedule consists of two main parts. The first part is when the schedule consists of hard constraints, while the second is when the schedule has resolved all the hard constraints. In the first part, when the gradient of the cost curve < 0, this implies that the cost is reducing. When the gradient ≥ 0, this means that there is no change of the cost or the cost is increasing in that Markov chain. When this happens, it is possible that the schedule is trapped in a local minimum, thus reheating occurs. Reheating enables the solution to jump out of the local minimum. When the hard constraints are resolved, the schedule is feasible. Using a geometric cooling schedule ensures that the cost will converge slowly. The temperature change is at every M iterations, similar to geometric cooling scheduling.

The results of the hybrid geometric function schedule from Tables I and II are much better than those of the geometric cooling schedule. 90% of the test runs are able to generate feasible solutions. The main advantage of this cooling schedule is that the rate of change of the temperature is directly proportional to the rate of change of the schedule cost. Thus, the temperature will never decrease too quickly or too slowly. In general, the geometric cooling curve with reheating decays exponentially and is similar to the geometric cooling curve in Fig. 3. The cooling is adaptive to changes to schedule cost.

It can be observed that although the cooling schedule realized 90% of generated schedules, the number of iterations required to achieve $C_H = 0$ does not follow any particular trend. This is because the solution search is performed randomly.

C. Huang's Equation

A more complex cooling schedule explored by Huang *et al*. [10] makes use of the standard deviation of the cost of previous Markov chains. In this type of cooling schedule, the temperature is directly dependent on the results of the scheduling. The equation is:

$$T_{i+1} = T_i \exp(-\frac{\lambda T_i}{\sigma}) \qquad (4)$$

where λ is a constant less than 1 and σ is the standard deviation of the cost of the previous Markov chain. To avoid a drastic decrement of the cooling schedule, the ratio of T_{i+1}/T_i is restricted to 0.9. This means that the temperature drop between two successive iterations cannot be too high and it does not imply that Huang's equation is following the geometric cooling curve. Instead, Huang's cooling curve is determined by (4). In the test runs, $\lambda = 0.1$. Test is conducted at every M iterations to determine if the temperature has reached equilibrium before the temperature is updated.

Out of the ten test runs in Tables I and II, five runs produce feasible results. The cost variation of the initial Markov chains is small, thus, the standard deviation, σ, is small too. Although there is a mechanism to restrict the drastic drop of T_{i+1}/T_i, the temperature decreases very fast due to the exponential decay nature of the cooling schedule. Fig. 4 shows the temperature curve for a typical test run. It can be seen that the temperature drops very fast within the first two hundred iterations. This will easily cause the solution to be locked at a local minimum.

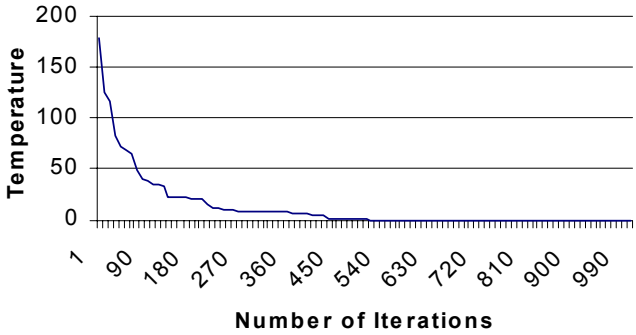

Fig. 4. Temperature curve for Huang's equation.

D. Quadratic Equation

Anderson *et al*. [11] suggested a quadratic equation cooling schedule for use in a class of network design problems in telecommunications. This cooling schedule is a function of the starting temperature (T_0), ending temperature (T_F) and the iteration number i as shown in (5).

$$T_{i+1} = ai^2 + bi + c \qquad (5)$$

where

$$a = (T_0 T_F)/(I_{\max})^2$$

$$b = 2(T_F - T_0)/(I_{\max})$$

$$c = T_0$$

in which $I_{\max}$ is the maximum number of iterations and could be chosen large so that more iterations can be covered. T_F is usually chosen such that $T_0 >> T_F$. Thus,

$$(T_F - T_0) \approx -T_0 .$$

If $I_{\max}$ is chosen such that

$$I_{\max} >> T_0 ,$$

then

$$-T_0 / I_{\max} << T_0 .$$

It can be easily observed that the second term in (5) is the only negative component since $2(T_F - T_0)/(I_{\max})$ must be less than 0. In our experiments, $I_{\max} = 1000$, $T_0 = 178$ and $T_F = 0.1$. The rate of decrement of the cooling schedule largely depends on $(T_F - T_0)$ and $I_{\max}$ because $b >> a$. As a result, the rate of cooling is slow as it is affected mostly by the linear coefficient, b, rather than the quadratic coefficient, a. This characteristic is confirmed in Fig. 5. Since the decrement of the rate of temperature is too slow, the algorithm is not able to converge to an optimal solution effectively as can be seen from Tables I and II.

E. Improved Quadratic Function

To improve the rate of falling of the cooling schedule, the variable b of (5) is modified to

$$b = \varphi(T_F - T_0)/(I_{\max}) \qquad (6)$$

where φ is a constant that is to be determined experimentally. The value of φ will control the rate of the temperature decrement. In the test runs, $\varphi = 1.7$. The temperature curve is shown in Fig. 6. The rate of temperature reduction has been adjusted so that the final solution is able to converge in 6 out of 10 test runs shown in Tables I and II.

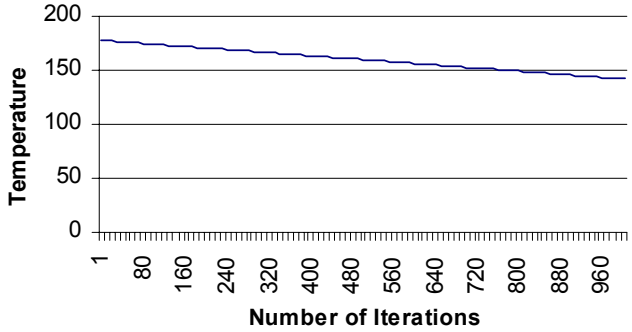

Fig. 5. Temperature curves for quadratic equation.

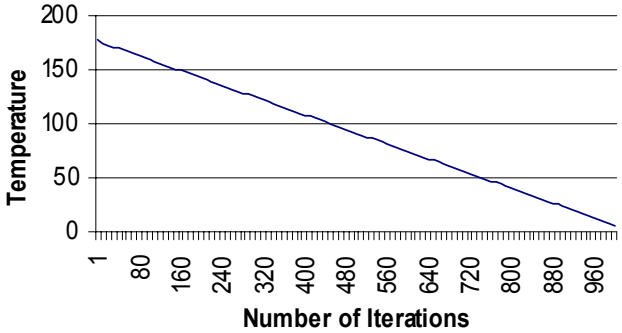

Fig. 6. Temperature curves for improved quadratic equation.

TABLE III
SUMMARY OF COOLING SCHEDULES

Cooling Schedule	% Schedules that can be implemented	Highest Cost of feasible schedule	Lowest Cost of feasible schedule	Average Cost of feasible schedule	Average No. of Iterations to achieve feasible schedule
Geometric $\alpha = 0.80$	0%	****	****	****	****
Geometric $\alpha = 0.95$	40%	230	112	166	262
Hybrid Geometric Function with Reheating	90%	232	14	114	508
Huang's Equation	50%	424	230	318	638
Quadratic Equation	20%	700	526	613	873
Improved Quadratic Equation	60%	218	2	115	404

VI. SOLUTION RESULTS

A. Comparison of Cooling Schedules

Table III summarizes the results of the cooling schedules discussed above. Among the types of cooling schedules, the hybrid geometric function with reheating cooling schedule produces the highest number of feasible schedules. It also has the best average cost of the feasible schedule. Although the average number of iterations is about two times that of geometric cooling, it is not a concern as this can be improved significantly if a more powerful computer is used.

B. Quality of Results from Simulated Annealing

The staff workload timetable generated from the simulated annealing algorithm is compared with the user specification. Due to the complexity of the problem, a zero cost may not be obtained. Table IV tabulates the assigned hours against the intended hours of all staff members in the division under test. It can be seen that 23 out of the 29 staff members were assigned the exact requested hours. The rest of the staff members were within ±3 hours from their intended hours. Thus, violation of soft constraints has been kept to a minimum. The total time taken to generate the schedule was 4 hours and 16 minutes.

Fig. 7 shows a sample workload schedule of a staff member (THN) whose allocated hours is the highest under test. On Thursday of week 3, THN is assigned a total of 6 hours. It can be seen that the software has avoided assigning THN more than 4 hours of continuous teaching. It is also verified that all the subjects allocated to the staff members has no hard constraint and medium constraint violations.

From the above discussion, it is clear that a workload scheduling solution is considered acceptable it must not only satisfy the intended hours for each individual staff member but also provide a clash-free timetable for each staff member. It is not always possible to make all assigned hours exactly equal to all intended hours for all staff because certain laboratory hours must be assigned as a multiple of three. Also, the total assigned teaching hours in

TABLE IV
BREAKDOWN RESULTS OF TEACHING ASSIGNMENT

Staffs' Initials	Total Assigned Hours	Intended Hours	Difference
CCH	198	198	0
CHC	200	200	0
CPK	183	183	0
DMA	63	63	0
EC	219	219	0
EW	222	222	0
GBH	204	201	3
GWL/1	206	205	1
HDJ	225	222	3
JC	189	190	-1
JCC	174	174	0
LCH	36	36	0
LKT	195	195	0
LMH	219	222	-3
MJG	203	203	0
NLS	192	195	-3
OTH	162	162	0
RG	102	102	0
SKY	212	212	0
SL	172	172	0
TCM	36	36	0
THK	213	213	0
THN	223	223	0
TYC	222	222	0
VO	210	210	0
YKS	170	170	0
YL	205	205	0
ZYP	214	214	0

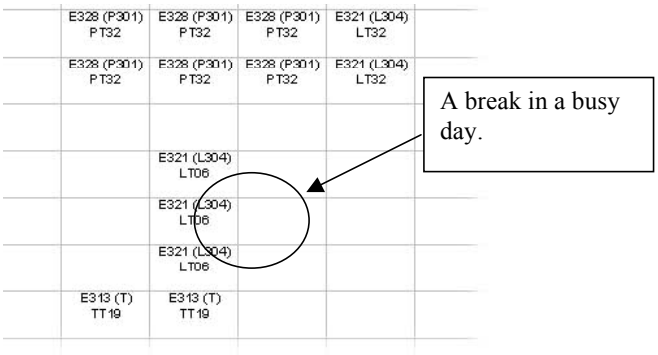

Fig. 7. Sample schedule of staff.

the division may not distribute nicely among all staff due to the teaching hours pre-assigned by the HOD. The true optimal solution may never be known unless the iterative solution is allowed to execute without setting an iteration limit. For all practical purposes, a final cost in the neighborhood of 20 is considered very good.

C. Comparison with Results from Genetic Algorithms

Since the success implementation of the SA-based workload scheduling project, there has been an attempt to implement the same project using genetic algorithms (GA) [12]. Based on the initial GA runs, the results are in general inferior to those obtained from the SA algorithm and the program takes about two days to run.

Nevertheless, the GA-based project did reveal that the technique might be suitable for the implementation of the workload scheduling project. The execution time may be reduced if the length of the DNA array, which stores the binary genes, can be shortened and the solution process is fine-tuned for optimal performance.

VII. CONCLUSION

This study uses simulated annealing to solve a staff workload scheduling problem. It also presented some possible cooling schedules.

Based on the percent schedules that can be implemented and the average cost of feasible schedules, the hybrid geometric function with reheating is selected for the actual implementation. The execution time although is an important consideration, is not a major issue since a savings of couple of hours does not make any difference when the software is invoked for overnight execution.

Simulated annealing, although requires long computational time, is still acceptable. More importantly, simulated annealing has the advantage of high solution quality and it is easy to implement.

REFERENCES

[1] D. C. Rich, "A Smart Genetic Algorithm for University Timetabling," in *Proc. Int. Conf. on the Practice and Theory of Automated Timetabling PTAT'95*, Selected Papers, pp. 181-197, Springer-Verlag, 1995.

[2] M. A. S. Elohamed, P. Coddington, and G.,Fox, "A Comparison of Annealing Techniques for Academic Course Scheduling," in *Proc. Second Int. Conf. on the Practice and Theory of Automated Timetabling*, *PTAT'97*, pp. 92-112, 1997.

[3] J. Sheung, A. Fan, and A. Tang, "Time tabling using genetic algorithm and simulated annealing," in *Proc. IEEE TECON'93*, pp. 448-451, Beijing, China, 1993.

[4] M. Hasan, T. AlKhamis, and J. Ali, "A comparison between simulated annealing, genetic algorithm and tabu search methods for the unconstrained quadratic Pseudo-Boolean function, " *Computer & Industrial Engineering*, vol. 38, no. 3, pp323-341, Oct. 2000.

[5] H. Youssef, S. M. Sait, and H. Adiche, "Evolutionary algorithms, simulated annealing and tabu search: a comparative study," *Engineering Applications of Artificial Intelligence*, vol. 14, no. 2, pp. 167-181, Apr. 2001.

[6] K. P. Ng, *Combinatorial Optimization: Using Hopfield Nets, Simulated Annealing, Boltzmann Machines and Genetic Algorithms*, NTU Project Q155.NKP, 1994.

[7] P. J. M. Laarhoven, *Simulated Annealing: Theory and Applications*, Kluwer Academic Publishers, 1987.

[8] S. M. Sait and H. Youssef, *VLSI Physical Design and Automation: Theory and Practice*, McGraw-Hill Book Company, 1995.

[9] H. B. Gooi and A. N. Arunasalam, "Automatic generation of one-line diagrams, " in *Proc. Int. Power Engineering Conf.*, pp. 26-30, Singapore 1993.

[10] M. D. Huang, F. Romeo, and A. K. Sangiovanni-Vincentelli, "An efficient general cooling schedule for simulated annealing," in *Proc. IEEE International Conf. on Computer-Aided Design*, pp. 381-384, Santa Clara, Nov. 1986.

[11] K. Anderson, R. V. V. Vidal, and V. B. Iverson, "Design of a teleprocessing communication network using simulated annealing", in *Proc. IEEE Int. Conf. on Computer Aided Design*, pp. 381-384, Santa Clara, 1986.

[12] W. K. Tan and K. M. Lee, *Development of an Automatic Optimal Timetable Planning Software for the School of Electrical and Engineering*, B.Eng. Final Year Project Report, School of Electrical & Electronic Engineering, Nanyang Technological University, 2001.

M. L. Ng received the B.Eng. and M.Eng. degrees in Electrical and Electronics Engineering from Nanyang Technological University, Singapore, in 1999 and 2001, respectively.

During his Masters, he researched on various optimization algorithms such as simulated annealing and genetic algorithm. He had used his research results to solve the staff workload scheduling problem in the School of EEE, NTU, Singapore.

Since 2002, he has been with ST Training & Simulation, Singapore, as a Software Engineer where he was primarily involved in the development of simulation projects.

H. B. Gooi received his B.Sc. from National Taiwan University, M.Sc. from University of New Brunswick and Ph.D. from Ohio State University in 1978, 1980, and 1983, respectively.

From 1983 to 1985, he was an Assistant Professor in the EE Department at Lafayette College, Pennsylvania. From 1985 to 1991, he was a Senior Engineer with Empros (now Siemens), Minneapolis.

In 1991, he joined the School of EEE, NTU as a Senior Lecturer. Since 1999, he has been an Associate Professor. His current research interests are probabilistic reserve assessment and scheduling applications.

C. Lu received his B.Sc. from Tsinghua University, China, M.Sc. and Ph.D. from University of Manchester Institute of Science and Technology, UK in 1985, 1987, and 1990, respectively.

He joined the School of EEE, NTU as a Lecturer in 1991. Since 1999, he has been an Associate Professor. He is also with the Institute for Infocomm Research, Singapore since 2002. His current research interests are in the area of optical communication systems and networks. He has published over 100 journal and conference papers.