Benchmarking: SUN High Performance Computing MPI versus MPICH on two applications

Saleh Elmohamed, Roman Markowski, and Geoffrey Fox

Northeast Parallel Architectures Center
Syracuse University, Syracuse, NY 13244

{saleh, roman, gcf}@npac.syr.edu

January 10, 1998

ABSTRACT

1 Introduction

... ...

2 The Timetabling Problem

... ...

3 Mean-Field Annealing

... ...

4 The Rule-Based System

... ...

5 Simulated Annealing

Simulated annealing (SA) has been widely used for tackling different combinatorial optimization problems, particularly academic scheduling [4, 7, 8]. The basic algorithm is described in Figure 1. The results obtained depend heavily on the cooling schedule used. We initially used the most commonly known and used schedule, which is the geometric cooling, but later tried adaptive cooling, as well as the method of geometric reheating based on cost [3].

A comprehensive discussion of the theoretical and practical details of SA is given in [1, 27, 32, 34]. It suffices here to say that the elementary operation in the Metropolis method for a combinatorial problem such as scheduling is the generation of some new candidate configuration, which is then automatically accepted if it lowers the cost (C), or accepted with probability $\exp(-\Delta C/T)$, where T is the temperature, if it would increase the cost by $\Delta(C)$. Also, in Figure 1, s is the current schedule and $s^{'}$ is a neighboring schedule obtained from the current neighborhood space ($\mathcal{N}_s$) by swapping two classes in time and/or space.

Thus the technique is essentially a generalization of the local optimization strategy, where, at non-zero temperatures, thermal excitations can facilitate escape from local minima.

The SA algorithm has advantages and disadvantages compared to other global optimization techniques. Among its advantages are the relative ease of implementation, the applicability to almost any combinatorial optimization problem, the ability to provide reasonably good solutions for most problems (depending on the cooling schedule and update moves used), and the ease with which it can be combined with other heuristics, such as expert systems, forming quite useful hybrid methods for tackling a range of complex problems. SA is a robust technique, however, it does have some drawbacks. To obtain good results the update moves and the various tunable parameters used (such as the cooling rate) need to be carefully chosen, the runs often require a great deal of computer time, and many runs may be required.

Depending on the problem to which it is applied, SA appears competitive with many of the best heuristics, as shown in the work of Johnson *et al.* [21].

1. Generate an initial schedule s.
2. Set the initial best schedule $s^* = s$.
3. Compute cost of s : $C(s)$.
4. Compute initial temperature T_0.
5. Set the temperature $T = T_0$.
6. While *stop criterion* is not satisfied do:
   (a) Repeat *Markov chain length* (M) times:
      i. Select a random neighbor $s^{'}$ to the current schedule, $(s^{'} \subset \mathcal{N}_s)$.
      ii. Set $\Delta(C) = C(s^{'}) - C(s)$.
      iii. If $(\Delta(C) \leq 0$ {downhill move}):
         - Set $s = s^{'}$.
         - If $C(s) < C(s^*)$ then set $s^* = s$.
      iv. If $(\Delta(C) > 0$ {uphill move}):
         - Choose a random number r uniformly from $[0, 1]$.
         - If $r < e^{-\Delta(C)/T}$ then set $s = s^{'}$.
   (b) Reduce (or update) temperature T.
7. Return the schedule s^*.

Figure 1: The Simulated Annealing Algorithm

5.1 Timetabling Using the Annealing Algorithm

The most obvious mapping of the timetabling problem into the SA algorithm involves the following constructs:

1. a **state** is a timetable containing the following sets:
   - P: a set of professors.
   - C: a set of classes.
   - S: a set of students.
   - R: a set of classrooms.
   - I: a set of time intervals.
2. a **cost** or "energy" $\mathrm{E}(P, C, S, R, I)$ such that:
   - $\mathrm{E}(P)$: is the cost of assigning more than maximum number of allowed classes M_p to the same professor, plus scheduling one or more classes that cause a conflict in the professor's schedule.

- E(C): is the cost of scheduling certain classes at/within the same time period in violation of the exclusion constraint, for example.
- E(S): is the cost of having two or more classes conflict in time; plus cost of having in the schedule one or more classes that really don't meet the student's major, class requested, or class requirements; plus the cost of not having the classes evenly spread out over the week, etc.
- E(R): is the cost resulting from assigning room(s) of the wrong size and/or type to a certain class.
- E(I): is the cost of having more or less time periods than required, plus cost of an imbalanced class assignments (a certain period will have more classes assigned to than others, etc.).

3. A **swap** (or a move) is the exchange of one or more of the following: class c_i with class c_j in the set C with respect to time periods I_i and I_j, and/or with respect to classroom R_i and R_j, respectively. Generally, this step is referred to as class swapping.

Along with all of the necessary constraints, the simulated annealing algorithm also takes as input data the following: the preprocessor output in the form of lists of scheduled and non-scheduled classes and their associated professors and room types, a list of rooms provided by the registrar's office, a department to building distance matrix, a list of students and their class preferences, and a list of classes that are not allowed to be scheduled simultaneously.

To use simulated annealing effectively, it is crucial to use a good cooling schedule, and a good method for choosing new trial schedules, in order to efficiently sample the search space. We have experimented with both these areas, which are discussed in the following sections.

5.2 The Annealing Schedules

Three annealing schedules have been used in our experiments to update the temperature of the SA algorithm in Figure 1: geometric cooling, adaptive cooling, and adaptive reheating as a function of cost.

The first schedule we have used is **geometric cooling**, where the new temperature ($T^{'}$) of the SA algorithm is computed using

$$T^{'} = \alpha T \ , \tag{1}$$

where α ($0 < \alpha < 1$) denotes the cooling factor. Typically the value of α is chosen in the range 0.90 to 0.99. This cooling schedule has the advantage of being well understood, having a solid theoretical foundation, and being the most widely used annealing schedule. Our results obtained from using this standard cooling schedule will be used as a baseline for comparison with those using the other two schedules, which allow the rate of cooling to be varied.

The second annealing schedule we used is the method of **reheating as a function of cost** (RFC), which was used for timetabling by Abramson *et al.* [3], but the ideas behind it are due to Kirkpatrick *et al.* [22, 23] and White [35]. Before introducing this schedule we first summarize a few relevant points on the concept of specific heat (C_H). Specific heat is a measure of the variance of the cost (or energy) values of states at a given temperature. The higher the variance, the longer it presumably takes to reach equilibrium, and so the longer one should spend at the temperature, or alternatively, the slower one should lower the temperature.

Generally, in combinatorial optimization problems, phase transitions [16, 26] can be observed as sub-parts of the problem are resolved. In some of the work dealing with the traveling salesman problem using annealing [24], the authors often observe that the resolution of the overall structure of the solution occurs at high temperatures, and at low temperatures the fine details of the solution are resolved. As reported in [3], applying a reheating type procedure, depending on the phase, would allow the algorithm to spend more time in the low temperature phases, thus reducing the total amount of time required to solve a given problem.

In order to calculate the temperature at which a phase transition occurs, it is necessary to compute the specific heat of the system. A phase transition occurs at a temperature $T(C_H^{max})$ when the specific heat is maximal (C_H^{max}), and this triggers the change in the state ordering. If the best solution found to date has a high energy or cost then the super-structure may require re-arrangement. This can be done by raising the temperature to a level which is higher than the phase transition temperature $T(C_H^{max})$. Generally, the higher the current best cost, the higher the temperature which is required to escape the local minimum. To compute the aforementioned maximum specific heat, we employ the following steps [3, 34, 27].

At each temperature T, the annealing algorithm generates a set of configurations $\mathcal{C}(T)$. Let C_i denote the cost of configuration i, $C(T)$ is the average cost at temperature T, and $\sigma(T)$ is the standard deviation of the cost at T.

At temperature T, the probability distribution for configurations is:

$$P_i(T) = \frac{e^{\frac{-C_i}{kT}}}{\sum_j e^{\frac{-C_j}{kT}}} \, . \tag{2}$$

The average cost is computed as:

$$< C(T) >= \sum_{i \in \mathcal{C}} C_i P_i(T) \, . \tag{3}$$

Therefore, the average square cost is:

$$< C^2(T) >= \sum_{i \in \mathcal{C}} C_i^2 P_i(T) \, . \tag{4}$$

The variance of the cost is:

$$\sigma^2(T) =< C^2(T) > - < C(T) >^2 \; . \tag{5}$$

Now, the specific heat is defined as:

$$C_H(T) = \frac{\sigma^2(T)}{T^2} \, . \tag{6}$$

The temperature $T(C_H^{max})$ at which the maximum specific heat occurs, or at which the system undergoes a phase transition, can thus be found.

Reheating sets the new temperature to be

$$T = K \cdot C_b + T(C_H^{max}) \, , \tag{7}$$

where K is a tunable parameter and C_b is the current best cost. Reheating is done when the temperature drops below the phase transition (the point of maximum specific heat) and there has been no decrease in cost for a specified number of iterations, i.e. the system gets stuck in a local minimum. Reheating increases the temperature above the phase transition (see equation 7), in order to produce enough of a change in the configuration to allow it to explore other minima when the temperature is reduced again.

The third cooling schedule we have tried is **adaptive cooling**. In this case, a new temperature is computed based on the specific heat, i.e. the standard deviation of all costs obtained at the current T. The idea here is to keep the system close to equilibrium, by cooling slower close to the phase transition, where the specific heat is large. There are many different ways of implementing this idea, we have chosen the approach taken by Huang *et al.* [18], which was shown to yield an efficient cooling schedule. Let T_j denote the current temperature, at step j of the annealing schedule. After calculating $\sigma(T_j)$ from equation 5, the new temperature T_{j+1} is computed as follows:

$$T_{j+1} = T_j \cdot e^{-\frac{aT_j}{\bar{\sigma}(T_j)}} , \tag{8}$$

where a is a tunable parameter. Following suggestions by Otten and van Ginneken [27] and Diekmann *et al.* [6], $\sigma(T_j)$ is smoothed out in order to avoid any dependencies of the temperature decrement on large changes in the standard deviation σ. We used the following standard method to provide a smoothed standard deviation $\bar{\sigma}$:

$$\bar{\sigma}(T_{j+1}) = (1-\omega)\sigma(T_{j+1}) + \omega\sigma(T_j)\frac{T_{j+1}}{T_j} \tag{9}$$

and set ω to 0.95. This smoothing function is used because it follows (from the form of the Boltzmann distribution, see [32, 35]) that it preserves the key relationship:

$$\frac{d}{dT}C(T) = \frac{\bar{\sigma}^2(T)}{T^2} = C_H \tag{10}$$

Note that reheating can be used in conjunction with any cooling schedule. We have used it with adaptive cooling.

5.3 The Choice of Moves

The performance of any application of simulated annealing is highly dependent on the method used to select a new trial configuration of the system for the Metropolis update. In order for the annealing algorithm to work well, it must be able to effectively sample the parameter space, which can only be done with efficient moves.

The simplest method for choosing a move is to swap the rooms or timeslots of two randomly selected classes. However this is extremely inefficient, since most of the time random swapping of classes will increase the overall cost, especially if we are already close to obtaining a valid solution (i.e. at low temperature), and will likely be rejected in the Metropolis procedure. This low acceptance of the moves means this simple method is very inefficient, since a lot of computation is required to compute the change in cost and do the Metropolis step, only to reject the move.

What is needed is a strategy for choosing moves that are more likely to be accepted. A simple example is in the choice of room. If we randomly choose a new room from the list of all rooms, it will most likely be rejected, since it may be too small for the class, or an auditorium when, for example, a laboratory is needed. One possibility is to create a subset of all the rooms which fulfill the hard constraints on the room for that particular class, such as the size and type of room. Now we just make a random selection for a room for that class only from this subset of feasible rooms, with an acceptance probability that is sure to be much higher. In addition, each class in our data set comes with a "type-of-space-needed" tag which is used along with other information to assign the class to the right room. This effectively separates the updates into independent sets based on room type, so for example, laboratories are scheduled separately from lectures. In our method we carry out the scheduling of lectures first, followed by scheduling of laboratories making sure that during the course of this process no lecture and its associated laboratory are scheduled in the same time period.

In effect, we have embedded a simple expert system into the annealing algorithm in order to improve the choice of moves, as well as using a more complex expert system as a preprocessor for the annealing step. When used to choose the moves for annealing, the main function of the rule-based system is to ensure that all the trial moves satisfy the hard constraints. Many of the rules dealing with the medium and soft constraints are softened or eliminated, since reducing the cost of these constraints is done using the Metropolis update in the annealing algorithm.

Another of the modifications to the rule-based system is that while the version used in the preprocessor is completely deterministic, the version used in choosing the moves for annealing selects at random from multiple possibilities that satisfy the rules equally well. This extra freedom in choosing new schedules, plus the extra degree of randomness inherent in the annealing update, helps prevent the system getting trapped in a local minimum before it can reach a valid schedule, which is the problem with the standard deterministic rule-based system.

To improve further on the move strategy, we can take the subset of possible move choices that we have created for each class, and choose from them probabilistically rather than randomly. There may be certain kinds of moves that are more likely to be effective, so our move strategy is to select these moves with a higher probability. For example, swapping a higher level class (e.g. graduate) with a lower level class (e.g. a first or a second year type) generally has a higher acceptance, since there is little overlap between students taking these classes. Furthermore, we have experimented with two kinds of swaps, those that only involve classes offered by the same department or college and the second, swaps between classes of different departments and colleges.

Generally, the swap methods we have taken here can be considered as heuristics for pruning the neighborhood or narrowing the search space, which provides much more efficient moves and in turn an overall improvement in the results.

6 Experimental Results

Our computations were done with a number of goals in mind. The main objective was to provide a schedule which satisfied all hard constraints and minimized the cost of medium and soft constraints, using real-life data sets for a large university. We also aimed to find an acceptable set of annealing parameters and move strategies for general timetabling problems of this kind, and to study the effect of using a preprocessor to provide the annealing program with a good starting point. Finally, we wanted to make a comparison of the performance of the three different cooling schedules, geometric cooling, adaptive cooling, and reheating based on cost.

We spent quite some time finding optimal values for the various parameters for the annealing schedule, such as the initial temperature, the parameters controlling the rate of cooling (α for geometric cooling, a for adaptive cooling) and reheating (K), and the number of iterations at each temperature (for more details, see Ref. [11]). Johnson *et al.* [21] noted in their SA implementation for the traveling salesman problem (TSP) that the number of steps at each temperature (or the size of the Markov chain) needed to be at least proportional to the "neighborhood" size in order to maintain a high-quality result. From our experiments we found the same to be true for the scheduling problem, even though it is very different from the TSP. Furthermore, in a few tests for one semester we fixed the number of classes and professors but varied the number of rooms and time slots, and found that the final result improves as the number of iterations in the Markov chain becomes proportional to a combination of the number of classes, rooms and time slots. We also observed the same behavior when we fixed the number of rooms and time slots but varied number of classes.

Our study case involved real scheduling data covering three semesters at Syracuse University. The size and type of the three-semester data is shown in Table 1. Nine types of rooms were used: auditoriums, classrooms, computer clusters, conference rooms, seminar rooms, studios, laboratories, theaters, and unspecified types. Staff and teaching assistants are considered part of the set of professors. Third semester (summer) data was much smaller than other semesters, however, there were additional space and time constraints and fewer available rooms. Our data was quite large in comparison to data used by other researchers. For example, high school data used by Peterson and colleagues [13, 14] consists of approximately 1000 students, 20 different possible majors, and an overall periodic school schedule (over weeks). In the case of Abramson *et al.* [2], their data set was created randomly and was relatively small, and they stated that problems involving more than 300 tuples were very difficult to solve.

Table 1 lists all major components of the data we have used. Timetabling problems can be characterized by their *sparseness*. After the required number of lessons N_l have been scheduled, there will be $N_{sp} = (N_x N_t - N_l)$ spare space-time slots, hence, the sparseness ratio of the problem is defined as the ratio $N_{sp}/(N_x N_t)$. The denser the problem, the lower the sparseness ratio, and the harder the problem is to solve. Also, for dense problems, there is an additional correlation involving the problem size. Table 2 shows the sparseness of the three-semester data. For university scheduling, the sparseness ratio generally decreases as the data size (particularly the number of classes) increases, so the problem becomes harder to solve. Including student preferences makes the problem much harder, but these are viewed as medium constraints and thus are not necessarily satisfied in a valid solution.

Our overall results are shown in Tables 3 and 4. These tables show the percentage of classes that could be scheduled in accordance with the hard constraints. In each case (apart from the expert system, which is purely deterministic), we have done 10 runs (with the same parameters, just different random numbers), and the tables show the average of the 10 runs, as well as the best and worst results. The MFA results are different only due to having different initial conditions. Each simulated annealing run takes about 10-20 hours on a Unix workstation, while a single MFA run takes approximately an hour and an expert system run takes close to two hours.

As expected, each of the methods did much better for the third (summer) semester data, which has a higher sparseness ratio. Our results also confirm what we expected for the different cooling schedules for simulated annealing, in that adaptive cooling performs better than geometric cooling, and reheating improves the result even further.

When a random initial configuration is used, simulated annealing performs very poorly, even worse than the expert system (ES). However, there is a dramatic improvement in performance when a preprocessor is used to provide a good starting point for the annealing. In that case, using the best cooling schedule of adaptive cooling with reheating as a function of cost, we are able to find a valid class schedule every time.

In the case of mean-field annealing, the overall results are generally below those of SA and ES. In addition, we have found in the implementation of this method that the results were quite sensitive to the size of the data as well the type of constraints involved. If we confine ourselves to the set of hard constraints, the results are as good as or even better than the other methods. However if we take into account the medium and soft constraints, that is, the overall cost function, this method does not perform as well.

Student preferences are included only as medium constraints in our implementation, meaning that these do not have to be satisfied for a valid solution, but they have a high priority. For the valid schedules we have produced, approximately 75% of the student preferences were satisfied. This is reasonably good (particularly since other approaches do not deal with student preferences at all), but we are working to improve upon this result.

7 Conclusions

We have successfully applied simulated annealing to the difficult problem of academic scheduling for a large university. Feasible schedules were obtained for real data sets, including student preferences, without requiring enormous computational effort.

Mean-field annealing works well for small scheduling problems, but does not appear to scale well to large problems with many complex constraints. For this problem, both simulated annealing and the rule-based system were more effective than MFA. It is more difficult to tune the parameters for MFA than for simulated annealing, and because of the complexity and size of the Potts neural encoding, there seems to be no clear way of preserving the state of a good initial configuration provided by a preprocessor when using MFA.

Using a preprocessor to provide a good initial state greatly improved the quality of the results for simulated annealing. In theory, using a good initial state should not be necessary, and any initial state should give a good result, however in practice, we do not have an ideal cooling schedule for annealing, or an ideal method for choosing trial moves and efficiently exploring the search space, and there are restrictions on how long the simulation can take. In general, for very hard problems with large parameter spaces that can be difficult to search efficiently, and for which very slow cooling would be much too time-consuming, we might expect that a good initial solution would be helpful. We used a fairly complex rule-based expert system for the preprocessor, however the type of preprocessor may not be crucial. Other fast heuristics could possibly be used, for example a graph coloring approach [25], or it may be possible to just utilize the schedule from the same semester for the previous year. A modified version of the rule-based system was used to choose the trial moves for the simulated annealing, and the high acceptance rate provided by this system was crucial to obtaining good results.

As expected, for the simulated annealing, adaptive cooling performed better than geometric cooling, and using reheating improved the results even further. The best results were obtained using simulated annealing with adaptive cooling and reheating as a function of cost, and with a rule-based preprocessor to provide a good initial solution. Using this method, and with careful selection of parameters and update steps, we were able to generate solutions to the class scheduling problem using real data for a large university. None of the other methods were able to provide a complete solution.

Our main conclusion from this work is that simulated annealing, with a good cooling schedule, optimized parameters, carefully selected update moves, and a good initial solution provided by a preprocessor, can be used to solve the academic scheduling problem at a large university, including student preferences. Similar approaches should prove fruitful for other difficult scheduling problems.

Acknowledgments

	First Semester	Second Semester	Third Semester
Rooms	509	509	120
Classes	3839	3590	687
Professors	1190	1200	334
Students	13653	13653	2600
Buildings	43	43	11
Schools/Colleges	20	21	17
Departments/ Course Prefix	143	141	108
Areas of Study (majors)	200	200	200

Table 1: Size of the data set for each of the three semesters.

Academic Time Period	sparseness ratio
First Semester	0.50
Second Semester	0.53
Third Semester	0.62

Table 2: The sparseness ratios of the problem for the data sets for each of the three semesters. Lower values indicate a harder problem.

Academic Time Period	Algorithm	Scheduled (average) %	highest Scheduled %	lowest Scheduled %
First Semester	SA (geometric)	65.00	67.50	56.80
	SA (adaptive)	67.80	70.15	61.20
	SA (cost-based)	70.20	72.28	68.80
	ES	76.65	76.65	76.65
	MFA	65.60	71.00	61.00
Second Semester	SA (geometric)	65.65	68.00	57.10
	SA (adaptive)	68.50	70.10	60.77
	SA (cost-based)	75.14	77.68	70.82
	ES	79.00	79.00	79.00
	MFA	67.20	75.00	65.00
Third Semester	SA (geometric)	83.10	86.44	68.50
	SA (adaptive)	85.80	89.00	70.75
	SA (cost-based)	91.20	95.18	85.00
	ES	96.80	96.80	96.80
	MFA	88.00	95.00	82.00

Table 3: Percentage of classes scheduled using the different methods. The averages and highest and lowest values were obtained using 10 independent runs for simulated annealing (SA) and mean-field annealing (MFA). The expert system (ES) is deterministic so the results are from a single run. *No preprocessor was used with the three methods.*

Academic Time Period	Algorithm	Scheduled (average) %	highest Scheduled %	lowest Scheduled %
First Semester	SA (geometric)	93.90	95.12	85.20
	SA (adaptive)	98.80	99.20	95.00
	SA (cost-based)	100.0	100.0	100.0
Second Semester	SA (geometric)	95.00	98.95	89.40
	SA (adaptive)	99.00	99.50	98.50
	SA (cost-based)	100.0	100.0	100.0
Third Semester	SA (geometric)	97.60	98.88	90.90
	SA (adaptive)	100.0	100.0	100.0
	SA (cost-based)	100.0	100.0	100.0

Table 4: Percentage of scheduled classes, averaged over 10 runs of the same initial temperature and other parameters, for three terms using simulated annealing *with an expert system as preprocessor.*

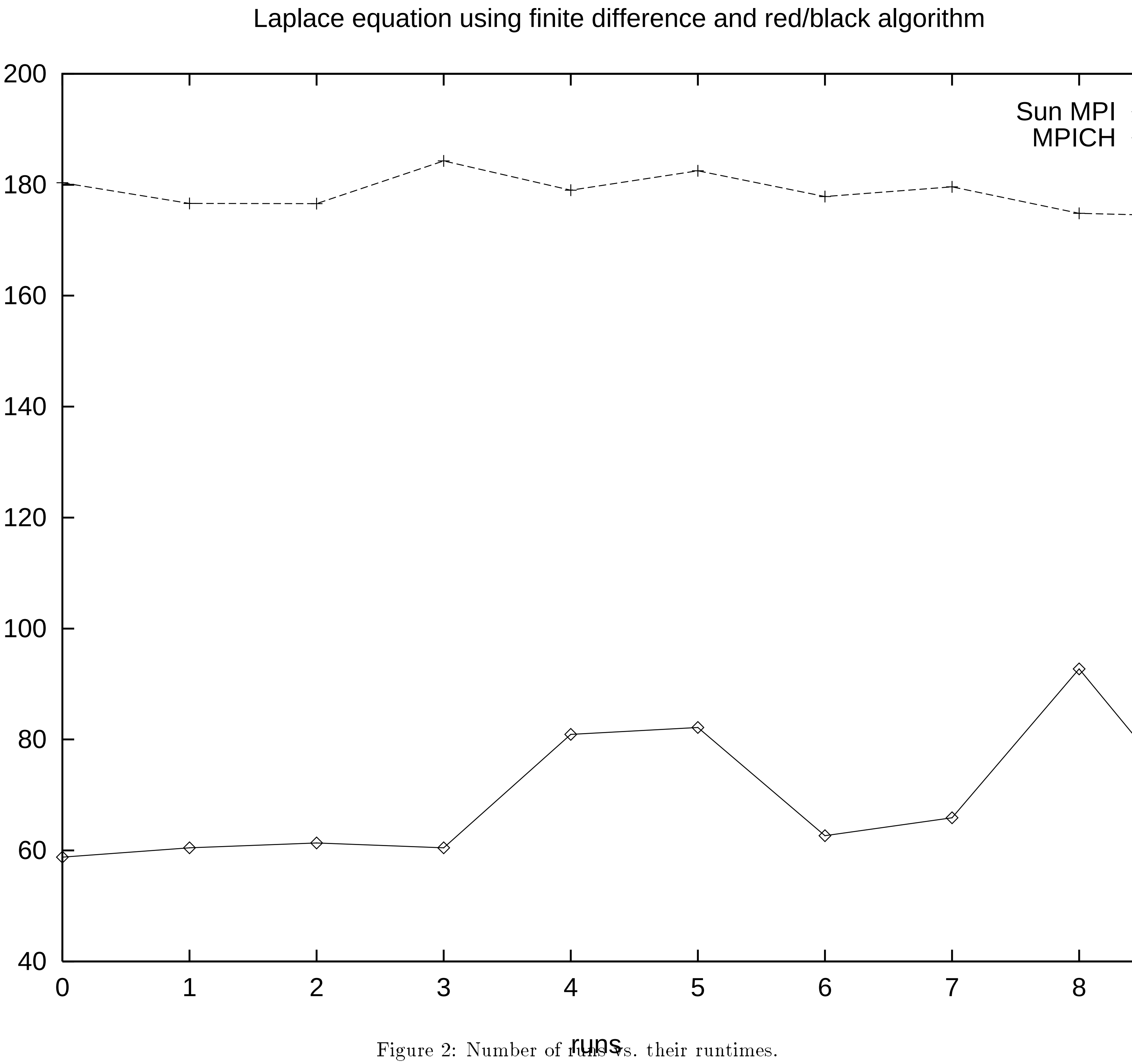

Figure 2: Number of runs vs. their runtimes.

References

[1] Aarts, E. H., J. Korst, and P. J. van Laarhoven, "Simulated annealing," in Local Search in Combinatorial Optimization, E. H. Aarts and J. K. Lenstra (eds.), John Wiley and Sons. (to appear in 1997)

[2] Abramson, D., "Constructing school timetables using simulated annealing: sequential and parallel algorithms," *Management Science*, 37(1), 98-113, 1991.

[3] Abramson, D., H. Dang, and M. Krishnamoorthy, "An Empirical Study of Simulated Annealing Cooling Schedules," Griffith Univ. report, Nathan, Qld, Aus. 1994; "Simulated Annealing Cooling Schedules for the School Timetabling Problem," submitted to Asia Pacific Journal of Operations Research, 1996.

[4] Applied Simulated Annealing, Rene Vidal (ed), Lecture Notes in Economics and Mathematical Systems no. 396, Springer-Verlag, 1993.

[5] de Werra, D., "An introduction to timetabling," *European Journal of Operational Research*, 19, 151-162, 1985.

[6] Diekmann, R., R. Lüling, and J. Simon, "Problem independent distributed simulated annealing and its applications," in Applied Simulated Annealing, R. V. Vidal (ed), Lecture Notes in Economics and Mathematical Systems, Springer 1993.

[7] Dowsland, K., Using Simulated Annealing for Efficient Allocation of Students to Practical Classes. Stat and OR Group, EBMS, Univ College of Swansea, UK, 1993.

[8] Dowsland, K. and J. Thompson, "Variants of Simulated Annealing for the Examination Timetabling Problem," Working Paper, Statistics and OR Group, European Business Management School, Swansea SA2 8PP, UK. (1994)

[9] Eiselt H. A., and G. Laporte, "Combinatorial Optimization Problems with Soft and Hard Requirements," *J. Operational Research Society*, vol. 38, No. 9, pp. 785-795, 1987.

[10] Elmohamed, S., G. C. Fox, P. Coddington, "Course Scheduling using Mean-Field Annealing, Part I: algorithm and Part II: implementation," Northeast Parallel Architectures Center technical report SCCS-782, Syracuse University, 1996.

[11] Elmohamed, S., P. Coddington, G.C. Fox, "Academic Scheduling using Simulated Annealing with a Rule-Based Preprocessor", Northeast Parallel Architectures Center technical report SCCS-781, Syracuse University, 1997.

[12] Gee, Andrew, private communication.

[13] Gislén, L., B. Söderberg, C. Peterson, "Teachers and Classes with Neural Nets," *Inter. Jr. of Neural Systems* 1, 167 (1989).

[14] Gislén, L., B. Söderberg, C. Peterson, "Scheduling High Schools with Neural Nets," Lund University Preprint LU-TP-91-9, Lund, Sweden (1991), (appeared in Neural Computation 1992).

[15] Hertz, J., A. Krogh and R. Palmer, Introduction to the Theory of Neural Computation, Addison-Wesley, Redwood City, Ca. (1991)

[16] Hogg, T., B. Huberman, and C. Williams (editors), Artificial Intelligence, special issue on *Phase transitions and the search space*, 81, 1996.

[17] Hopfield, J. J., and D. W. Tank, "Neural Computation of Decisions in Optimization Problems," *Biological Cybernetics* 52, 141 (1985).

[18] Huang, M., F. Romeo, and A. Sangiovanni-Vincentelli, "An efficient general cooling schedule for simulated annealing," IEEE International Conf. on Computer Aided Design (ICCAD), 1986, pp. 381-384.

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

[20] Johnson, D., C. Aragon, L. McGeoch, and C. Schevon, "Optimization by Simulated Annealing: an Experimental Evaluation, Part II (Graph Coloring and number partitioning)," *Operations Research* 39, No. 3, 865-892 (1991).

[21] Johnson, D., and L. McGeoch, "The Traveling Salesman Problem: A Case Study in Local Optimization," in Local Search in Combinatorial Optimization, E. H. Aarts and J. K. Lenstra (eds.), Wiley and Sons. (to appear in 1997).

[22] Kirkpatrick, S., C. D. Gelatt, Jr., and M. P. Vecchi, "Optimization by Simulated Annealing," *Science* 220 (13 May 1983), 671-680.

[23] Kirkpatrick, S., "Optimization by simulated annealing: Quantitative studies," *J. Stat. Physics* 34 (1984), 976-986.

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

[25] Miner, S., S. Elmohamed, and H. W. Yau, "Optimizing Timetabling Solutions Using Graph Coloring," 1995 NPAC REU program, NPAC, Syracuse University, 1995.

[26] Mouritsen, O. G., Computer Studies of Phase Transitions and Critical Phenomena. Springer-Verlag, Berlin, 1984.

[27] Otten, R., and L. van Ginneken, The Annealing Algorithm. Kluwer Academic Publishers, 1989.

[28] Peterson, C., and B. Söderberg, "Artificial Neural Networks and Combinatorial Optimization Problems," To appear in *Local Search in Combinatorial Optimization,* E.H.L. Aarts and J.K. Lenstra, eds. Wiley and Sons. (to appear 1997)

[29] Peterson, C., and B. Söderberg, "A New Method for Mapping Optimization Problems onto Neural Nets", International Jr. of Neural Systems 1, 3 (1989).

[30] Schaerf, A., "A survey of automated timetabling," Department of Software Technology, Report CS-R9567, CWI, Amsterdam, The Netherlands.

[31] Simmen, Martin, Personal Communication.

[32] Sorkin, Gregory, Theory and Practice of Simulated Annealing on Special Energy Landscapes, PhD. Thesis, Dept. of Electrical Engineering and Computer Science, University of California, Berkeley, July 1991.

[33] Thompson, J., and K. Dowsland, "General Cooling Schedules for Simulated Annealing Based Timetabling Systems," Proceedings of the 1st International Conf. on the Practice and Theory of Automated Timetabling, Napier Univ., Edinburgh 1995.

[34] van Laarhoven, P. J. and E. H. Aarts, Simulated Annealing: Theory and Applications. D. Reidel, Dordrecht (1987).

[35] White, S. R., "Concepts of scale in simulated annealing," *Proceedings of the IEEE Int. Conference on Circuit Design*, pp 646-651, 1984.