International Journal of Mathematics and
Computer Science, **19**(2024), no. 4, 1129–1135

Using New Metrics to find a good Examination Schedule

P. Smith, P. Wang

Department of Mathematics and Statistics
St. Francis Xavier University
Antigonish, Nova Scotia, Canada B2G 2W5

email: pwang@stfx.ca

(Received March 15, 2024, Accepted April 20, 2024,
Published June 1, 2024)

Abstract

The problem of finding an optimal examination schedule is known to be an NP-complete problem. Researchers often employ heuristics such as simulated annealing and genetic algorithms to find good examination schedules. In this paper, we compare the three heuristic approaches from the perspective of students.

1 Introduction

The examination scheduling problem is a very well-known combinatorial optimization problem. Carter [1] gave a comprehensive review of examination timetabling methods. It is the process of assigning examinations to a particular slot in the timetable and a particular room. That is, a number of examinations must be assigned to a fixed number of time slots subject to the following constraints:
(i) Hard constraints – two examinations cannot be assigned in the same time slot if one or more students must take both examinations.
(ii) Students comfort constraints – most universities try to minimize the number of students having two examinations in consecutive time slots.

Key words and phrases: Graph theory, simulation annealing, generic algorithm.
AMS (MOS) Subject Classifications: 05Cxx.
ISSN 1814-0432, 2024, http://ijmcs.future-in-tech.net

(iii) Physical constraints – Some physical constraints such as a fixed number of classrooms at any given time slot.
(iv) Faculty constraints – For the large size classes, it would be beneficial for instructors to arrange their examinations early in the examination period so that the instructors can hand in the final grades before the deadlines. At St. Francis Xavier University (STFX), there are approximately 3500 students and about 400 courses offered. Examinations are held over a 10-day period with 3 time slots per day. In this paper, three different heuristic algorithms are employed to find an optimal solution or a near optimal solution based on two different metric.

2 Goodness Metric and three algorithms

We shall use the graph presentation to organize our data where a note of graph is a class at STFX and two notes are joined by an edge if and only if there is at least one student enrolled in both classes. It follows that the scheduling problem becomes a graph colouring problem. That is, can we colour all of notes with 30 colours?

2.1 Common metric and three approaches

In order to find an optimal schedule, a measurement on the "Goodness" of a particular examination schedule shall be defined. We shall use students "discomfort"level as the weight for each student. Let i/j denote i examinations in j days. We use the weight defined in Table 1. Then the cost function of any particular examination schedule (S) is defined as $Total\_weight(S) = \sum_{each\ student} w_i$. In turn, an optimal examination schedule is a schedule with the minimum cost.

First, we try a greedy algorithm. Based on the university's current examination schedule, the greedy program goes through all the courses sequentially. Take a course from the course list (sequentially). Then calculate the changes in the Total_weight if it would be moved to another time slot providing there is no conflict is created. This is done by a classical graph colouring method. That is, a note can be moved from one colour set to another colour set if this note does not join to any other note in both colour sets. Among all the possible moves, the examination will be moved permanently to a new time slot if it attaints to the lowest Total_weight. A substantial improvement is made by the greedy approach (see Table 1). Clearly, the greedy algorithm is relatively easy to code and produced a much-improved schedule. In fact, only

12,000 examination schedule evaluations are needed so it is a time efficient program.

Table 1: Comparison table 1

	6/2	**5/2**	**3/1 and 4/2**	**3/1**	**4/2**	**2/1**	**Total_weight**
Weight	60	50	40	30	20	10	
Original	0	0	12	28	7	839	14185
Greedy	0	0	2	1	3	443	8620
SA	0	0	0	1	2	331	4740

It is well known that the greedy algorithm may not produce an optimal solution. Simulated annealing is one of the techniques that can be used to overcome this. It is a means of finding good solutions to combinatorial optimization problems [2]. Ross and Corne [3] have an article on the application of simulated annealing on this topic.

In a simulation annealing program, a move is a transition from one state of the solution space to another. In our program, a prospective move is found by randomly selecting an examination from the currant schedule and place it into a randomly selected time slot without creating a conflict. We call the schedules before and after a move the old schedule and the new schedule respectively. The cost of the move is calculated as follows: cost = Total_weight of the new schedule − Total_weight of the old schedule. Initially, the program favours cost decreasing moves since the goal is to minimum the total weight or obtain a near minimum weight schedule. However, by allowing only such moves, it is likely that the final solution is a local minimum, rather than a global minimum. To escape from a local minimum, cost increasing moves must be made. If a move decreases the cost, it is always accepted. Otherwise, it is accepted with probability $P(\Delta E) = e^{-\Delta E/T}$, where T is the temperature and ΔE is the increase in cost that would result from this prospective move. Initially, T is large and virtually all moves are accepted. Gradually, T is decreased thus decreasing acceptance of cost increasing moves. Eventually, the system will reach a state in which very few moves are accepted. In such a state, the system is said to be frozen. The sequence of decreasing temperatures is called the annealing schedule. The next temperature is obtained by $T_{n+1} = \alpha T_n$, where α is the cooling rate. We obtain our best result in the test runs by using $\alpha = 0.9$ and $T_0 = 1000$. A limited number of moves are accepted at each temperature level, and we use max_moves = 65 as a limit. Moreover, there is a limit for the number of moves attempted

at each temperature and we use max_attempted = 800. Once the maximum number of attempted moves has been reached or the maximum number of moves has been reached the temperature is lowered and a new iteration begins. The process stops if the number of accepted moves has not reached its maximum in a given number of iterations. That is, we consider the system frozen if less than one in 12.31 (800/65) attempted moves is accepted. All the above parameters are chosen based on our experiment results. The following diagram (see Figure 1) illustrates the distinguished feature, escape the local minimum, of the simulated annealing approach. As one can see from the diagram, Total_weight increases initially to escape the local minimums and decreases very slowly in the fine turning stage (after 2,000 moves). Over 18,000 moves were accepted. The diagram only shows the first 4,000 moves because Total_weight changes very little after 4,000 moves. In order to evaluate the cost of a move, the Total_weight must be calculated. Computationally, this could be very expensive. Hence our program is designed in such a way that it only changes the Total_weight for those students whose examinations have been rescheduled. The result (see Table 1) of SA program is impressive. The Total_weight has been effectively reduced by almost one half of the one obtained from the greedy approach.

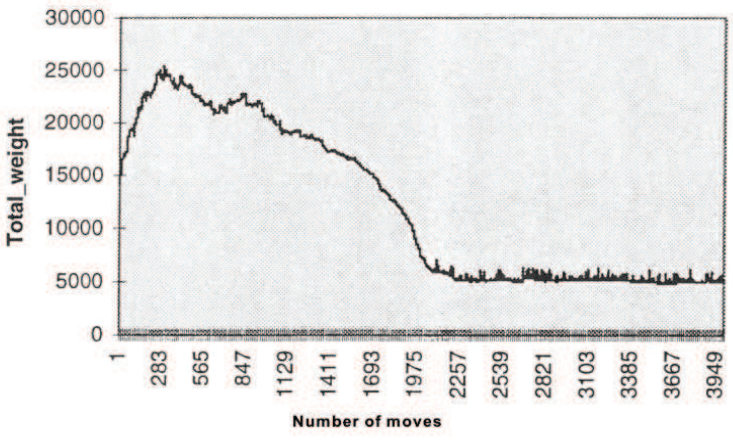

Figure 1: Number of moves

A Genetic algorithm can be used to solve many optimization problems (see [4], [6] and [5]). In the following we describe how we use the generic algorithm to optimize the examination scheduling problem.

1. **Data structure**: Let $E(i)$ be a time slot in the examination schedule ($1 \leq E(i) \leq 30$) where the i-th examination ($1 \leq i \leq 400$) is assigned in time slot $E(i)$. This array is analogous to a chromosome in biology, which is a string of 400 genes and each of gene can take any allele from the set $\{1, 2, ..., 30\}$.

2. **Evaluation function**: The Total_weight was used as the fittest evaluation function. We maintain a pool of 30 examination schedules. The schedules that are the fittest will procreate and the rest eventually will die off, childless.

3. **Genetic operators**: As with biological parents, two examination schedules combine and contribute part of their characteristics to create their offspring, the new examination schedule. First, we rank the 30 schedules in descending order according to the metric we defined above. Secondly, we select two parents. They are selected according to their rank in the population, biases the probability $(1 - \frac{rank}{30})$ of that examination schedule being selected to parent the next generation. The crossover that produces the next generation is defined as follows:

Let E_1 and E_2 be two parent examination schedules and E' be the newly created offspring examination schedule. First, we fill $E'(i)$ by $E_1(i)$ if $E_1(i) = E_2(i)$ for $1 \leq i \leq 400$. Then we shall fill the remaining $E'(i)$ by either $E_1(i)$ or $E_2(i)$ providing this does not create a conflict in the time slot $E_1(i)$ in the new scheduling E'. If neither time slot $E_1(i)$ nor $E_2(i)$ can be used in the new scheduling E' without create a conflict, then we shall assign it to an extra time slot after the existed 30 time slots. Then a penalty of 2500 is added to the Total_weight of the schedules with an extra time slot. The idea is the new schedule will inherit the good traits (the common genes) from its parents assuming the parents are good schedules at this stage of optimization. Then we shall rank all schedules and eliminate the last one from the pool.

To avoid the chance of premature convergence (sometimes called inbreeding), a diversity must be introduced into the population. In each iteration, we select an examination randomly and assign it into a randomly selected time slot provided that there is no conflict in the schedule. More sophisticated mutations such as introduce diversity only between the two similar schedules will increase the computation time substantially.

The result shows that the Total_weight is reduced to 11,005 with 2 unscheduled examinations on average. These two unscheduled examinations added 5000 total penalty to the Total_weight. One can see that the genetic algorithm is better than the one obtained from the greedy algorithm but

not as good as the one obtained from the simulated annealing algorithm. However, it provides more alternatives to the registrar's office since there is a pool 30 good examination examination scheduling for the registrar to choose from. Considering we are trying to schedule 400 examinations for 3500 students and there are only a few students who are going to be affected with the two unscheduled examinations, the registrar's office may be willing to accommodate the few individual students who have conflicts in order to accommodate other requests.

2.2 A new metric

The penalties commonly used in defining Total_weight are mostly arbitrarily decided and more likely imposed by the administration's desire to minimize particular cases such as 3 examinations within 24 hours. The natural question is what is the "dream examination schedule"from a student's perspective? The obvious answer is that the five final examinations should be evenly distributed during the 10 day examination period. That is, one examination in every two days and the last examination in the very last time slot of examination period. Let $d_{i,j}$ be the number of time slots between the end of i-th examination and the beginning of the j-th examination. New_metric $=\sum_{each\ student} |5 - d_{i,j}|$. This implies the new metric will be 0 if a student has one examination in every two days. The metric will add penalty of 5 to the New_metric if a student has back to back examinations. We run the SA program again with this new metric (see the comparison result in Table 2).

Table 2: Comparison table 2

	6/2	**5/2**	**3/1 and 4/2**	**3/1**	**4/2**	**2/1**	New_metric
Old schedule	0	0	0	1	2	311	245,867
New schedule	0	0	0	1	0	395	238,848

As one can see from Table 2, there are 84 more students who have two examinations in the same day in the new schedule. However, the value of New_metric was reduced by more than 7,000. We checked the bottom 200 students with the worst examination schedules in both new and old schedules and observed that they are much better off with the New schedule.

References

[1] M.W. Carter, A survey of practical applications of examination timetabling, Operations Research, **34,** no. 2, (1986), 193–202.

[2] S. Kirkpatrick, C.D. Gelatt, Jr., M.P. Vecchi, Optimization by Simulated Annealing, Science, **220,** no. 4598, (1983), 671–680.

[3] P. Ross, D. Corne, Comparing Genetic Algorithms, Simulated Annealing, and Stochastic Hillclimbing on Timetabling Problems, Proceeding of A/SB Workshop on Evolutionary Computing, (1995), 94–102.

[4] L. Davis, Genetic, Algorithm and Simulated Annealing, Research Notes in Artificial Intelligence, Morgan Kaufmann Publishers, 1993.

[5] D.E. Goldberg, Genetic Algorithms in search, optimization and Machine Leaning, Addison Wesley Publishing Company, Inc., 1989.

[6] E. Burke, D. Ellim, R. Weare, A Genetic Algorithm Based University Timetabling System, Proceedings of the 2nd East-West International Conference on Computer Technologies in Education, **I,** (1994), 35–40.