A Simple Heuristically Guided Search for the Timetable Problem

E.K. Burke, J.P. Newall, R.F. Weare
{ekb|jpn}@cs.nott.ac.uk

Automated Scheduling and Planning Group
Department of Computer Science
University of Nottingham
Nottingham NG7 2RD
UK

ABSTRACT

Domain specific heuristics have been utilised in solutions to the timetable ever since the first attempts at automation were made[foxley68, cole64]. While methods incorporating heuristics generally produce quite acceptable results in a very small CPU time they lack the optimisation capabilities of the more CPU intensive search methods. In this paper we will outline a simple efficient technique based on heuristics that includes a variable degree of nondeterminism to facilitate a search through the solution space. Careful use of heuristics should mean that this simple search technique will target more desirable areas of the search space. This we hope will provide a compromise between deterministic methods and full blown search methods, and also determine the benefits from using heuristics in search methods in general. In this paper we will show by experimentation that such a heuristically guided search can often outperform a pure heuristic approach (with the added benefit of backtracking) when using the same heuristic. We will also speculate on the possible advantages and disadvantages of utilising heuristics in more advanced search strategies such as Genetic Algorithms.

1 Introduction

1.1 The Examination Timetable Problem

While heuristics can be applied to timetable problems in general, the experiments presented in this paper will deal with examination timetabling problems only. At its most basic this problem has two constraints that must be satisfied by any feasible timetable:

- For any two exams scheduled in the same period there should not be any students enrolled for both exams.
- The seating capacity for any period should not be exceeded.

An institution may also impose other temporal constraints between exams, for example:

- If two conflicting exams are scheduled on the same day, then there should be a full period between them.
- Minimise occurrences of two conflicting exams being scheduled on the same day.
- One exam may need to be scheduled before another, related, exam.
- Two exams may need to be scheduled at the same time, as they contain similar material.

A number of unary constraints may also be imposed, such as:

- An exam may need to be scheduled in a particular period, or within a limited set of periods.
- An exam may require a particular room.

With most exam timetabling problems it is the spread constraints that present the greatest difficulty as the other constraints usually arise much less often. This is a combinatorial optimisation problem, often without any perfect solution.

1.2 Heuristic Methods

One of the most common approaches that use heuristics is the sequential method, where we apply a sequencing strategy to the problem. Approaches of this kind have been used for timetabling problems since the early sixties[foxley68, cole64] as well as more recently[carter94], and can produce quite acceptable results with minimal equipment requirements. The method involves using a heuristic to estimate how difficult an exam will be to schedule then we can order the events by decreasing difficulty. Scheduling in this order means that a considerably better timetable should be found than if the events were scheduled in a random order. When scheduling the events there may be a choice of periods in which to schedule, in which case another strategy for period selection is required. Example strategies could be ‘placing in the first available period’, ‘placing in the period with lowest cost’ or maybe ‘placing in the period with the most similar exams (in terms of shared neighbours) already scheduled’.

Another method that often utilises heuristics is the *cluster* approach[white79]. This involves grouping the exams into clusters, as the name suggests, where any

exams within a cluster do not conflict with any other exams within the same cluster. These clusters correspond to periods, but assigning which cluster to which period presents another problem. Usually the assignment of clusters to periods is done so as to minimise the number of adjacent conflicts. This method does have considerable limitations, namely that once clusters are formed it may not be possible to construct a good quality timetable with those particular clusters, and will not be considered further here.

2 A Heuristically Guided Random Search

While the sequential approaches mentioned above can produce acceptable results quickly we may be able to improve on this by adding an element of nondeterminism to the process. This will facilitate iteration of the process, resulting in a simple search strategy which might compare well with other more elaborate search strategies.

2.1 The Heuristic Backtracking Method

The heuristic backtracking method used here is similar to the approach used by Carter et Al[carter94] without the max-clique initial stage. Firstly we must order all the events by our sequencing strategy, which will be dictated by the following heuristics:

- *Largest Degree First*. We will schedule those events with the greatest number of conflicting events first. We would expect these events to be more difficult to schedule.
- *Largest Colour Degree First*. In this strategy we schedule first those events with the greatest number of conflicting events that have already been scheduled. Generally these would be more difficult to place in the timetable than those with little or no conflicting events already in the timetable. Largest Degree first is used to break ties.
- Least Saturation Degree First. This approach was proposed by Brelaz[brelaz79]. We schedule first those events with the least number of valid periods currently available. This will give priority to those events that have very few periods available, perhaps because a large number of their conflicting events have already been scheduled, or maybe because the number of suitable periods was limited to begin with. In either case such events may prove difficult or impossible to schedule later on in the process. Largest degree first is again used to break ties.

Having decided on our sequencing strategy we take each event in turn and schedule it in the valid period where that event will cause least penalty (in terms of our objective function). If there is no valid period in which to schedule an event we must consider unscheduling some of the events already scheduled in order to make room for it. In this case each period of the timetable is examined and will fall into one of the following three cases:

1. The current event could not be scheduled in the period even if there were no conflicting events already in that period. Therefore the period is no longer considered.
2. The current event could be scheduled into the period if a number of already existing events were to be removed from the period. If this is the case the cost of scheduling in this period is the number of events that would have to be bumped from the timetable.
3. The current event could be scheduled into the period if a number of already existing events were removed. However one of these events has already been bumped by the current event and will not be permitted to do so a second time. Therefore this period is no longer considered. This is to avoid the possibility of looping.

From the set of periods that fall into case 2 we select the one which involves bumping the least number of events. If no periods at all fall into case 2 then the event must be left unscheduled. On scheduling in this period the existing events that were removed from the period are placed at the head of our sequence for rescheduling.

2.2 Heuristic Random Search

While a sequencing method such as the one above provides a quite acceptable solution, at least with a good choice of heuristic, it suffers in that it only produces a single solution which may not be as good as it could be. One exception to this is a technique proposed by Broder[broder64] which involved embedding a random element into the process to break ties. This meant that the entire process could be iterated a number of times with the best solution being selected. Another approach is that proposed by Corne and Ross[corne96] which used ‘peckish’ assignment (similar to the tournament method used here) to place events (presumably in a random or ‘as is’ order) in periods when generating the initial population. This functioned by choosing the best period for each event from a subset of periods rather than the entire set. While this process is capable of being iterated as a search process in itself the lack of any meaningful ordering of the events would probably mean that it would produce far from optimal results.

By similarly introducing a random element into the

pure heuristic method we may be able to improve on the results found. This approach will differ from that method in that firstly we will not implement a backtracking procedure, as this will not be as vital as when we generate a single solution and will add considerable extra time requirement in an iterated method. Secondly instead of selecting the best event according to our heuristic we will use one of the following methods to select the next event.

- *Tournament Selection*. For this we generate a random subset of the events not yet scheduled and then choose the best event according to our heuristic from within this subset. For these experiments we shall be using tournament sizes of 5%, 10% and 20% of the full set of events.
- *Bias Selection*. Here we order our events according to our sequencing strategy but rather than picking the best event every time we select an event at random from the best n. Here we will be selecting from either the best 2, best 5 or best 10 events.

As before we then schedule the chosen event in the valid period causing least penalty, or if no valid period exists we simply leave it unscheduled. If there is a tie between periods we choose the earlier of the periods.

We can then iterate this entire procedure a number of times, keeping track of the best solution found so far.

2.3 Random Search

While we want to show that the proposed hybrid method can outperform heuristic methods it is also necessary for us to show that it can outperform random search. For the random search we simply take the exams in some randomly determined order then schedule each in the period with least penalty, as we did for the heuristic methods. Again if there is a tie between periods we schedule in the earlier of the periods. This can then be iterated like the hybrid method, with a copy of best solution found so far kept.

3 Results

In order to compare the three methods presented here a number of real test problems were used. All of this data can be obtained over the internet from

`ftp://ie.utoronto.ca/mwc/testprob`

Table 1 shows the codes used here for the relevant institutions' data, and Table 2 shows some of the characteristics of the data sets.

Table 1: Data sets used with codes

Data	Institution
carf92	Carleton University
kfu	King Fahd University
utas	University of Toronto, Arts & Sciences

This data presents a reasonable covering of average density problems, and each can be considered challenging merely to find a feasible solution. For all of the above problems it is unlikely that any perfect solutions exists, though this assumption can only be disproved by actually finding such a solution.

For all the methods experimented with the same objective quality function was used of course, which can be represented by the function operating on some timetable t,

$$Penalty(t) = 5000.unscheduled(t) + 3.sameDay(t) + overNight(t)$$

where *unscheduled* is the number of exams not scheduled in a valid period (which we weight by 5000 to discourage incomplete timetables), *sameDay* is the number of adjacent student conflicts between periods on the same day (weighted here by 3 to express an importance over conflicts occurring over night) and *overNight* is the number of logically adjacent student conflicts that occur with at least a night in between (for instance conflicts between the last period on Monday and the first period on Tuesday).

For the purpose of establishing adjacencies the timetable structure was to have three adjacent periods per day from Monday to Friday followed by a single period on Saturday. This continues for as many weeks, or partial weeks, to obtain the necessary number of periods for the problem.

Table 2: Characteristics of Data Sets

Data	No. Exams	No. Students	No. Enrolments	Seats per Period	Number of Periods	Density of Conflict Matrix
carf92	543	18,419	54,062	2,000	36	0.14
kfu	486	5,349	25,118	1,955	20	0.06
utas	622	21,267	58,981	2,800	37	0.12

3.1 The Heuristic Methods

The proposed algorithm was run on a number of real problems along with the heuristic backtracking algorithm to see how they compare. In the case of the heuristic random search each instance was run ten times with distinct random seeds. The graphs shown in figures 1 to 3 show the results found by the heuristically guided random search, taken as the average penalty over the test cases, using each of our heuristics. This will give us an idea of average performance over time. The horizontal line present in each graph represents the result found by the backtracking method when using the same heuristic. The individual results for each case are given in table 3, where the 100% tournament size represents the pure heuristic sequencing approach with backtracking.

Figure 1: Using Largest Degree

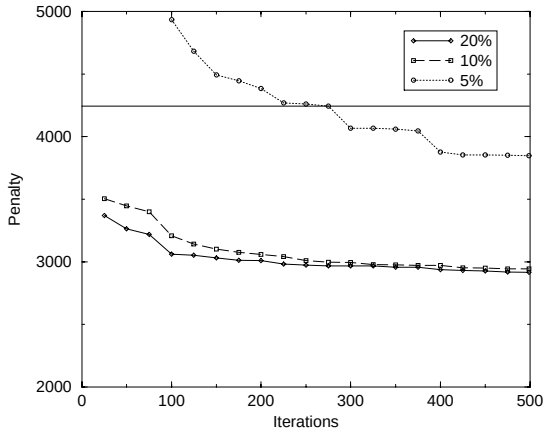


Figure 2: Using Colour Degree

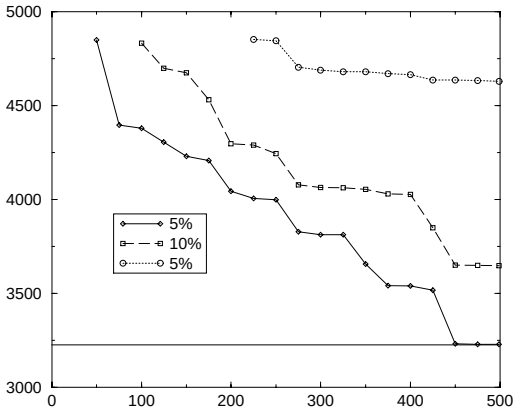
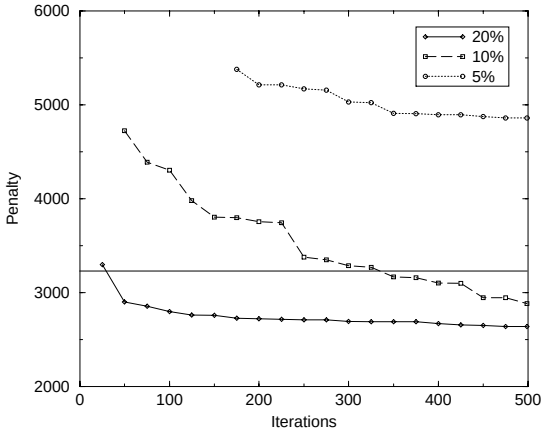


Figure 3: Using Saturation Degree



The results show that it is possible to exceed the heuristic backtracking solution using the heuristically guided random search within 500 iterations. As a guide to execution time 10 iterations of the guided search will take roughly as long a heuristic backtracking run using the same heuristic. Also it should be noted that larger tournament sizes involve more computation and will therefore take longer, with an iteration using a 20% tournament size taking approximately twice as long as each iteration when using a 5% tournament size.

The most notable of these sets of results is those utilising largest degree first. Using this within our search seems to produce considerably better results than the heuristic sequencing method using largest degree. On the other hand when using colour degree the result from sequencing is only just matched when using a 20% tournament size. Similarly when using Saturation degree the results are only marginally improved upon when using 10% and 20% tournament sizes. One possible reason for this is that colour and saturation degree produce much better solutions when used in heuristic sequencing, having the effect that there may be very few other solutions that are actually better. In a sense the more appropriate a heuristic is for a particular problem then maybe there is less need for a random element in the process to compensate for the mistakes that might be made by the heuristic.

Table 3: Individual results after 500 iterations

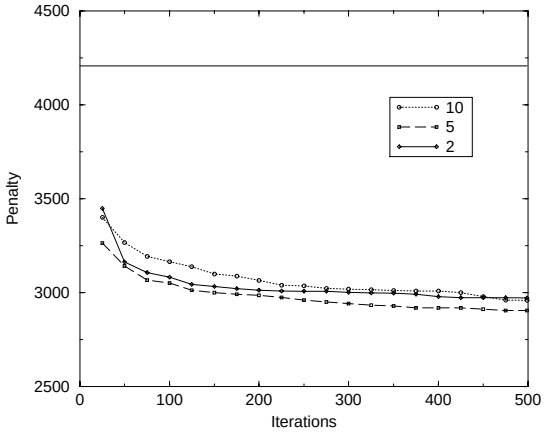
Heuristic	T-Size(%)	Penalty		
		kfu	carf92	utas
LD	5	5950	2738	2855
	10	3589	2555	2736
	20	3422	2588	2738
	100	4638	3209	4598
CD	5	8214	2773	2902
	10	5543	2604	2797
	20	4307	2583	2794
	100	4016	3074	2542
SD	5	9097	2606	2878
	10	3816	2305	2529
	20	3256	2218	2440
	100	3874	2915	2571

Looking at the results in terms of tournament size we see that in all the cases a 5% tournament size produces substantially worse timetables than the larger tournament sizes using the same heuristic as we would expect. However when we compare tournament sizes of 10% and 20% there appears to be only marginal differences in quality, though when using saturation degree better results were found much faster when using a 20% size. These observations would seem to suggest that there is probably an optimal tournament size of around 20%.

Turning our attention to the individual results we see that colour degree with a 20% tournament size only manages to outperform the pure method on the carf92 data set, bringing the average penalty down to that of the pure approach. Largest degree on the other hand manages substantial improvements for 10% and 20% tournament sizes, though the difference between these two themselves is negligible. On the whole though it is saturation degree with a 20% tournament size which gives us the best result for all of the test cases.

Having shown that the method using tournament selection can outperform the backtracking method it remains to be seen if this might be improved on further by using bias selection, described above. Here we use bias selection sizes of 2, 5 and 10, the results are shown in figure 4.

Figure 4: Largest degree using bias selection



In terms of final solution bias selection produces results comparable to tournament selection with 10% and 20% sizes. An interesting note though is that there does not appear to be any degradation in the results for larger bias sizes, which there was correspondingly when using smaller tournament sizes. This is probably due mostly to the fact that we always choose one of the best scoring events (according to our heuristic), whereas with a 5% tournament size it is possible that we have a tournament consisting of some of the lowest scoring events. The bias approach also has the advantage of requiring less computation, at least for largest degree, which only requires a single initial ordering. Random Search

A multi-start random search was run for 10,000 iterations on each of the problems to see how this would compare with the heuristic methods. Despite being executed for substantially longer than the other two methods the search, as we might expect, did not come anywhere close to the previous result reaching a minimum penalty of 82945. It is most likely that the high penalty here is largely due to the search mechanism's inability to find complete timetables for the densely conflicting problems used here, rendering it almost useless on any difficult problem.

4 Conclusions

While the results found by combining nondeterminism with heuristics could be criticised as being only relatively small improvements (a reduction of up to 25% in penalty) on those found using a heuristic sequencing method with backtracking, it should be noted that these solutions are very sparsely spread throughout what is, after all, a very large search space. To actually find these improvements in a still relatively short period of time is a considerable achievement. As it stands, this method shows itself to be very competent on the timetabling

problem, and forms a good compromise approach between sequencing methods and the more time consuming optimisation methods such as Tabu search,[hertz91], Simulated annealing[thompson96] and Genetic algorithms[davis91, corne94]. Perhaps a useful conclusion to draw would be based on how the solution quality improves much more over backtracking when used with largest degree. As largest degree would seem to be less appropriate when used with backtracking than colour degree or saturation degree were, it is possible that an element of non-determinism can help improve solution quality when a heuristic is almost, but not entirely, suitable for a problem. This could perhaps improve solution quality when small numbers of side constraints are added to problems such as the ones outlined here, which maybe are not applicable to the chosen heuristic.

We will compare the results of the method presented here to with those of one such optimisation procedure, namely the timetabling memetic algorithm presented by Burke et Al[burke96a]. From this we can see that we can produce tangibly better results using this method than we could using the more time consuming memetic algorithm, even though here we are also optimising the overnight conflicts. While doing this though, we should take into account that the more densely conflicting problems used here were not a strong point of the memetic approach. It should also be noted that the memetic algorithm does not rely on a suitable choice of heuristic, something that all heuristic methods will suffer from.

While this method shows worth in itself, the main observation to make from these results is how heuristic approaches can be improved with the addition of simple nondeterminism. It follows from this that maybe combining heuristics with a more advanced search strategy could probably yield even better results, though this would of course rely on the suitability of the heuristic used for each individual problem.

As future work the authors will be experimenting with methods of utilising heuristics with the memetic algorithm shown in [burke96a]. A logical application would be to use the method in this paper to produce the initial population for the algorithm. This would be similar to the approach used by Corne and Ross[corne96] but the use of proven heuristics on which to base the ordering of the events should lead to better quality initial population and hopefully a better final solution. This should give the algorithm a head start while maintaining a diverse population on which to optimise further.

References