Comparing Genetic Algorithms, Simulated Annealing, and Stochastic Hillclimbing on Timetabling Problems

Peter Ross, Dave Corne

Department of Artificial Intelligence, University of Edinburgh, 80 South Bridge, Edinburgh EH1 1HN, U.K.

Abstract. Much recent research has focussed on applying genetic algorithms (GAs) to real educational institution timetabling problems. This work is generally successful, but it is as yet unclear whether a simpler stochastic hillclimbing (SH) strategy would generally do just as well, and how both GA and SH might compare with the use of simulated annealing (SA) on timetabling problems. We begin to investigate these concerns by comparing GA, SH, and SA on a collection of real timetabling problems. Comparisons are done in terms of final solution quality, and number of distinct solutions obtained. When considering the latter criterion, we necessarily compare the GA with modified SH and SA algorithms which continually restart to look for new solutions. The main conclusions are that SH and SA are generally the best strategy as far as solution quality is concerned. For a certain fairly small range of problems though, the GA either betters or equals the performance of SA and SH, but delivers the added value of a large number of usefully distinct, equally good solutions. Finally, we note that our results are to be taken in the context of particular implementations of SA, SH, and GA; although steps are taken to optimise parameters and such for each implementation, different conclusions may have been reached if, in particular, we had used more sophisticated SA cooling schedules, and/or more sophisticated GA operators. Such complexities concerning GA/SA comparisons in general are discussed.

1 Introduction

Research on the application of Genetic Algorithms (GAs) to timetable optimisation problems is gathering pace. Since early papers on the subject presented some initial ideas [1, 4, 5], later researchers have pursued a number of different approaches on individual examples of timetabling problems [7, 2, 10]. The central problem is illustrated well by the exam timetabling case. A number of events (usually examinations), must be assigned timeslots subject usually to the following constraints:

1. There are a finite, usually small, set of timeslots. Eg, 3 per day over 10 days.
2. Obviously, two exams must not be set at the same time if one or more students need to sit both of them.

3. It would be nice to avoid cases of students having to sit exams in consecutive timeslots.

Constraint 3 above, and similar constraints, make this an extremely hard combinatorial optimisation problem. The problem is essentially to satisfy constraints 1 and 2, while optimising over violations of constraint 3 and similar. Often, for example, room assignments need to be made for each event, with associated room capacity constraints, further complicating the problem.

There seem to be currently three main approaches to using GAs in this area. The differences lie in the representation strategy and operators used. Paechter et al [8] use an indirect encoding of a timetable, containing essentially a list of instructions for a timetable building procedure which makes sure to satisfy constraints as it goes, somewhat like a greedy algorithm, typically ending with a collection of events which cannot be thus placed in the timetable without violating a constraint; Paechter's method is to minimise the number of such unplaced events. Burke et al [2], on the other hand, use a combination of graph colouring techniques with GA search. All timetables are feasible, satisfying all constraints *except* that extra timeslots are used when necessary. That is, instead of 'unplaced' events, Burke et al place events into extra timeslots. Their strategy is to apply operators to such timetables which preserve satisfaction of all the constraints, with the objective of minimising the number of extra timeslots used. Finally, Ross et al's method is to allow constraint violations in the timetable, but penalise such violations according to a simple penalty function strategy; this approach is then backed up by powerful intelligent local mutation operators which then aid in the process of minimising the extent of the constraint violations. Also, research is going ahead on the use of Simulated Annealing (SA) for real timetabling problems. Thompson & Dowsland [11], for example, use the same representation as Ross et al but use SA instead of GA to solve real timetabling problems in the University of Wales at Swansea.

What is conspicuously rare in this line of research so far is a comparative study of different approaches. We perform such a study here, comparing the performance of Ross et al's GA-based approach briefly discussed above, with Simulated Annealing, and Stochastic Hillclimbing (SH), on five variants of a real, challenging timetable optimisation problem.

In section 2 we give details of the test problems, review the representation strategy and neighbourhood operator common to each algorithm, and then give details of each algorithm. Section 3 briefly discusses the performance measures we use in this study. Section 4 then outlines the general experimental approach, and presents the raw results of a large number of experiments on each of the test problems. Finally, section 5 offers some interpretation of the results, and a concluding discussion.

2 Problems and Algorithms

Each of stochastic hillclimbing (SH), simulated annealing (SA), and a simple genetic algorithm (GA) is tested here on a range of timetabling problems. The