A Hybrid Grouping Genetic Algorithm for Examination Timetabling

Wilhelm Erben, Peng Yu Song

Department of Computer Science, FH Konstanz - University of Applied Sciences,
78462 Konstanz, Germany
erben@fh-konstanz.de

1 Introduction

In this paper, we present an improvement of the Grouping Genetic Algorithm (GGA) which has been described in [6]. The algorithm has been enhanced with local optimisation techniques which are based on problem-specific heuristics and incorporated into the genetic operators. First experiments on well-known benchmark problems on examination timetabling show that the new hybrid algorithm (HGGA) produces much better results than the original GGA.

1.1 Examination Timetabling, Graph Colouring and Grouping Problems

In the examination timetabling problems considered in this paper, given exams have to be assigned to a number of *periods* in such a way that the *hard constraints*

(1) No student may have two exams in the same period
(2) The sum of students taking an exam in the same period must not exceed a certain capacity limit.

are satisfied. In addition, we try to avoid *near-clashes of order n*, if possible:

- No student should have to sit two exams which are only n periods apart (n=1,2,3,4).

The basic examination problem in which only the clashing constraint (1) is to be satisfied is easily mapped into a graph colouring problem: The *exams* are represented as *nodes* of a graph, and an edge between two nodes corresponds to constraint (1). The nodes have to be coloured in such a way that two nodes connected by an edge receive different *colours*, i.e. the corresponding exams are scheduled to different *periods*.

Graph colouring is an instance of the family of *grouping problems* ([7]) which consist of partitioning, due to some hard constraints, a set of entities into mutually

disjoint subsets, so-called *groups*. In graph colouring (in examination timetabling) the nodes (exams) are assigned to those *groups (colours; periods)* in such a way that no group contains two adjacent nodes (two conflicting exams).

1.2 The Grouping Genetic Algorithm

In our *Grouping Genetic Algorithm* (GGA) [6], which has essentially been designed as proposed by Falkenauer [7], a chromosome is made up of *groups* as genes. It could, for instance, be represented as

$$(\{1,2,5\}, \{3,7\}, \{4\}, \{6,8\}),$$

indicating that exams 1, 2 and 5 are scheduled in the first period, exams 3 and 7 in the second, etc.

The following *genetic operators*, working on these groups, have been defined:

For *crossover*, some randomly chosen groups from one parent are injected into the other chromosome. As a result, some groups are not disjoint any more. Hence the affected groups of the second parent are deleted, and those exams that afterwards do not belong to a group any more are rescheduled using a first fit algorithm.

There are two *mutation operators:* The first one simply eliminates one or a few groups, chosen at random, and tries to reschedule their exams, applying again the first fit algorithm. In the second mutation operator the positions of two randomly chosen groups are exchanged.

A fitness function is used which is based on the heuristic that as many examinations with large numbers of conflicts as possible should be assigned to each period. It also penalizes near-clashes, using large weights for near-clashes of low order.

2 The Hybrid Grouping Genetic Algorithm

The GGA has been enhanced with two 'sequential' heuristics, the first one mainly aiming at improving the search for optimal solutions to mere graph colouring problems, the other one being used in examination timetabling for the reduction of the number of near-clashes of order n.

Both heuristics are incorporated into the crossover and mutation operators, and will be described in the following sections.

2.1 Mutation and Crossover with Replacement

The basic GGA had already shown very good results on a wide range of randomly generated hard-to-colour graphs. It was even excellent when applied to real-life exam timetabling problems, as long as near-clashes were disregarded.

Nevertheless, we felt that we still could improve its performance by adding new heuristics. Inspired by the so-called *bin packing crossover with replacement* which was introduced by Falkenauer [7] in his design of a grouping genetic algorithm for

bin backing problems, we enhanced the mutation and crossover operators described in section 1.2 in the following way:

Prior to the application of the first fit algorithm which is used to reinsert exams of eliminated groups, we first check, taking the groups one by one, whether some of the (scheduled) exams of the group can be *replaced* by unscheduled ones of higher degree (i.e. with a larger number of conflicts), without violating the hard constraints.

As a consequence, some of the previously scheduled exams are unscheduled after this replacement, but - as a countermove to that - a number of more difficult-to-assign exams are now successfully scheduled. Some of the groups contain more examinations with larger numbers of conflicts than before, and it will be easier now to reinsert the remaining unscheduled exams by use of the first fit algorithm.

2.2 Separating Mutation

The above described procedure further guides the search for well-filled timeslots and, as a consequence, for a timetable using only a low number of periods. When applied to real life problems, however, this seems to be counterproductive: The closer examinations with large numbers of conflicts are scheduled together, the more near-clashes are likely to occur.

In order to reduce the number of near-clashes we introduced a further mutation operator as follows: Two groups, G and H say, are chosen at random. Again, the exams of group G are rescheduled, but prior to trying to insert them into existing groups a new group G' is created and placed between period H and its adjacent period H'. An exam e of group G is inserted into G' if no student enrolled in e has to sit any exam in period H or H'. Otherwise e is rescheduled using the first fit algorithm, as before.

In this way, groups H and H' are separated by group G', and if there was a near-clash of order 1 before – caused by a student who has to sit exams in the previously successive periods H and H' - this is removed by application of the new operator, which we call *separating mutation of order 1*. In a similar way separating mutation of higher order can be defined.

2.2 Results and Conclusion

We have performed first experiments on Carter's data set [4]. They clearly show that the hybrid version is superior to the original GGA. But the HGGA still does not compare well enough to some other published methods (cf. [1], [2], [3], [5], [8]), and we will have to perform further tests before we shall be able to present significant results.

We still have to check, in particular, which combination of the new operators will be best. While mutation with replacement should be frequently used during the first iteration steps of the algorithm, an increasing number of separating mutations should be performed later in the process. The idea is that first a timetable using as few periods as possible is produced, regardless of the number of near-clashes. In the second phase, however, this 'compact' timetable should be extended to more periods in order to allow students to rest for a while before they have to sit their next exam. Therefore best settings of the parameters which control the interaction of the operators have to

be found, and also the fitness function should be dynamically changed in order to support this basic idea.

References

1. Burke, E.; Newall, J.: Enhancing Timetable Solutions with Local Search Methods. In: Burke, E.; De Causmaecker, P. (eds.): Practice and Theory of Automated Timetabling IV (PATAT 2002, Gent, Belgium, August, selected revised papers). Lecture Notes in Computer Science, Vol. 2740. Springer-Verlag, Berlin Heidelberg New York (2003) 195-206
2. Burke, E.; Newall, J.: A Multi-Stage Evolutionary Algorithm for the Timetable Problem. The IEEE Transactions on Evolutionary Computation, Vol 3.1 (1999) 63-74
3. Carter, M.; Laporte, G.; Lee, S.Y.: Examination Timetabling: Algorithmic Strategies and Applications. Journal of the Operational Research Society 47(3) (1996) 373-383
4. Carter, M.: ftp://ftp.mie.utoronto.ca/pub/carter/testprob
5. Di Gaspero, L.; Schaerf, A.: Tabu Search Techniques for Examination Timetabling. In: Burke, E.; Erben, W. (eds.): Practice and Theory of Automated Timetabling III (PATAT 2000, Konstanz, Germany, August, selected papers). Lecture Notes in Computer Science, Vol. 2079. Springer-Verlag, Berlin Heidelberg New York (2001) 104-117
6. Erben, W.: A Grouping Genetic Algorithm for Graph Colouring and Examination Timetabling. In: Burke, E.; Erben, W. (eds.): Practice and Theory of Automated Timetabling III (PATAT 2000, Konstanz, Germany, August, selected papers). Lecture Notes in Computer Science, Vol. 2079. Springer-Verlag, Berlin Heidelberg New York (2001) 132-156
7. Falkenauer, E.: Genetic Algorithms and Grouping Problems. John Wiley & Sons, Chichester (1998)
8. Merlot, L.; Bowland, N.; Hughes, B.; Stuckey, P.: A Hybrid Algorithm for the Examination Timetabling Problem. In: Burke, E.; De Causmaecker, P. (eds.): Practice and Theory of Automated Timetabling IV (PATAT 2002, Gent, Belgium, August, selected revised papers). Lecture Notes in Computer Science, Vol. 2740. Springer-Verlag, Berlin Heidelberg New York (2003) 207-231