A Multi-start Very Large Neighbourhood Search Approach with Local Search Methods for Examination Timetabling

Salwani Abdullah and Edmund K. Burke

Automated Scheduling, Optimisation and Planning Research Group, School of Computer Science & Information Technology, University of Nottingham, Jubilee Campus, Wollaton Road, Nottingham NG8 1BB, United Kingdom
{sqa, ekb@cs.nott.ac.uk}

Abstract

This paper investigates a hybridisation of the very large neighbourhood search approach with local search methods to address examination timetabling problems. In this paper, we describe a 2 phase approach. The first phase employs "multi start" to carry out search upon a very large neighbourhood of solutions using graph theoretical algorithms implemented on an improvement graph. The second phase makes further improvements by utilising a local search method. We present experimental results which show that this combined approach compares favourably with other algorithms on the standard benchmark problems.

Introduction: Formalising the Examination Timetabling Problem

Examination timetabling is concerned with an assignment of exams into a limited number of timeslots so that no student is assigned two or more exams at the same time. Overviews of the scientific literature on university timetabling can be seen in, for example, [9,13,22]. The objective, in the problems we tackle, is to try to spread the exams as evenly as possible throughout the schedule. We can represent the examination timetabling problem using a formal model which was adapted from the version presented in [11]. We are given the input for the examination timetabling problem as follows:

- N is the number of exams
- P is the given number of available timeslots
- M is the number of students
- $C = (c_{ij})_{NxN}$ is the conflict matrix where each element, denoted by $c_{ij}(i,j \in \{1,...,N\})$, represents the number of students taking exams i and j.
- t_k $(1 \le t_k \le P)$ specifies the assigned timeslot for exam k $(k \in \{1,...,N\})$

The objective is to minimise the proximity cost which can be formulated as a minimisation of Expression (1):

$$\frac{\sum\limits_{i=1}^{N-1} F(i)}{M} \qquad (1)$$

where

$$F(i) = \sum_{j=i+1}^{N} c_{ij} \cdot proximity\ (t_i, t_j) \qquad (2)$$

and

$$proximity\ (t_i, t_j) = \begin{cases} 2^{5}/2^{|t_i - t_j|} & if\ \ 1 \le |t_i - t_j| \le 5 \\ 0 & otherwise \end{cases} \qquad (3)$$

subject to

$$\sum_{i=1}^{N-1} \sum_{j=i+1}^{N} c_{ij} \cdot \lambda(t_i, t_j) = 0$$

where

$$\lambda(t_i, t_j) = \begin{cases} 1 & if\ \ t_i = t_j \\ 0 & otherwise \end{cases} \qquad (4)$$

Equation (2) represents a cost value for an exam i which is given by the proximity value multiplied by the number of students in conflict. Equation (3) represents that proximity value for two exams with common students [14]. Equation (4) represents the requirement that no student is asked to sit two exams at the same time.

A "Multi-start Two Phase Approach"

The very large neighbourhood search method has produced the best known results on some of the standard international benchmark problems for examination timetabling [1]. Many of the other successful approaches in the literature represent hybridisations that often involve local search. See, for example, [10,21]. These observations led us to explore the hybridisation of the very large

neighbourhood search approach with local search methods and we briefly present the results of this hybridisation in this paper. We propose a multi-start very large neighbourhood search in the first phase. See [3,4,5] for the details of the large neighbourhood search approach and its application to various optimisation problems. The approach in the first phase is restarted several times with a new initial solution in order to diversify the search [19].

```
Set initial solution, Sol, by employing
saturation degree heuristic by [8];
Calculate initial cost function f(Sol);
Set best solution, Sol_best ← Sol;
Define not_improving_length;
Set number of iteration, NumOfIte;
Set not_improving_counter ← 0;
Set iteration = 0;
do while (iteration < NumOfIte)
    Phase 1:
    Execute large neighbourhood search to obtain
    new solution [1];
    Evaluate new solution f(Sol*);
    if (f(Sol*) ≤ f(Sol_best))
        Sol ← Sol*;
        Sol_best ← Sol*;
        Set not_improving_counter ← 0;
    else
        Execute exponential monte carlo as in
        [7]:
        δ = f(Sol*)- f(Sol);
        Generate RandNum, a random number in
        [0,1];
        if (RandNum < e^−δ )
            Sol ← Sol*;
            Set not_improving_counter ← 0;
        else
            Increase not_improving_counter by
            1;
            if (not_improving_counter ==
            not_improving_length)
                Sol ← Sol_best;
    iteration++;
end do

 Phase 2: Run great deluge and simulated
annealing algorithms, each seeded with Sol_best as
an initial solution;
```

Figure 1. The pseudo code for our multi-start algorithm

We can briefly define our model for the first phase approach in a few steps. Firstly, we treat the examination timetabling problem as a variant of a partitioning problem. Secondly, we define our neighbourhood structure through a cyclic exchange operation that yields a very large neighbourhood structure. Next, we proceed to the important part of the large neighbourhood search algorithm which is the construction of the improvement graph. Lastly, we determine the improving moves in the improvement graph. These moves are found heuristically by employing a modified shortest path label-correcting algorithm for the improvement graph which was adapted from a method presented in [2]. The basic idea of this

modified shortest path label-correcting algorithm is to find a shortest distance from one exam (as a source exam) to other exams in the improvement graph. The details for the first phase approach can be seen in [1]. The algorithm always accepts a better solution and a worse solution will be accepted with a probability generated from the exponential monte carlo acceptance criteria which is only based on the solution quality [7]. We continue the process until the end of the *do-while* loop (see Figure 1).

In the second phase, we employ a local search method to make a further improvement to the best solution obtained from the first phase. We consider the great deluge [18] and simulated annealing algorithms [20]. We note that the great deluge approach has produced very good results for examination timetabling [10,11]. It is related to simulated annealing. The pseudo code for our implementation of the great deluge algorithm is presented in Figure 2.

```
Set initial solution as Sol_best taken from
large neighbourhood search approach, Sol;
Calculate the initial cost function, f(Sol);
Set best solution, Sol_best ← Sol;
Set estimated quality of final solution,
estimatedquality;
Set number of iteration, NumOfIte;
Set initial level: level ← f(Sol);
Set decreasing rate β;
Set iteration ← 0;
Set not_improving_counter ← 0;
do while (iteration < NumOfIte)
    Select an exam at random and assign to a
    random feasible timeslot;
    Evaluate new solution, f(Sol*);
    if (f(Sol*) < f(Sol))
        Sol ← Sol*;
        Sol_best ← Sol*;
        not_improving_counter ← 0;
    else
        if (f(Sol*)≤ level)
            Sol ← Sol*;
        else
            Increase not_improving_counter
            by 1;
            if (not_improving_counter ==
            not_improving_ length_GDA)
                exit;
    level = level - β;
    iteration++;
end do;
```

Figure 2. The pseudo code for the great deluge algorithm

We define a number of iterations, *NumOfIte* and an estimated quality of final solution *f(estimatedquality)*. The quality value of a solution, *Sol*, is represented by *f(Sol)*. A *decreasing rate*, *β* is calculated using the following formula (adapted from [11]):

$$\beta = (f(Sol) - f(estimatedquality)) / (NumOfIte)$$

The *level* is equal to the quality value of the initial solution, *f(Sol)* at the start and will decrease by the value *β*. In the *do-while* loop, a neighbour is defined by randomly

selecting an exam and assigning it to a valid timeslot. The cost function resulting from the new neighbour (move) is calculated using the formula as defined in Expression (1). A worse solution is accepted if the quality of the new solution, *f(Sol*)* is less than the *level*. The new solution is updated and the process continues until the number of iterations is greater than *NumOfIte* or if there is no improvement for a certain number of iterations, referred to as "*not_ improving_ length_ GDA*" in the pseudo code.

The simulated annealing algorithm applied in this paper is presented in Figure 3.

```
Set initial solution as Sol_best taken from
large neighbourhood search approach (Figure
1), Sol;
Calculate the initial cost function, f(Sol);
Set best solution, Sol_best ← Sol;
Set number of iteration, NumOfIte;
Set initial temperature T_0;
Set final temperature T_f;
Set decreasing temperature rate as α where α
= (log(T_0) - log(T_f))/NumOfIte;
Set T ← T_0;
do while (T > T_f)
     Define neighbour of Sol by randomly
     assigning exam e_i, i ∈ {1,…,N} to a
     valid timeslot t_n, n ∈ {1,..,P} to
     generate new solution called Sol*;
     Calculate f(Sol*);
     if (f(Sol*) < f(Sol))
          Sol ← Sol*;
          Sol_best ← Sol*;
     else
          Generate a randum number called
          RandomNumber;
          if (RandomNumber ≤ e^{-δ/T})
               Sol ← Sol*;
     T = T-T*α;
end do;
```

Figure 3. The pseudo code for the simulated annealing algorithm

We use the same parameters as in [11] where the initial temperature T_0 is equal to 5000 and the final temperature T_f is equal to 0.05. The number of iterations, *NumOfIte* is set to be 10000000. At every iteration, T is decreased by α where α is defined as:

$$\alpha = (\log (T) - \log (0.05)) / NumOfIte$$

In the do-while loop, a neighbour is defined by randomly selecting an exam and assigning it to a valid timeslot. A worse candidate solution is accepted if the randomly generated number, *RandomNumber*, is less then $e^{-\delta/T}$. The process continues until the temperature T is less than the final temperature T_f.

Experiments and Results

Our program was coded in Visual C++ and the experiments were run on a PC with an Athlon 1.2 GHz processor and 256 MB RAM and Windows 2000. Our algorithm was evaluated on the public data sets made available by [14]. We ran the experiments for 500000 and 10000000 iterations using the very large neighbourhood search approach and the great deluge (and simulated annealing) algorithm respectively which takes approximately five hours. Experimental results show that the great deluge outperforms the simulated annealing algorithm. This may be because of unsuitable values of the initial and final temperatures. Also, it may be due to the strength of the boundary penalty used in the great deluge algorithm. Table 1 shows the comparison of our final results compared to other published results in the literature.

Table 1. Comparison of results

Dataset	M1	M2	M3	M4	M5	M6
car-f-92	4.1	6.2	5.2	6.0	4.10	4.3
car-s-91	4.8	7.1	6.2	6.6	4.65	5.1
ear-f-83	36.0	36.4	45.7	**29.3**	37.05	35.1
hec-s-93	10.8	10.8	12.4	**9.2**	11.54	10.6
kfu-s-93	15.2	14.0	18.0	13.8	13.90	**13.5**
lse-f-91	11.9	10.5	15.5	**9.6**	10.82	10.5
sta-f-83	159.0	161.5	160.8	158.2	168.73	157.3
tre- s-92	8.5	9.6	10.0	9.4	8.35	8.4
uta-s-92	3.6	3.5	4.2	3.5	3.20	3.5
ute-s-92	26.0	25.8	29.0	**24.4**	25.83	25.1
yor-f-83	**36.2**	41.7	41.0	**36.2**	37.28	37.4

Dataset	M7	M8	M9	M10	M11	M12
car-f-92	4.4	4.2	4.4	**3.93**	4.84	4.56
car-s-91	5.4	4.8	5.2	**4.53**	5.41	5.29
ear-f-83	34.8	35.4	34.9	33.71	38.19	37.02
hec-s-93	10.8	10.8	10.3	10.83	12.72	11.78
kfu-s-93	14.1	13.7	**13.5**	13.82	15.76	15.81
lse-f-91	14.7	10.4	10.2	10.35	13.15	12.09
sta-f-83	134.9	159.1	159.2	151.52	**141.08**	160.42
tre- s-92	8.7	8.3	8.4	**7.92**	8.85	8.67
uta-s-92	-	3.4	3.6	**3.14**	3.54	3.57
ute-s-92	25.4	25.7	26.0	25.39	32.01	27.78
yor-f-83	37.5	36.7	**36.2**	36.35	40.13	40.66

Legend:
M1:The method presented in this paper; M2:Carter et al., 1996 [14]; M3:Di Gaspero and Schaerf, 2001 [17]; M4:Caramia et al., 2001 [15]; M5:Burke and Newall, 2003 [1]; M6:Merlot et al., 2003 [21]; M7:Casey and Thompson, 2003 [16]; M8:Burke et al., 2004 [11]; M9:Abdullah et al., 2006 [1]; M10:Yong and Petrovic, 2005 [23]; M11:Burke et al., 2006 [12]; M12:Asmuni et al., 2005 [6].

The best results are presented in bold. Following a personal communication with Jonathan Thompson [16], it appears that the result on the *sta-f-83* problem was achieved on a slightly different (and slightly smaller) version of the problem than the standard one. As such, the best result on the standard problem is that provided by M11. It is interesting to compare our results with the method of [1]. The reason why the technique in [1] outperforms our method even though we employed the same very large neighbourhood search approach is, we believe, because we limit the size of the improvement graph and this may limit the search towards a certain

region of the search space. However, it reduces the time taken when compared to the method of [1] and is still able to obtain better results than it on three out of the eleven datasets (there are also ties on three datasets). On the whole, our hybridisation algorithm works reasonably well across all problem instances and it does not perform worst in any of the comparisons.

Conclusions and Future Work

In this paper, we employed a multi start technique which hybridised the very large neighbourhood search approach with local search methods. We also employed a diversification strategy, by restarting the search process in order to find a better solution. Even though the experiments carried out in this work demonstrate that the method presented here only obtains one best result (and that represents a tie with two other methods), they show that the combination of the very large neighbourhood search methodology with local search can produce a feasible and good quality timetable and is also able to reduce the time taken to obtain a good solution compared to the time spent by our previous method from [1]. Moreover, it provides results that are consistently good across the all the benchmark problems.

References

1. Abdullah, S., Ahmadi, S., Burke, E.K., Dror, M. 2006. *Investigating Ahjuja-Orlin's Large Neighbourhood Search Approach for Examination Timetabling.* Accepted for publication in OR Spectrum, to appear 2006.
2. Ahuja, R.K., Magnanti, L.T., Orlin, J.B. 1993. *Network Flows: Theory, Algorithms, and Applications.* ISBN 1000499012, Prentice Hall, New Jersey, 133-165.
3. Ahuja, R.K., Orlin, J.B., Sharma, D. 2001. *Multiexchange Neighbourhood Search Algorithm for Capacitated Minimum Spanning Tree Problem.* Math Problem 91, 71-97.
4. Ahuja, R.K., Ozlem Ergun, Orlin, J.B., Abraham, O. Punnen. 2001. *A Survey Of Very Large-scale Neighborhood Search Techniques.* Discrete Applied Mathematics 123, 75-102.
5. Ahuja, R.K., Orlin, J.B., Sharma, D. 2003. *A Composite Neighbourhood Search Algorithm for Capacitated Spanning Tree Problem.* Operations Research Letters 31, 185-194.
6. Asmuni, H., Burke, E.K., and Garibaldi, J.M. 2005. *Fuzzy Multiple Ordering Criteria for Examination Timetabling.* In Edmund Burke and Michael Trick, editors, The Practice and Theory of Automated Timetabling. Lecture Notes in Computer Science 3616. Springer-Verlag, Berlin, 334-353.
7. Ayob, M., Kendall, G. 2003. *A Monte Carlo Hyper-Heuristic To Optimise Component Placement Sequencing For Multi Head Placement Machine.* Proc. of the International Conference on Intelligent Technologies, InTech'03, Chiang Mai, Thailand, 132-141.
8. Brelaz, D. 1979. *New Methods to Color the Vertices of A Graph.* Communication of the ACM 22(4), 251-256.
9. Burke, E.K., Petrovic, S. 2002. *Recent Research Direction in Automated Timetabling.* European Journal of Operational Research 140, 266-280.
10. Burke, E.K., Newall, J.P. 2003. *Enhancing Timetable Solutions with Local Search Methods.* In Edmund Burke and Patrick De Causmaecker, editors, The Practice and Theory of Automated Timetabling. Lecture Notes in Computer Science 2740. Springer-Verlag, Berlin, 195-206.
11. Burke, E.K., Bykov, Y., Newall, J.P., Petrovic, S. 2004. *A Time-Predefined Local Search Approach to Exam Timetabling Problem.* IIE Transactions 36(6), 509-528.
12. Burke, E.K., Amnon Meisels, Petrovic, S. and Qu, R. 2006. *A Graph-Based Hyper Heuristic for Timetabling Problems.* Accepted for publication in the European Journal of Operational Research, to appear 2006.
13. Carter, M.W., Laporte, G. 1996. *Recent Developments in Practical Examination Timetabling.* In Edmund Burke and Peter Ross, editors, The Practice and Theory of Automated Timetabling. Lecture Notes in Computer Science 1153. Springer-Verlag, Berlin, 3-21.
14. Carter, M.W., Laporte, G., Lee, S.Y. 1996. *Examination Timetabling: Algorithmic Strategies and Applications.* Journal of the Operation Research Society 74, 373-383.
15. Caramia, M., Dell'Olmo, P., Italiano, G.F. 2001. *New Algorithms for Examinations Timetabling.* In Naher, S., Wagner, D., editors, Algorithm Engineering 4th Int. Workshop, Proc. WAE 2000 Saarbrucken, Germany, September. Lecture Notes in Computer Science 1982. Springer-Verlag, Berlin, 230-241.
16. Casey, S., Thompson, J. 2003. *GRASPing the Examination Scheduling Problem.* In Edmund Burke and Patrick De Causmaecker, editors, The Practice and Theory of Automated Timetabling. Lecture Notes in Computer Science 2740. Springer-Verlag, Berlin, 232-244.
17. Di Gaspero, L., Schaerf, A. 2001. *Tabu Search Techniques for Examination Timetabling.* In Edmund Burke and Wilhelm Erbens, editors, The Practice and Theory of Automated Timetabling. Lecture Notes in Computer Science 2079. Springer-Verlag, Berlin, 104-117.
18. Dueck, G. 1993. *New Optimization Heuristics. The Great Deluge Algorithm and the Record-to-Record Travel.* Journal of Computational Physics 104, 86-92.
19. Higgins, A.J. 2001. *A Dynamic Tabu Search for Large-Scale Generalised Assignment Problems.* Computers and Operations Research, 28(10), 1039-1048.
20. Kirkpatrick, S., Gelatt, C.D., Vecchi, M.P. 1983. *Optimisation by Simulated Annealing.* Science, 220, 671-380.
21. Merlot T.G. Liam, Boland, N., Hughes, D.B., Stuckey, J. P. 2003. *A Hybrid Algorithm for the Examination Timetabling Problem.* In Edmund Burke and Patrick De Causmaecker, editors, The Practice and Theory of Automated Timetabling. Lecture Notes in Computer Science 2740. Springer-Verlag, Berlin, 207-231.
22. Petrovic, S., Burke, E.K. 2004. *University Timetabling.* Ch. 45 in the Handbook of Scheduling: Algorithms, Models, and Performance Analysis (eds. J. Leung), Chapman Hall/CRC Press.
23. Yong, Y., Petrovic, S. 2005. *A Novel Similarity Measure for Heuristic.* In Edmund Burke and Micheal Trick, editors, The Practice and Theory of Automated Timetabling. Lecture Notes in Computer Science 3616. Springer-Verlag, Berlin, 247-269.