USING A RANDOMISED ITERATIVE IMPROVEMENT ALGORITHM WITH COMPOSITE NEIGHBOURHOOD STRUCTURES FOR THE UNIVERSITY COURSE TIMETABLING PROBLEM

Salwani Abdullah[*] Edmund K. Burke[*] Barry McCollum[†]

[*]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
[†]Department of Computer Science, Queen's University Belfast, Belfast BT7 1NN United Kingdom
b.mccollum@qub.ac.uk

Abstract: The course timetabling problem deals with the assignment of a set of courses to specific timeslots and rooms within a working week subject to a variety of hard and soft constraints. Solutions which satisfy the hard constraints are called feasible. The goal is to satisfy as many of the soft constraints as possible whilst constructing a feasible schedule. In this paper, we present a composite neighbourhood structure with a randomised iterative improvement algorithm. This algorithm always accepts an improved solution and a worse solution is accepted with a certain probability. The algorithm is tested over eleven benchmark datasets (representing one large, five medium and five small problems). The results demonstrate that our approach is able to produce solutions that have lower penalty on all the small problems and two of the medium problems when compared against other techniques from the literature. However, in the case of the medium problems, this is at the expense of significantly increased computational time.

1. INTRODUCTION

In this paper, a randomised iterative improvement algorithm with composite neighbourhood structures for university course timetabling is presented. The approach is tested over eleven benchmark datasets that were introduced by Socha et al. (2002). The results demonstrate that our approach is capable of producing high quality solutions against others that appear in the literature. An extended abstract that describes this work was published in Abdullah et al. (2005b). The paper is organised as follows. The next section describes the university course timetabling problem in general and very briefly discusses the relevant timetabling literature. Section 3 presents a discussion of the literature on composite neighbourhood structures with a particular emphasis upon the employment of such structures in a variety of applications. Section 4 describes, in some detail, our randomised iterative improvement algorithm. The pseudo code of the implemented algorithm is also presented in this section. Experiments and results to evaluate the

performance of the heuristic are discussed in Section 5. Section 6 presents a brief summary of the paper.

2. THE UNIVERSITY COURSE TIMETABLING PROBLEM

Carter and Laporte (1998) defined course timetabling as:

> *"a multi-dimensional assignment problem in which students, teachers (or faculty members) are assigned to courses, course sections or classes; events (individual meetings between students and teachers) are assigned to classrooms and times"*

In university course timetabling, a set of courses is scheduled into a given number of rooms and timeslots within a week and, at the same time, students and teachers are assigned to courses so that the meetings can take place.

The course timetabling problem is subject to a variety of hard and soft constraints. Hard constraints need to be satisfied in order to produce a *feasible* solution. In this paper, we test our approach on the problem instances introduced by Socha et al. (2002) who present the following hard constraints:

- *No student can be assigned to more than one course at the same time.*
- *The room should satisfy the features required by the course.*
- *The number of students attending the course should be less than or equal to the capacity of the room.*
- *No more than one course is allowed to be assigned to a timeslot in each room.*

Socha et al. also present the following soft constraints that are equally penalised:

- *A student has a course scheduled in the last timeslot of the day.*
- *A student has more than 2 consecutive courses.*
- *A student has a single course on a day.*

The problem has

- A set of N courses, $e = \{e_1,\ldots,e_N\}$.
- 45 timeslots.
- A set of R rooms.
- A set of F room features.
- A set of M students.

The objective of this problem is to satisfy the hard constraints and to minimise the violation of the soft constraints.

In the last few years, several university course timetabling papers have appeared in the literature. Socha et al. (2002) presented a local search technique and an ant based methodology. They tested their approach on eleven test problems. These eleven problems were produced by Paechter's[1] course timetabling test instance generator and are the instances used to evaluate the method described in this paper. Since then, several papers have appeared which have tested their results on the same instances. Burke *et al.*

[1] http://www.dcs.napier.ac.uk/~benp/

(2003a) introduced a tabu-search hyperheuristic where a set of low level heuristics compete with each other. The goal was to raise the level of generality of search systems and the method was tested on a nurse rostering problem in addition to course timetabling. A graph hyper-heuristic was presented by Burke *et al*. (2006) where, within a generic hyper-heuristic framework, a tabu search approach is employed to search for permutations of constructive heuristics (graph colouring heuristics). Abdullah *et al.* (2005a) employed a variable neighbourhood search with a fixed tabu list which is used to penalise the unperformed neighbourhood structures. Other papers which test against these instances can be seen in Socha et al. (2003) who discuss ant algorithm methodologies at length and Rossi-Doria et al. (2003) who compare several metaheuristic methods.

In addition to the problem instances introduced by Socha et al (2002), Paechter's generator was also used to produce the problem sets for a timetabling competition held in 2002 (see http://www.idsia.ch/Files/ ttcomp2002). They generated twenty instances for the competition itself and another three unseen instances to further check the performance of the algorithms. Some papers have recently appeared which test their methodologies on these competition problems. Kostuch (2005) presented a three phase approach which employs Simulated Annealing. This approach won the competition mentioned above and had 13 best results of the 20 instances in the competition. Burke et al. (2003b) employed a Great Deluge method which generated 7 best results out of the 20 competition problems mentioned above. This method also produced some poor results on some problems which is why it came 3rd in the competition (because the competition used an average measure). The hybrid local search methodology which came 4th in the competition is described in Di Gaspero and Schaerf (2006). Arntzen and Løkketangen (2004) developed a tabu search method which came 5th in the competition. Lewis and Paechter (2004) designed several crossover operators and tested them against the competition datasets. They concluded that their results were not "state of the art". A hybrid metaheurstic approach has recently appeared in the literature which is tested on these competition problems and which produces improved results to those generated by the competition (Chiarandini et al. 2006). Also, Kostuch and Socha (2004) investigated the possibility of using a statistical model to predict the difficulty of timetabling problems and they employed the competition instances.

In 2005, Lewis and Paechter used the same instance generator to create another sixty "hard" test instances (Lewis and Paechter 2005). They tested their grouping genetic algorithm on these sixty instances but were concerned only with feasibility.

In addition to the university course timetabling papers which have used problems produced by Paechter's generator, several other articles have recently appeared which represent case studies on real university timetabling instances. Examples include Avella and Vasil'Ev (2005), Daskalaki et al. (2004), Dimopoulou and Miliotis (2004) and Santiago-Mozos et al. (2005).

Other aspects of university course timetabling have been widely discussed in the literature over the last thirty years or so. A survey of practical approaches to the problem, up to 1998, can be seen in Carter and Laporte (1998). The following papers represent a comprehensive list of surveys and overviews of educational timetabling (which include issues related to University course timetabling) i.e. Bardadym (1996), Burke et al. (1997), Burke and Petrovic (2002), Burke et al. (2004), Carter (2001),

Petrovic and Burke (2004), Schaerf (1999), de Werra (1985) and Wren (1996).

3. COMPOSITE NEIGHBOURHOOD STRUCTURES: RESEARCH AND DEVELOPMENTS

A composite neighbourhood structure subsumes two or more neighbourhood structures. The advantage of combining several neighbourhood structures is that it helps to compensate against the ineffectiveness of using each type of structure in isolation (Grabowski and Pempera, 2000 and Liaw 2003). For example, a solution space that is easily accessible by insertion moves may be difficult to reach using swap moves. Some examples of composite neighbourhood structures that are available in the literature are discussed here.

Grabowski and Pempera (2000) applied a composite neighbourhood structure for sequencing jobs in a production system that consists of exchanges and the insertion of elements. Gopalakrishnan et al. (2001) used three moves (swap, add and drop) in a tabu search heuristic for preventive maintenance scheduling. The decision on which move to use depends on the current state of the search. The interaction of the moves makes it possible to carry out a strategic search. The computational results show that the approach can improve the solution quality when compared to the local heuristics employed by Gopalakrishnan et al. (1997).

Liaw (2003) also used a composite neighbourhood structure in the tabu search approach for the two-machine preemptive open shop scheduling problem. The tabu search switches to the other neighbourhood structures (between an insertion move that shifts one job from its current position to a new position and a swap move that exchanges the position of two jobs) after a number of iterations without any improvements. Computational experiments have shown that this scheme significantly improves the performance of tabu search in terms of solution quality. The neighbourhood used in Ouelhadj (2003) has a composite structure where the tabu search approach, applied to the dynamic scheduling of a hot strip mill agent, employed three neighbourhood schemes (swap, shift and inversion) alternately. Computational experiments showed that the composite structure improves the solution quality compared with tabu search using a single neighbourhood. Another example of a composite neighbourhood structure was presented by Landa Silva (2003). He employed several neighbourhood structures (relocate, swap and interchange) in different metaheuristics (iterative improvement, simulated annealing and tabu search) and applied this to a space allocation problem in an academic institution.

Bilge et al. (2004) used a “hybrid” neighbourhood structure in a tabu search algorithm for the parallel machine total tardiness problem. The “hybrid” structure consists of the complete “insert neighbourhood” with the addition of a partial “swap neighbourhood”. In an insert move operation, two jobs are identified and the first job is placed in the location that precedes the location of the second job. Then, a swap move places each job in the location that was previously occupied by the other job.

4. THE RANDOMISED ITERATIVE IMPROVEMENT ALGORITHM

This algorithm presented here always accepts an improved solution and a worse solution is accepted with a certain probability.

4.1 The Neighbourhood Structures

The different neighbourhood structures and their explanation can be outlined as follows:

N1: Select two courses at random and swap timeslots.

N2: Choose a single course at random and move to a new random feasible timeslot.

N3: Select two timeslots at random and simply swap all the courses in one timeslot with all the courses in the other timeslot.

N4: Take 2 timeslots (selected at random), say t_i and t_j (where $j>i$) where the timeslots are ordered t_1, t_2, …, t_{45}. Take all the exams in t_i and allocate them to t_j. Now take the exams that were in t_j and allocate them to t_{j-1}. Then allocate those that were in t_{j-1} to t_{j-2} and so on until we allocate those that were in t_{i+1} to t_i and terminate the process.

N5: Move the highest penalty course from a random 10% selection of the courses to a random feasible timeslot.

N6: Carry out the same process as in N5 but with 20% of the courses.

N7: Move the highest penalty course from a random 10% selection of the courses to a new feasible timeslot which can generate the lowest penalty cost.

N8: Carry out the same process as in N7 but with 20% of the courses.

N9: Select one course at random, select a timeslot at random (distinct from the one that was assigned to the selected course) and then apply the kempe chain from Thompson and Dowsland (1996).

N10: This is the same as N9 except the highest penalty course from 5% selection of the courses is selected at random.

N11: Carry out the same process as in N9 but with 20% of the courses.

4.2 The Algorithm

In the approach presented in this paper, a set of the neighbourhood structures outlined in subsection 4.1 is applied. The hard constraints are never violated during the timetabling process.

The pseudo code for the algorithm implemented in this paper is given in Figure 1. The algorithm starts with a feasible initial solution which is generated by a constructive heuristic as discussed in Abdullah et al. (2005a). Let K be the total number of neighbourhood structures to be used in the search (K is set to be 11 in this implementation) and $f(Sol)$ is the quality measure of the solution Sol. At the start, the best solution, Sol_{best} is set to be Sol. In a *do-while* loop, each neighbourhood i where $i \in \{1,\ldots,K\}$ is applied to Sol to obtain $TempSol_i$. The best solution among $TempSol_i$ is identified, and is set to be the new solution Sol^*. If Sol^* is better than the best solution in hand, Sol_{best}, then Sol^* is accepted. Otherwise, the exponential Monte Carlo acceptance criterion is applied. This accepts a worse solution with a

certain probability. The criterion is discussed in Ayob and Kendall (2003). The new solution *Sol** is accepted if the generated random number in [0,1], *RandNum*, is less than the probability which is computed by $e^{-\delta}$ where δ is the difference between the cost of the old and new solutions (i.e. $\delta = f(Sol^*) - f(Sol)$). The Monte Carlo method will exponentially increase the acceptance probability if δ is small. The process is repeated and stops when the termination criteria is met (in this work the termination criteria is set as the number of evaluations i.e. 200000 evaluations or when the penalty cost is zero).

```
Set the initial solution Sol by employing a
constructive heuristic;
Calculate initial cost function f(Sol);
Set best solution Sol_best ← Sol;
do while (not termination criteria)
     for i = 1 to K where K is the total number of
     neighbourhood structures
       Apply neighbourhood structure i on Sol, TempSol_i;
       Calculate cost function f(TempSol_i);
     end for
     Find the best solution among TempSol_i where i ∈
     {1,…,K} call new solution Sol*;
     if (f(Sol*) < f(Sol_best))
        Sol ← Sol*;
        Sol_best ← Sol*;
     else
        Apply an exponential Monte Carlo where:
        δ = f(Sol*) -  f(Sol));
        Generate RandNum, a random number in [0,1];
        if (RandNum < e^-δ )
             Sol ← Sol*;
     end if
end do
```

Figure 1. The pseudo code for the randomised iterative improvement algorithm

5. EXPERIMENTS AND RESULTS

The approaches are coded in Microsoft Visual C++ version 6 under Windows. All experiments were run on an Athlon machine with a 1.2GHz processor and 256 MB RAM running under Microsoft Windows 2000 version 5. We evaluate our results on the instances taken from Socha et al (2002) and which are available at http://iridia.ulb.ac.be/~msampels/tt.data/. We employed the same initial solutions as in Abdullah et al. (2005a). The experiments were run for 200000 iterations which takes approximately eight hours for each of the medium datasets and at most 50 seconds for the small datasets. Note that course timetabling is a problem that is usually tackled several months before the schedule is required. An eight hours run for course timetabling is perfectly acceptable in a real world environment. This is a scheduling problem where the time taken to solve the problem is not critical. The emphasis in this paper is on generating good quality solutions and the price to pay for this can be taken as being a large amount of computational time.

The experiments for the course timetabling problem discussed in this paper were tested on the benchmark course timetabling problems proposed

by the Metaheuristics Network that need to schedule 100-400 courses into a timetable with 45 timeslots corresponding to 5 days of 9 hours each, whilst satisfying room features and room capacity constraints. They are divided into three categories: *small*, *medium* and *large*. We deal with 11 instances: 5 *small*, 5 *medium* and 1 *large*. The parameter values defining the categories are given in Table 1.

Table 1. The parameter values for the course timetabling problem categories

Category	*small*	*medium*	*large*
Number of courses	100	400	400
Number of rooms	5	10	10
Number of features	5	5	10
Number of students	80	200	400
Maximum courses per student	20	20	20
Maximum student per courses	20	50	100
Approximation features per room	3	3	5
Percentage feature use	70	80	90

The best results out of 5 runs obtained are presented. Table 2 shows the comparison of the approach in this paper with other available approaches in the literature on the five small problems. Table 3 illustrates our comparison on the medium/large problems. The term "x%Inf." in Table 3 indicates a percentage of runs that failed to obtain feasible solutions.

The best results are presented in bold in both tables. Note that the only methods that were able to obtain feasible solutions for the large problem were the ant method (Socha et al, 2002) and the graph based hyper-heuristic (Burke et al, 2006) with the ant method being better.

It can be seen that the randomised iterative improvement algorithm has better results than Abdullah et al. (2005a) on all five medium datasets with the same (best result) penalty cost for the *small* instances. Our approach has better results than the local search method (Socha et al, 2002) on three of the medium instances and on all five of the small datasets. Our method has higher quality results when compared against the ant approach (Socha et al, 2002) on four of the small problems, with both approaches being able to obtain zero penalty on the other. Our algorithm gets better results than the ant technique on two of the medium instances. The iterative improvement approach is has better penalty values than the tabu search hyper-heuristic (Burke et al. 2003a) on three of the small datasets and both methods get zero penalty on the other two. It was better values on just two of the medium sets. The iterative approach obtained better results than the graph based hyper-heuristic (Burke et al. 2006) on all datasets except the large one.

Note that our approach has the very best results across seven of the eleven datsets (although it does perform very poorly on the large one). It is particularly effective on the *small* problems, taking approximately 50 seconds to obtain zero penalties as opposed to, for example, the algorithms of (Socha et al) which take 90 seconds. It is quite effective on the *medium* problems but at the expense of a high level of computational time. It takes our algorithm about 8 hours to produce these solutions for the medium problems whereas, for example, it takes the (Socha et al, 2002) methods 900 seconds (15 minutes). The need for the long run time is probably due to some neighbourhood structures in our method being less effective on this type of problem.

Table 2. Comparison of results on the small datasets

Data Set	Initial Solution	Randomised Iterative Improvement Algorithm		VNS with tabu (Abdullah et al. 2005a) (Best)	Local search (Socha et al. 2002) (Median)	Ant Algorithm (Socha et al. 2002) (Median)	Tabu-based hyper-heuristic (Burke et al. 2003a) (Best)	Graph hyper-heuristic (Burke et al. 2006) (Best)
		Best	Median					
s1	261	0	**0**	**0**	8	1	1	6
s2	245	0	**0**	**0**	11	3	2	7
s3	232	0	**0**	**0**	8	1	**0**	3
s4	158	0	**0**	**0**	7	1	1	3
s5	421	0	**0**	**0**	5	**0**	**0**	4

Table 3. Comparison of results on the medium/large datasets

Data Set	Initial Solution	Randomised Iterative Improvement Algorithm		VNS with tabu (Abdullah et al. 2005a) (Best)	Local search (Socha et al. 2002) (Median)	Ant Algorithm (Socha et al. 2002) (Median)	Tabu-based hyper-heuristic (Burke et al. 2003a) (Best)	Graph hyper-heuristic (Burke et al. 2006) (Best)
		Best	Median					
m1	914	242	245	317	199	195	**146**	372
m2	878	161	**162.6**	313	202.5	184	173	419
m3	941	265	267.8	357	77.5% Inf.	**248**	267	359
m4	865	181	183.6	247	177.5	**164.5**	169	348
m5	780	151	**152.6**	292	100% Inf.	219.5	303	171
l	100% Inf	-	-	100% Inf.	100% Inf.	**851.5**	80% Inf. 1166	1068
Data Set Key: l = large, m1 = medium1, m2 = medium 2 and so on.								

Figures 2 and 3 show the behaviour of the randomised iterative improvement algorithm applied to the *small1* and *medium5* datasets, respectively. In all the figures, the x-axis represents the number of evaluations whilst the y-axis represents the penalty cost. The graphs illustrate the exploration of the search space. The curves move up and down because worse solutions are accepted with a certain probability in order to escape from local optima. The penalty cost can be quickly reduced at the beginning of the search where there is (possibly) a lot of room for improvement. It is believed that better solutions can be obtained in these experiments (particularly on the smaller problems) because the composite neighbourhood structures offer flexibility for the search algorithm to explore different regions of the solution space. The graphs for the *small* datasets show that our algorithm is able to obtain zero penalties in less than 1500 evaluations which is an improvement upon Burke et al. (2003a) which set the number of evaluations at 12000 for *small* datasets.

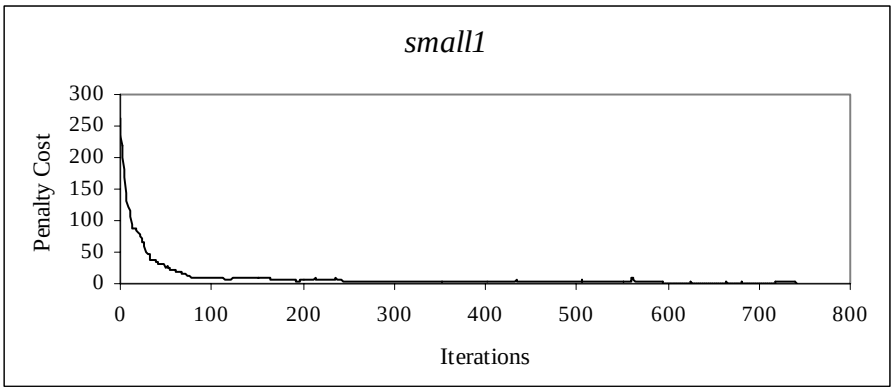

Figure 2. The behaviour of the randomised iterative improvement algorithm on the *small1* dataset

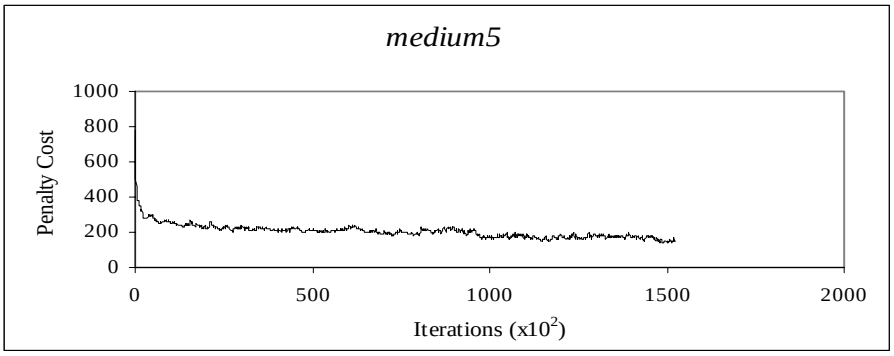

Figure 3. The behaviour of the randomised iterative improvement algorithm on the *medium5* dataset

Figures 4 and 5 show the frequency charts of the neighbourhood structures that have been selected to be used by the randomised iterative improvement algorithm for the *small* and *medium* datasets, respectively. The x-axis represents the datasets while the y-axis represents the frequency of the neighbourhood structures being employed throughout the search.

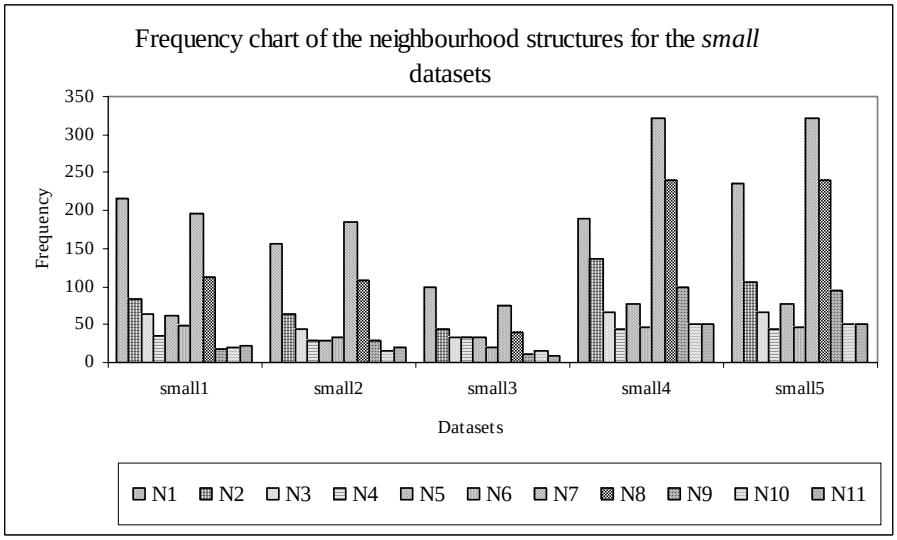

Figure 4. Frequency of the neighbourhood structures used for the *small* datasets

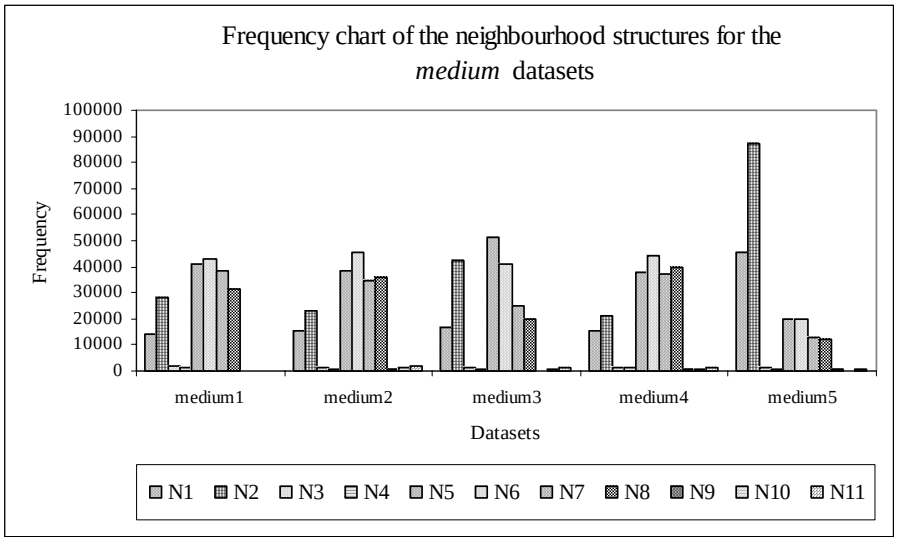

Figure 5. Frequency of the neighbourhood structures used for the *medium* datasets

It can be seen, from Figure 4, that the neighbourhood structures "N1", "N2", "N7" and "N8" are the most popular structures used in the algorithm for *small* datasets. The popular structures for the *medium* datasets are "N1", "N2", "N5", "N6", "N7" and "N8" as shown in Figure 5. This illustrates that the most popular neighbourhood structures that are being supplied to the randomised iterative improvement algorithm are almost the same between the *small* and *medium* datasets (i.e. "N1", "N2", "N7" and "N8"). However, as the problem gets larger, there may be fewer and more sparsely distributed solution points (feasible solutions) in the solution space since too many courses are conflicting with each other. Thus, the approach may need extra neighbourhood structures (i.e. "N5" and "N6" in this case) to force the search algorithm to diversify its exploration of the solution space by moving from one neighbourhood structure to another. Further investigation was carried out to support the claim that the composite neighbourhood structure performs better than the single neighbourhood structure by employing selected neighbourhood structures separately i.e. "N1", "N2", "N5", "N6", N7" and "N8" (which are the most popular neighbourhood structures used for the *small* and *medium* datasets). The *small* datasets are able to obtain zero penalty in less than 1500 evaluations. Thus, for the experiments carried out here, the number of evaluations for the *small* datasets is set as equal to the number of evaluations where the best solutions are obtained (i.e. 873, 707, 413, 1012 and 1329 evaluations for *small1*, *small2*, *small3*, *small4* and *small5*, respectively). The number of evaluations for the *medium* datasets remains the same. Table 4 gives the comparison of the performance of variants of the randomised iterative improvement algorithm in terms of penalty cost (objective function value). The results demonstrate that the algorithm with composite neighbourhood structures is uniformly the best in terms of penalty cost compared to other randomised iterative improvement algorithm variants.

Table 4. Comparison of the performance of the randomised iterative improvement algorithm on single and composite neighbourhood structures

Dataset	Initial solution	Randomised iterative improvement algorithm neighbourhoods						
		N1	N2	N5	N6	N7	N8	Composite
small1	261	76	21	26	54	5	8	**0**
small2	245	64	27	47	59	9	6	**0**
small3	232	68	45	69	33	6	18	**0**
small4	158	63	39	44	18	5	9	**0**
small5	421	112	33	49	64	7	12	**0**
medium1	914	381	345	548	713	539	701	**242**
medium2	878	364	337	556	675	555	643	**161**
medium3	941	420	401	731	773	764	774	**265**
medium4	865	332	317	549	615	546	603	**181**
medium5	780	414	355	650	685	702	699	**151**
large	100%Inf.	-	-	-	-	-	-	-

Figures 6 and 7 illustrate the behaviour of the randomised iterative improvement algorithm using a single neighbourhood structure compared to the composite neighbourhood structure applied on the *small1* and *medium5* datasets, respectively.

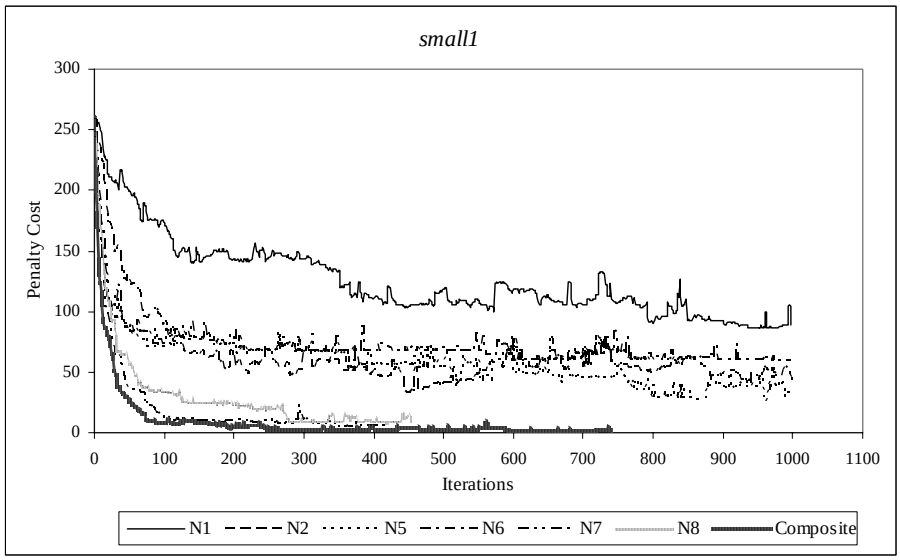

Figure 6. The behaviour of the randomised iterative improvement algorithm using single and composite neighbourhood structures applied on the *small1* dataset

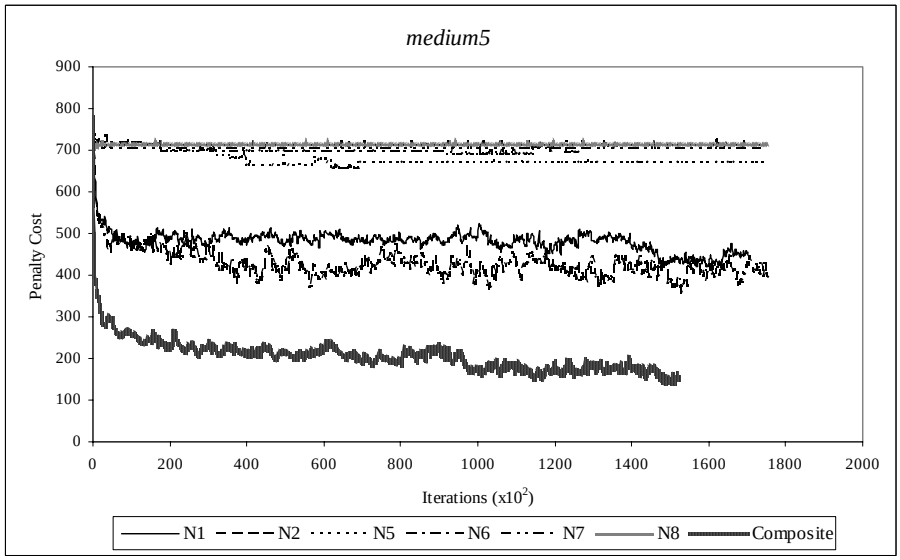

Figure 7. The behaviour of the randomised iterative improvement algorithm using single and composite neighbourhood structures applied on the *medium5* dataset

The diagrams show the convergence of the penalty cost of the algorithm for *small1 and medium5* for a number of evaluations for which the best solution is found. It can be seen that the randomised iterative improvement algorithm with the composite neighbourhood is significantly better than other variants with single neighbourhood in terms of solution quality given the same number of evaluations. All the other problems of the family have the same behaviour as in Figures 6 and 7.

6. CONCLUSION AND FUTURE WORK

This paper has focused on investigating a composite neighbourhood structure with a randomised iterative improvement algorithm for the university course timetabling problem. Preliminary comparisons indicate that this algorithm is competitive with other approaches in the literature. Indeed, it produced seven solutions that were better than or equal to the

published penalty values on these eleven instances although it did require significant computational time for the medium/large problems. It is an approach that is particularly effective on smaller problems. Further experiments were carried out to demonstrate that it is more effective to employ composite neighbourhood structures rather than a single neighbourhood structure because of the different ways of search that are represented by various neighbourhood structures.

Future research will be aimed at exploring how the algorithm could intelligently select the most suitable neighbourhood structures according to the characteristics of the problems. Another direction of future research will investigate the integration of a population-based approach with a local search method.

Acknowledgement

This work has been supported by the Public Services Department of Malaysia (JPA) and the University Kebangsaan Malaysia (UKM).

REFERENCES

[1] Abdullah S, Burke EK and McCollum B (2005a) An investigation of variable neighbourhood search for university course timetabling. In: Proceedings of *The 2nd Multidisciplinary International Conference on Scheduling: Theory and Applications (MISTA 2005)*, New York, USA, July 18th-21st, pp 413-427.

[2] Abdullah S, Burke EK and McCollum B (2005b). Using a Randomised Iterative Improvement Algorithm with Composite Neighbourhood Structures for University Course Timetabling. In: Proceedings of the 6th Metaheuristics International Conference (MIC 05), Vienna, Austria, August 22nd-26th, in CD-ROM, 2005.

[3] Avella P and Vasil'Ev I (2005) A Computational Study of a Cutting Plane Algorithm for University Course Timetabling. *Journal of Scheduling* 8(6), pp 497-514.

[4] Ayob M and Kendall G (2003) A monte carlo hyper-heuristic to optimise component placement sequencing for multi head placement machine. *Proceedings of the International Conference on Intelligent Technologies, InTech'03*, Thailand, Dec 17th-19th, pp 132-141.

[5] Bardadym VA (1996) Computer-aided school and university timetabling: A new wave. Practice and Theory of Automated Timetabling V (eds. Burke and Ross), Springer Lecture Notes in Computer Science Volume 1153, pp 22-45.

[6] Bilge Ü, Kiraç F, Kurtulan M and Pekgün P (2004) A tabu search algorithm for the parallel machine total tardiness problem. *Computers & Operations Research* 31(3), pp 397-414.

[7] Burke EK, Jackson KS, Kingston JH and Weare RF (1997) Automated timetabling: The state of the art, *The Computer Journal* 40(9), pp 565-571.

[8] Burke EK and Petrovic S (2002) Recent research direction in automated timetabling. *European Journal of Operational Research* 140, pp 266-280.

[9] Burke EK, Kendall G and Soubeiga E (2003a) A tabu search hyperheuristic for timetabling and rostering. *Journal of Heuristics* 9(6), pp 451-470.

[10] Burke EK, Bykov Y, Newall J and Petrovic S (2003b) A Time-Predefined Approach to Course Timetabling, *Yugoslav Journal of Operational Research* (YUJOR), Vol 13, No. 2, pp 139-151.

[11] Burke, EK, Kingston J and De Werra D (2004) *Applications to Timetabling*, in the Handbook of Graph Theory, (eds. Gross J and Yellen J), Chapman Hall/CRC Press, pp 445-474,

[12] Burke, E.K., McCollum, B., Meisels, A., Petrovic, S. and Qu, R., A Graph-Based Hyper Heuristic for Educational Timetabling Problems, *European Journal of Operational Research* 176(1), 1 January 2007, pp 177-192.

[13] Carter MW (2001) Timetabling, encyclopedia of operations research and management science (eds Gass and Harris), Kluwer, pp 833-836.

[14] Carter MW and Laporte G (1998) Recent developments in practical course timetabling. Practice and Theory of Automated Timetabling V (eds. Burke and Carter), Springer Lecture Notes in Computer Science Volume 1408, pp 3-19.

[15] Chiarandini M, Birattari M, Socha K and Rossi-Doria O (2006) An effective hybrid algorithm for university course timetabling. *Journal of Scheduling* 9(5), pp 403-432.

[16] Daskalaki S, Birbas T and Housos H (2004) An integer programming formulation for a case study in university timetabling. *European Journal of Operational Research* 153(1), pp 117-135.

[17] Dimopoulou M and Miliotis P (2004) An automated university course timetabling system developed in a distributed environment: A case study. *European Journal of Operational Research* 153(1), pp 136-147.

[18] de Werra D (1985) An introduction to timetabling. *European Journal of Operational Research* 19, pp 151-162.

[19] Di Gaspero L and Schaerf A (2006) Neighborhood portfolio approach for local search applied to timetabling problems. *Journal of Mathematical Modeling and Algorithms*, 5(1), pp 65-89.

[20] Gopalakrishnan M, Ahire SL and Miller DM (1997) Maximising the effectiveness of a preventive maintenance system: An adaptive modeling approach. *Management Science*, 43(6), pp 827-840.

[21] Gopalakrishnan M, Mohan S and He Z. (2001). A tabu search heuristic for preventive maintenance scheduling. *Computers* & *Industrial Engineering*, 40, pp 149-160.

[22] Grabowski J and Pempera J (2000) Sequencing of jobs in some production system: Theory and methodology. *European Journal of Operational Research*, 125, pp 535-550.

[23] Kostuch P (2005) The university course timetabling problem with a three-phase approach. Practice and Theory of Automated Timetabling V (eds. Burke and Trick), Springer Lecture Notes in Computer Science Volume 3616, pp 109-125.

[24] Kostuch P and Socha K (2004), Hardness Prediction for the University Course Timetabling Problem, Proceedings of the Evolutionary Computation in Combinatorial Optimization (EvoCOP 2004), Coimbra, Portugal, April 5-7, 2004, Springer Lecture Notes in Computer Science Volume 3004, pp 135-144.

[25] Landa Silva JD (2003) Metaheuristic and Multiobjective Approaches for Space Allocation. *PhD Thesis*, Department of Computer Science, University of Nottingham, United Kingdom.

[26] Lewis R and Paechter B (2004) New crossover operators for timetabling with evolutionary algorithms. *Proceedings of the* 5^{th} *International Conference on Recent Advances in Soft Computing* (ed. Lotfi), UK, December 16^{th}-18^{th}, pp 189-194.

[27] Lewis R and Paechter B (2005) Application of the groping genetic algorithm to university course timetabling. Evolutionary Computation in Combinatorial Optimisation (eds. Raidl and Gottlieb), Springer Lecture Notes in Computer Science Volume 3448, pp 144-153.

[28] Liaw CF (2003) An efficient tabu search approach for the two-machine preemptive open shop scheduling problem. *Computers* & *Operations Research* 30(14), pp 2081-2095.

[29] Ouelhadj D (2003) A multi-agent system for the integrated dynamic

scheduling of steel production. PhD Thesis, Department of Computer Science, University of Nottingham, United Kingdom.

[30] Petrovic S and Burke EK (2004) University timetabling, Ch. 45 in the *Handbook of Scheduling: Algorithms, Models, and Performance Analysis* (eds. J. Leung), Chapman Hall/CRC Press.

[31] Rossi-Doria O, Samples M, Birattari M, Chiarandini M, Dorigo M, Gambardella LM, Knowles J, Manfrin M, Mastrolilli M, Paechter B, Paquete L and Stützle T (2003). A comparison of the performance of different meta-heuristics on the timetabling problem. Practice and Theory of Automated Timetabling V (eds. Burke and De Causmaecker), Springer Lecture Notes in Computer Science Volume 2740, pp 329-354.

[32] Santiago-Mozos R, Salcedo-Sanz S, DePrado-Cumplido M, Carlos Bousoalz C. A two-phase heuristic evolutionary algorithm for personalising course timetables: A case study in a Spanish university. *Computers and Operations Research* 32, pp 1761-1776.

[33] Schaerf A (1999) A survey of automated timetabling. *Artificial Intelligence Review* 13(2), pp 87-127.

[34] Socha K, Knowles J and Samples M (2002) A max-min ant system for the university course timetabling problem. *Proceedings of the 3rd International Workshop on Ant Algorithms (ANTS 2002)*, Springer Lecture Notes in Computer Science Volume 2463, pp 1-13.

[35] Socha K, Sampels M and Manfrin M (2003) Ant algorithms for the university course timetabling problem with regard to the state-of-the-art. *Proceedings of 3rd European Workshop on Evolutionary Computation in Combinatorial Optimization (EvoCOP'2003)*, UK, April 14th-16th, Springer Lecture Notes in Computer Science Volume 2611, pp 335-345.

[36] Thompson J and Dowsland K (1996) Various of simulated annealing for the examination timetabling problem. *Annals of Operational Research* 63, pp 105-128.

[37] Wren A (1996) Scheduling, timetabling and rostering – A special relationship? Practice and Theory of Automated Timetabling V (eds. Burke and Ross), Springer Lecture Notes in Computer Science Volume 1153, pp 46-75.